Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
December 2008, 15(4): 51–54
www.sciencedirect.com/science/journal/10058885

The Journal of China Universities of Posts and Telecommunications
www.buptjournal.cn/xben

Multi-branch query tree protocol for solving RFID tag collision problem

JIAO Chuan-hai (✉), WANG Ke-ren

Laboratory 304 of Communications and Information Systems, Electronic Engineering Institute, Hefei 230037, China

Abstract

The technology of anti-collision is a key point in radio frequency identification (RFID) system. To avoid data collision, there are two approaches: ALOHA based algorithm and binary tree (BT) based algorithm. However, these cannot solve the collision problem completely, especially when the tag quantity is big and the tag ID is long. In this article, we present a multi-branch query tree (MBQT) protocol based on balanced incomplete block design (BIBD) code, and use 16-bit vectors derived from the BIBD as query prefix symbols of RFID reader. Compared with the conventional anti-collision algorithm, the theoretic analysis and simulation show that the proposed protocol improves the identification efficiency.

Keywords RFID, BIBD, MBQT

1 Introduction

RFID is a contact-less automatic identification system based on wireless radio frequency (RF) communication techniques. RFID has high potentials such as supply chain management, access control with identification card, and asset tracking system [1]. Accordingly, RFID techniques are considered the most hopeful ones and have become a research hotspot at present.

Generally, the RFID system is composed of a reader and tags, where the RFID reader reads data from each RF tag. When there is more than one tag within the interrogation zone of a reader, all the tags may send data at the same time, that is, more than one tag occupies the same RF communication channel simultaneously, which may lead to mutual interference. This event causes data loss that is referred to as a collision. One of the main disadvantages of the RFID system is its low tag identification efficiency by tag collision.

Therefore, we have to reduce tag collision to increase tag identification efficiency. So far, several tag anti-collision algorithms have been proposed, among which the ALOHA based algorithm and the binary search tree based algorithm [2–8] are the most widely used ones. However, they cannot solve the collision problem perfectly, especially when the tag quantity is large and the tag ID is long.

In this article, we propose an MBQT protocol based on the BIBD code, and use 16 bit vectors derived from the BIBD as RFID reader query prefix symbols. The remaining sections of this article are organized as follows: In Sect. 2, we review existing tag anti-collision protocols including the probabilistic algorithm and the deterministic algorithm. Section 3 describes the multi-branch query tree protocol based on the BIBD code. A simple instance is described to show how the MBQT protocol works in Sect. 4. Section 5 shows the computer simulation result and some suggestions for improving the protocol are proposed in Sect. 6. Section 7 concludes the article.

2 Tag anti-collision protocols review

Nowadays, there are two types of tag anti-collision protocols that are most widely used:

1) Probabilistic algorithm, such as ALOHA algorithm, slotted ALOHA algorithm, frame slotted ALOHA algorithm, dynamic frame slotted ALOHA algorithm, etc.

2) Deterministic algorithm, such as BT based algorithm and query tree (QT) algorithm.

2.1 Probabilistic algorithm

The simplest probabilistic algorithm is the ALOHA

Received date: 10-05-2008
Corresponding author: JIAO Chuan-hai, E-mail: jhb_8319@163.com

procedure, where tags send their data randomly after being activated by the reader. When collision occurs, they wait for random time and retransmit. In slotted ALOHA algorithm, time is divided into numerous small discrete time slots, and a tag can send its data at the beginning of its pre-specified slot. In this way, there will be no partial collision but successful transmission or complete collision for tags to send data. Although the slotted ALOHA algorithm can enhance channel utilization and throughput, it cannot guarantee reasonable response time when there are several tags near the reader. To guarantee the response time, the frame slotted ALOHA (FSA) algorithm is proposed based on the slotted ALOHA algorithm. In this scheme, N slots constitute a frame and each tag chooses a slot to transmit data in the frame. As the frame size N becomes larger, the probability of collision becomes lower, but the identification time becomes longer.

Dynamic frame slotted ALOHA (DFSA) algorithm changes the frame size for efficient tag identification [3–4,8]. To determine the frame size, it uses information such as the number of slots used to identify the tag and the number of the slots collided, etc.

However, the probabilistic algorithm or the ALOHA based algorithm cannot prevent collisions perfectly. Meanwhile, there will be tag starvation phenomenon, where a tag collapses repeatedly, making the tag not be identified for a long time.

2.2 Deterministic algorithm

The deterministic algorithm can be categorized into the BT based algorithm and the QT algorithm. Both thses require that all tags respond to the query command at the same time and thus the reader can identify the corrupted bits. In the BT algorithm, all tags have a register to save previous inquiring result, whereas, it has complicated tag implementation. The QT algorithm does not require the tag's own counter. Instead of using the counter, the reader transmits prefix and the tags respond to their remaining bits if they match the prefix. The QT algorithm is a memory-less protocol and the tags need lower functionality. However, it is slower than the BT algorithm for tag identification [9–10].

3 Multi-branch query tree protocol

In this article, we propose the MBQT protocol using multiple vector symbols instead of single binary symbol (0/1) as reader prefix. Compared with the traditional QT protocol, the MBQT protocol has the strong advantage of faster identification, low power consumption, and robustness. To solve the problem of small number of tags, we use 64 bit ID, which is composed of four 16 bit BIBD codes. Each 16 bit BIBD code is based on (16,4,1)-BIBD, and it can support 20^4 users. To further increase the number of supported tags, we can use the hybrid code scheme, where a small part uses the BIBD scheme to be compatible with electronic product code (EPC).

3.1 BIBD code and prefix symbol design

BIBD code belongs to combinatorics, which is one embranchment of mathematics. Combinatorial design is to choose a set of subsets, which satisfies some specific character from a given set.

Definition 1 Suppose $\boldsymbol{S}=\{s_1,s_2,...,s_v\}$ is a v-element set and $\boldsymbol{D}=\{\boldsymbol{D}_1,\boldsymbol{D}_2,...,\boldsymbol{D}_m\}$ is the set of k-element subset of $\boldsymbol{S}$. Let r be the number of k-element subset including a certain arbitrary element, such that each pair of element of $\boldsymbol{D}$ occurs together in exactly λ blocks. We call the combinatorial design $\{\boldsymbol{S},\boldsymbol{D}\}$ as the balanced incomplete block design, marked as (v,m,r,k,λ)-BIBD or (v,k,λ)-BIBD.

Actually, every block of (v,k,λ)-BIBD is picked out from the whole combination blocks that choose k elements from v elements arbitrarily. Therefore, we get the following theorem:

Theorem 1

1) The number of the whole combination blocks that choose k elements from v elements arbitrarily is:

$$m_1=C_v^k=\frac{v!}{(v-k)!k!} \tag{1}$$

2) The time of each pair of elements appearing in the block simultaneity is:

$$\lambda_1=C_{v-2}^{k-2}=\frac{(v-2)!}{(v-k)!(k-2)!} \tag{2}$$

3) If we request the time of each pair of elements appearing in the block simultaneity as $\lambda(\lambda\leqslant\lambda_1)$, then we can see that the number of k-element subset will decrease to $(\lambda/\lambda_1)m_1$.

$$m=\frac{\lambda}{\lambda_1}m_1=\lambda\frac{C_v^k}{C_{v-2}^{k-2}}=\lambda\frac{v(v-1)}{k(k-1)} \tag{3}$$

Definition 2 The incidence matrix $\boldsymbol{H}$ of (v,k,λ)-BIBD is a (0,1) matrix composed of m rows and v columns.

$$\boldsymbol{H}=[h_{ij}];\quad 1\leqslant i\leqslant m, 1\leqslant j\leqslant v \tag{4}$$

where

$$h_{ij}=\begin{cases}1; & s_j\in\boldsymbol{D}_i\\ 0; & s_j\notin\boldsymbol{D}_i\end{cases}$$

Then the number of subsets (16,4,1)-BIBD supported is:

$$m=1\times\frac{16\times(16-1)}{4\times(4-1)}=20 \tag{5}$$

The subsets are $\boldsymbol{D}_1=\{s_1,s_2,s_3,s_4\}$, $\boldsymbol{D}_2=\{s_5,s_6,s_7,s_8\}$, $\boldsymbol{D}_3=\{s_9,s_{10},s_{11},s_{12}\}$, $\boldsymbol{D}_4=\{s_{13},s_{14},s_{15},s_{16}\}$, $\boldsymbol{D}_5=\{s_1,s_5,s_9,s_{13}\}$,..., $\boldsymbol{D}_{20}=\{s_3,s_8,s_9,s_{14}\}$, respectively. Accordingly, the incidence matrix is:

$$\boldsymbol{H}_{20\times16}=$$

$$\begin{array}{c}\left[\begin{array}{cccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0\\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0\\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1\\
1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1\\
0&1&0&0&1&0&0&0&0&0&0&1&0&0&1&0\\
0&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0\\
1&0&0&0&0&0&1&0&0&0&0&1&0&1&0&0\\
0&0&1&0&1&0&0&0&0&1&0&0&0&0&0&1\\
0&0&0&1&0&1&0&0&1&0&0&0&0&0&1&0\\
1&0&0&0&0&0&0&1&0&1&0&0&0&0&1&0\\
0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0\\
0&0&0&1&1&0&0&0&0&0&1&0&0&1&0&0\\
0&1&0&0&0&0&1&0&1&0&0&0&0&0&0&1\\
0&0&1&0&0&1&0&0&0&0&0&1&1&0&0&0\\
0&0&1&0&0&0&0&1&1&0&0&0&0&1&0&0
\end{array}\right]\end{array}
\begin{array}{l}B_1\\B_2\\ \\ \\ \\ \\ \\ \\ \\ \\ \vdots\\ \\ \\ \\ \\ \\ \\ \\ \\B_{20}\end{array}$$

We make each row of the matrix represent one 16-bit symbol $B_i(i=1,2,...,20)$ as prefix symbol of the reader query signal. In this way, we can get 20 different prefix symbols.

3.2 Design of multi-branch query tree protocol

Multi-branch query tree protocol is composed of several circulations, with each circulation being composed of reader query and tag response. In each circulation, the reader sends out query signals to ask the tags whether their IDs contain a certain prefix symbol $B_i(i=1,2,...,20)$. If more than one tag responds, then the reader knows that there are at least two tags having the same prefix. The reader then appends symbol $B_j(j=1,2,...,20)$ to the prefix, and continues to query for longer prefix. When a tag matches a prefix uniquely, this tag might be identified. Therefore, by extending and changing the prefixes repeatedly until only one tag matches, the reader can identify all the tags.

In the traditional query tree protocol, a reader detects collision bit by bit, and all tags that match the prefix transmit their remaining bits. However, in the multi-branch query tree protocol, the reader can detect collision with a 16 bit vector symbol, and all tags that match the prefix will transmit the next 16 bit symbol of the tag ID. Figure 1 describes the flow of our protocol.

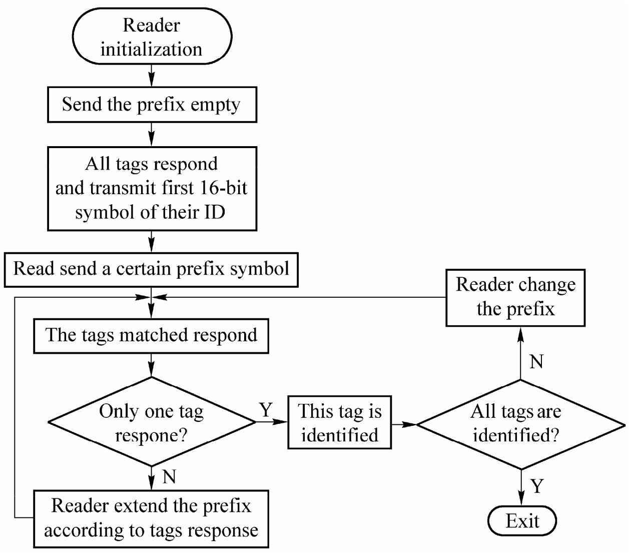

Fig. 1 The flow chart of multi-branch query tree protocol

4 A simple instance

Suppose there are four tags in the reader's operation range, and the ID of each tag is 64 bits composed of four (16,4,1) -BIBD codes. Table 1 shows the ID form of the tags.

Table 1 The ID form of tags

Tag	Tag 1	Tag 2	Tag 3	Tag 4
ID	$B_4B_{18}B_{13}B_5$	$B_4B_{18}B_{13}B_{10}$	$B_7B_{16}B_{12}B_3$	$B_9B_8B_{13}B_6$

We recognize these four tags using the multi-branch query tree protocol. First, the reader sends an empty prefix, and all four tags respond with their first symbols, which are B_4, B_4, B_7 and B_9.

The reader then uses these symbols as prefix step by step until all the tags are identified. Table 2 describes the whole identification procedure.

Table 2 The tag identification procedure using the multi-branch query tree protocol

Query depth	Reader prefix	Tag response				Explanation
		*T*1	*T*2	*T*3	*T*4	Ti represents tag *i*
1–1	empty	B_4	B_4	B_7	B_9	All tags respond with their first symbols
1–2	B_4	B_{18}	B_{18}	—	—	Tag 1 and tag 2 have the same prefix. — means No response.
1–3	B_4B_{18}	B_{13}	B_{13}	—	—	
1–4	$B_4B_{18}B_{13}$	B_5	B_{10}	—	—	Tag 1 and tag 2 are identified and turn into dormancy
2–1	B_7	—	—	B_{16}	—	Tag 3 is identified
2–2	B_7B_{16}	—	—	B_{11}	—	
2–3	$B_7B_{16}B_{12}$	—	—	B_3	—	
3–1	B_9	—	—	—	B_8	Tag 4 is identified
3–2	B_9B_8	—	—	—	B_{13}	
3–3	$B_9B_8B_{13}$	—	—	—	B_6	

As described above, the multi-branch query tree protocol needs only 4 iterations to identify one tag in the worst case while the traditional protocols might need more than 10 iterations.

5 Simulation result and analysis

Experiments are carried out upon different anti-collision algorithms to show the relationship between the identification delay and the number of tags. As indicated in Fig. 2, the identification time of the conventional FSA algorithm using the fixed frame size with 128 slots increases rapidly when the number of tags varies from 1 to 600. Especially, the performance of the FSA algorithm will be very poor when the number of tags becomes so large that the frame size is considerably lesser than it. The problem can be solved by increasing the fixed frame size but with worse performance when the number of tags is small. The frame size of the DFSA algorithm varies synchronously with the number of tags, which makes the performance of the DFSA algorithm outstanding, regardless of the number of tags. Compared with the anti-collision algorithms mentioned above, the identification speed of the traditional QT algorithm is slow. However, for the MBQT algorithm, the identification time will be longer than that of the DFSA algorithm only under the condition when the number of tags is very small. Generally, the performance of the MBQT algorithm is considerably better.

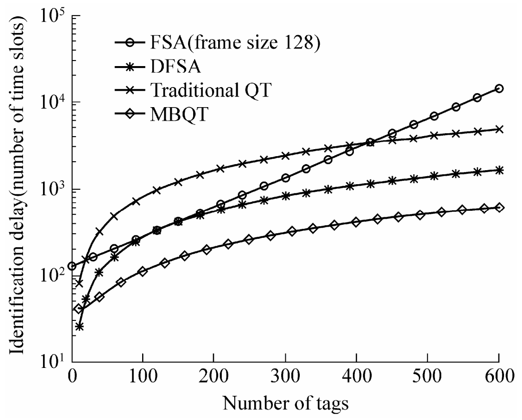

Fig. 2 Performance comparison of several anti-collision algorithms

6 Discussions

1) To increase the identification speed, the tag matching with the reader query prefix uniquely will not only respond to the next symbol but transmit the remaining symbols. For example, in the instance in Sect. 4, tag 3 is the only one that matches with the prefix B_7, which can respond to the remaining symbols $B_{16}B_{12}B_3$ at one time. Therefore, the reader can identify tag 3 at one time instead of three times.

2) To make sure that no tag fails to be identified, regressive or backtracking scheme with query prefix step-shortened after one query round can be adopted. As is seen in Table 2, the additional query depth 1–5, 1–6, and 1–7 with query prefix B_4B_{18}, B_4, and an empty one will be inserted respectively after the query depth 1–4.

3) To further increase the number of supported tags, the hybrid code scheme will be used on tag IDs. The serial number of the ID can be separated into several parts and different parts use different BIBD schemes to be compatible with EPC.

4) Compared with the traditional protocols, the MBQT protocol has the strong advantage of higher identification speed, low power consumptions, and robustness. However, the RFID reader requires higher performance for the MBQT protocol takes up more resource.

7 Conclusions

In this article, an MBQT protocol based on the BIBD code is proposed for managing RFID tag collision cases. Compared with conventional anti-collision algorithms, the protocol proposed can identify tags faster and more effectively, which is proved by theoretic analysis and simulation. The MBQT protocol will contribute to improving the performance of the RFID system.

References

1. Finkenzeller K. RFID handbook. Second edition. New York, NY, USA: John Wiley and Sons, 2003: 200–219
2. Zhen B, Kobayashi M, Shimizu M. Framed ALOHA for multiple RFID objects identification. IEICE Transactions on Communications, 2005, E88-B(3): 991–999
3. Vogt H. Efficient object identification with passive RFID tags. Proceedings of International Conference on Pervasive Computing, Aug 26–28, Zurich, Switzerland. Berlin, Germany: Springer-Verlag, 2002: 98–113
4. Vogt H. Multiple object identification with passive RFID tags. Man and cybernetics. Proceedings of IEEE International Conference on Systems, man and cybernetics. Oct 6–9, 2002, Yasmine Hammamet, Tunisia. Piscataway, NJ, USA: IEEE, 2002: 651–656
5. Myung J, Lee W, Srivastava J. Adaptive binary splitting for efficient RFID tag anti-collision. IEEE Communication Letters, 2006, 10(3): 144–146
6. Cha J R, Kim J H. Novel anti-collision algorithms for fast object identification in RFID system. Proceedings of the 11th International Conference on Parallel and Distributed Systems Workshops (ICPADS'05): Vol 2, Jul 20–22, 2005, Fukuoka, Japan. Piscataway, NJ, USA: IEEE Computer Society, 2005: 63–67
7. Wang T P. Enhanced binary search with cut-through operation for anti-collision in RFID systems. IEEE Communication Letters, 2006, 10(4): 236–238
8. Lee S R, Joo S D, Lee C W. An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification. Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05), Jul 17–21, San Diego, CA, USA. Piscataway, NJ, USA: IEEE Computer Society, IEEE, 2005: 166–172
9. Myung J, Lee W, Srivastava J, et al. Tag-splitting: adaptive collision arbitration protocols for RFID tag identification. IEEE transactions on parallel and distributed systems, 2007, 18(5): 1–13
10. Law C, Lee K, Siu K Y. Efficient memoryless protocol for tag identification. Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Aug 11, 2000, Boston, MA, USA. New York, NY, USA: ACM, 2000: 75–84