Analysis and Simulation of RFID Anti-collision Algorithms

Tao Cheng, Li Jin
School of Electronics and Information Engineering,
Beijing Jiaotong University, Beijing 100044, P.R.China
E-mail: chtao@bjtu.edu.cn

***Abstract* — In the RFID systems, tag collision arbitration is a significant issue for fast tag identification. This paper presents the analysis and simulation of several RFID anti-collision algorithms. In the anti-collision algorithms, ALOHA and Binary Tree algorithms are most widely used. The simulation is achieved by using C#.NET. In the simulation, EPC Class1 and EPC Class1 Genaration2 protocols are chosen to be the** criterions of the two algorithms**. Finally some results of the simulation are listed. According to the simulation results between the different algorithms, an advanced DFSA algorithm based on a grouping method is proposed.**

***Keywords* — RFID, anti-collision, ALOHA, Binary Tree, EPC Class 1, EPC Class1 Generation 2**

1. Introduction

Radio Frequency Identification (RFID) systems use radio frequency to automatically identify products. There are two parts in the RFID system, Reader and Tag. The RFID system can be classified into the active RFID (Tag with battery) system and the passive RFID (Tag without battery) system. Tag collision problem is one of the most important issues in the passive RFID system.

The different transmission frequencies are classified into the four basic ranges, LF (low frequency, e.g. 125KHz), HF (high frequency, e.g. 13.56MHz), UHF (ultra high frequency, e.g. 868MHz or 915MHz) and microwave (e.g. 2.4GHz). For LF and HF RFID, the read range is usually less than 60cm. For microwave RFID, because of the sensitivity to the environment, the maxim reader range is about 1m. For UHF RFID, the read range can generally reach to 5m. So the application of UHF RFID is more and more popular. In this paper, only the passive UHF RFID systems will be discussed.

Because the distance between reader and tag increases, the reader should identify many tags at one time. When there is more than one tag within the interrogation area of a reader, all the tags may send data at the same time which may lead to mutual interference. This event causes data loss and is referred to as a collision. A number of different methods are in use and in development today for preventing collisions, but all are related to making sure that only one tag "talks" at any one time. These methods are referred to anti-collision algorithms.

There are many protocols about UHF RFID, such as EPC Class1, EPC Class1 Generation 2(abbreviate as Gen2) and ISO 18000-6. In this paper, several familiar anti-collision algorithms and the algorithms used in different protocols will be discussed in details.

This work is supported by the Education Reform Foundation of Graduate School of Beijing Jiaotong University under Grant 47026.

2. Anti-collision Algorithms

For anti-collision algorithms in RFID systems, ALOHA and Binary Tree algorithms are most widely used. Both of them are based on TDMA method. The reader command is divided into several slots. When there is only one tag responses in one slot, reader can identify the tag correctly.

For ALOHA algorithm, there are many advanced versions, for example, Slotted ALOHA, Framed Slotted ALOHA and Dynamic Framed Slotted ALOHA.

2.1. ALOHA

ALOHA algorithm is a simple anti-collision method based on TDMA. When the tag reaches the interrogation area of a reader, the tag will transmit the data immediately, and when more than one tag response at the same time, the collision occurs. So the most disadvantage of this algorithm is the high probability of collision.

2.2. Slotted ALOHA

In Slotted ALOHA algorithm, the time is divided into several slots, and the tag must transmit data in one slot which it selects. So this method will decrease the probability of collision than ALOHA algorithm, but the reader and tag must communicate synchronously. When there is only one tag in one slot, reader can interrogate with tag and require the information of tag correctly. Due to the limitation of the number of slots, this algorithm used in the case that there are a few tags in the area.

2.3. Framed Slotted ALOHA (FSA)

In FSA algorithm, one frame consists of several slots, and the tag will choose one slot in a frame to transmit data. FSA algorithm uses a fixed frame size and does not change the size during the process of tag identification. In FSA algorithm, the frame size is decided by the reader. Tags generate a random number that is used to select a slot in one frame and each tag then response in the slot it selected. Reader will identify tags with multiple frames, so it can solve the problem in slotted ALOHA algorithm.

Since the frame size of FSA algorithm is fixed, its implementation is simple, but it has a weakness that drops

efficiency of tag identification. For example, in FSA algorithm, when the number of tags is small, it should choose a small number of slots in one frame, or it will cause waste of empty slots; when the number of tags is large, it should choose a large number of slots in one frame, or there will be too many collisions and it will take a long time to identify all tags.

2.4. Dynamic Framed Slotted ALOHA (DFSA)

DFSA algorithm changes the frame size for tag identification. So DFSA algorithm is more efficient than FSA algorithm to identify tags. Because of different methods to modify the frame size, DFSA algorithm has several versions. To determine the frame size, it uses the information such as the number of slots used to identify the tag and the number of the slots collided and so on. For example, when the number of the slots collided is larger than the upper limit, reader will add the number of slots in one frame, when the number of the empty slots is smaller than the lower limit, reader will decrease the number of slots in one frame.

DFSA algorithm has more advantages than other ALOHA-type algorithms, so it is most widely used.

2.5 Binary Tree

The Binary Tree algorithm is based on a tree model shown in figure 1. First divide the tags into two subsets, s0 and s1. Query s0 first, if no confliction occurs, then the tags are correctly identified; else if there are still conflictions, further divide s0 into subsets s00 and s01 recursively until all the tags in s0 are correctly recognized, then repeat this query procedure on s1.

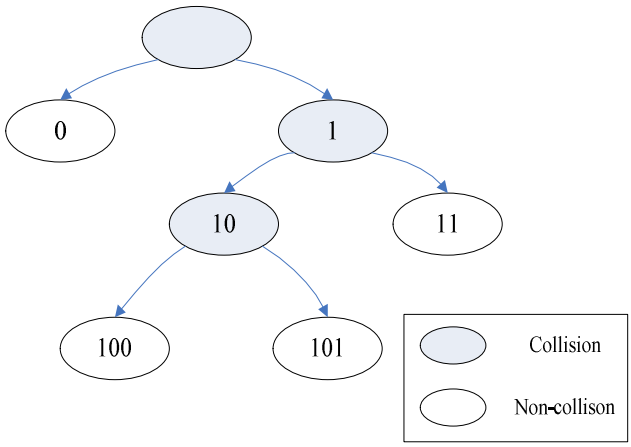

Figure 1 – Binary Tree algorithm

The Binary Tree algorithm consists of many rounds. In each round, the reader transmits a query command and the tags respond with their IDs. The query command has a prefix. Only tags of which IDs match the prefix respond. When only one tag answers, the reader successfully recognizes the tag. When more than one tags answer, collision occurs and the reader can't recognize the tags. The reader, however, can know the existence of tags which IDs match the prefix. Then the reader tries to query with 1-bit longer prefixes in next rounds. By extending the prefixes, the reader can recognize all the tags.

3. Anti-collision Algorithms in RFID protocols

For the UHF RFID protocols, EPC class 1 protocol[1] uses a bin-based Binary Tree algorithm; EPC class 1 Gen 2 protocol[2] uses a "Q"-protocol which is based on DFSA algorithm; ISO 18000-6 type A protocol uses DFSA algorithm; ISO 18000-6 type B protocol uses Binary Tree algorithm. For the simulation of the anti-collision algorithms, Class1 protocol is chosen as the criterion of Binary Tree algorithm and Gen2 protocol is chosen as the criterion of DFSA algorithm. The procedures of anti-collision algorithms in the two protocols are specified as follows.

3.1 Algorithm in Class1 protocol

Class 1 protocol uses Binary Tree algorithm. First, reader transmits a query command with a prefix, and tags match the prefix respond. When more than one tags answer, reader transmit a query command with 3 bits longer. This is the difference between Class1 and normal Binary Tree algorithm. In this case, each node in the Tree model will be divided into 8 subsets.

In Class1 protocol, the query command is defined as follows: Tags matching the prefix (which is [LEN] bits long) beginning at location [PTR] reply by sending 8 bits of the tag ID beginning with the bit at location [PTR] + [LEN]. The parameters of [PTR], [LEN] and the prefix are all defined by the reader. Figure 2 describes the query command.

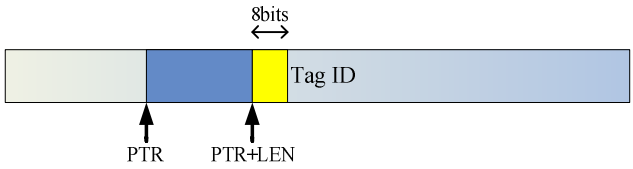

Figure 2 – Class1 query command

The 8-bit reply is communicated during one of eight slots delineated by pulses from the reader. The query command reply period is defined in the protocol and is illustrated in figure 3. The command consists of 8 slots which denote as Bin0 to Bin 7, and each slot has a 3-bit slot number. The communication slot is chosen to be equal to the value of the first 3-bit(MSB first) of the eight bit reply.

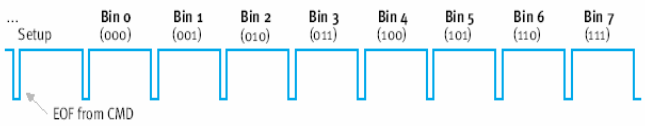

Figure 3 – Class1 query command reply period

If there is no collision in one slot, reader can identify tag correctly, if there is collision in one slot, reader will repeat the query command with 3-bit longer, and the 3 bits equal to the slot number. By extending the prefixes, reader will identify all the tags.

3.2 Algorithm in Gen2 protocol

Gen2 protocol uses DFSA algorithm. The protocol defines that all tags shall have a random number generator (RNG) and a 15-bit slot counter. Tags shall generate 16-bit random numbers (RN16) using the RNG, and shall have the ability to extract Q-bit subsets from an RN16 to preload the tag's slot counter. The Q parameter is an integer in the range (0,15) and is specified in the query command, and reader may send other commands to modify the Q parameter. Upon receiving a decrease command a Tag shall decrease its slot counter. The slot counter shall be capable of continuous counting, meaning that, after the slot counter decreases to 0000h it shall roll over and begin counting down from 7FFFh.

Upon receiving a Query command, tags shall pick a Q-bit random number in the range (0, 2^Q –1), inclusive, and shall load this value into their slot counter. Tags that pick a zero shall reply immediately. Tags that pick a nonzero value shall wait for other commands. There are three possible outcomes:

a) No Tags reply: The reader may issue another query command, or it may send the decrease command.

b) One Tag replies: The reader can identify the tag correctly.

c) Multiple Tags reply: The reader observes the collision and will issue a decrease command. All the tags will decrease their slot counter by 1.

In Gen 2 protocol, according to the collision, the reader can send a adjust command to modify the Q parameter. If tags receive the adjust command, tags will change the value of Q parameter and get a new random number which is Q-bit long. The protocol presents a method to estimate the slot number as follows:

An algorithm that a reader may use for setting the slot-count parameter Q in a query command is specified in figure 4. Qfp is a floating-point representation of Q; the reader rounds Qfp to an integer value and substitutes this integer value for Q in the identification process. Typical values for C are 0.1 < C < 0.5. The reader typically uses small values of C when Q is large, and larger values of C when Q is small.

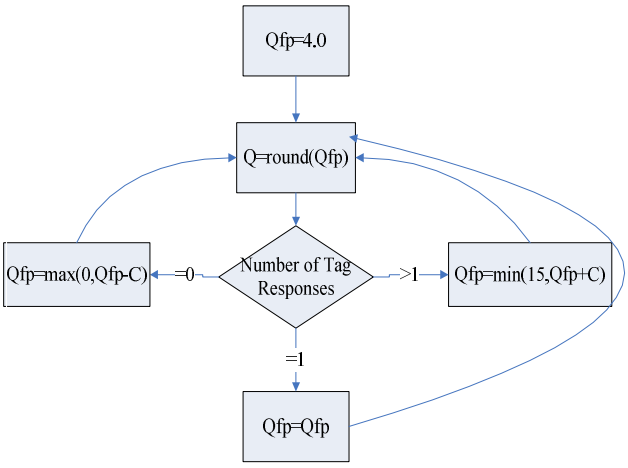

Figure 4 – Algorithm for choosing the slot-count parameter Q

According to the Q select algorithm, reader can change the number of slots in one frame automatically.

4. Simulation

To compare the anti-collision algorithms, the simulation is achieved by using C#.NET. The simulation can process the anti-collision procedure between reader and tags, and use different algorithms such as FSA, DFSA and Binary Tree. In the simulation, different parameters of algorithm can be selected, such as the number of tags to be read and the number of slots in one frame.

4.1 Simulation of FSA

The results in the simulation in FSA are listed in Table 1(The result shown is the average of ten times simulation). It shows the number of slots used when reader sends to interrogation tags by using different numbers of slots in one frame and different numbers of tags.

Table 1: Simulation results of FSA algorithm

Number of tags	Number of slots in one frame				
	16	64	256	512	1024
20	69	120	256	512	1024
50	345	157	382	512	1024
100	2870	462	382	768	1365
300	108515	4221	1530	1330	1706
500	539468	25106	1938	1788	2784

4.2 Simulation of DFSA

The simulation of DFSA algorithm is based on Gen2 protocol. Several results in the simulation of DFSA are listed in Table 2(The result shown is the average of ten times simulation). It shows the number of slots used when reader sends to interrogation tags by using different numbers of tags and different initial values of Q parameter. The init Q parameter represents that the slot numbers per frame is 2^Q.

Table 2: Simulation results of DFSA algorithm

Number of tags	Init value of Q parameter					
	4	6	8	9	10	12
20	62	68	69	69	71	76
50	184	157	160	161	167	172
100	385	381	356	363	384	385
300	1183	1176	1181	1146	1169	1167
500	1855	1851	1853	1852	1857	1865

The simulation results of FSA and DFSA algorithms are illustrated in figure 5.

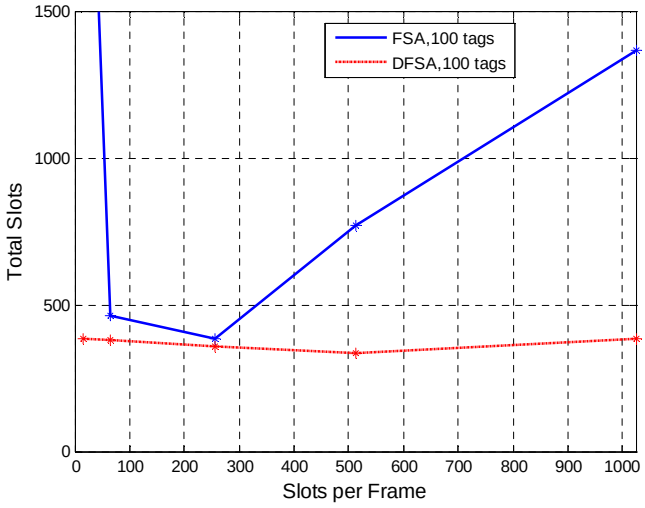

Figure5 –Comparation of FSA and DFSA algorithms

In the simulation, the total number of tags is designed to be 100. In the figure, the solid line represents the FSA algorithm, and the dotted line represents the DFSA algorithm which is used in Gen2 protocol.

According to the simulation results of FSA algorithm above, when the number of slots per frame is smaller than the number of tags, there is so many collision that it costs so many slots to identify tags; when the number of slots per frame is bigger than the number of slots, and along with the increase of the number of slots per frame, it costs more and more slots totally because of the waste in empty slots. So when the slot number is close to the tags number, reader can identify tags more efficiently.

According to the simulation result of DFSA algorithm above, the total numbers of slots change smoothly no matter what init Q parameter selects. So the DFSA algorithm can solve the FSA problem of inefficiency.

4.3 Simulation in Binary Tree

The simulation of Binary Tree algorithm is based on Class1 protocol. Several results in the simulation of Binary Tree algorithm are listed in Table 3(The results shown are the average of ten times simulation). It shows the numbers of slots used when reader sends to interrogation tags by using different tag numbers.

Table 3: Simulation results of Binary Tree algorithm

Number of tags	50	100	300	500	1000	1500	2000
Total slots	181	373	984	1763	3881	5485	7716

To compare the DFSA algorithm and the Binary Tree algorithm, the simulation results of the two algorithms are listed in Table 4(The results shown are the average of ten times simulation). It shows the numbers of slots used when reader identify tags by using different tag numbers.

Table 4: Simulation results of Binary Tree and DFSA algorithm

Total Slots	Number of tags						
	20	50	70	100	200	300	500
Binary Tree	80	181	256	373	684	984	1763
DFSA	62	157	225	356	694	1146	1855

To compare the Binary Tree and DFSA algorithms, the simulation results of the two algorithms are illustrated in figure 6.

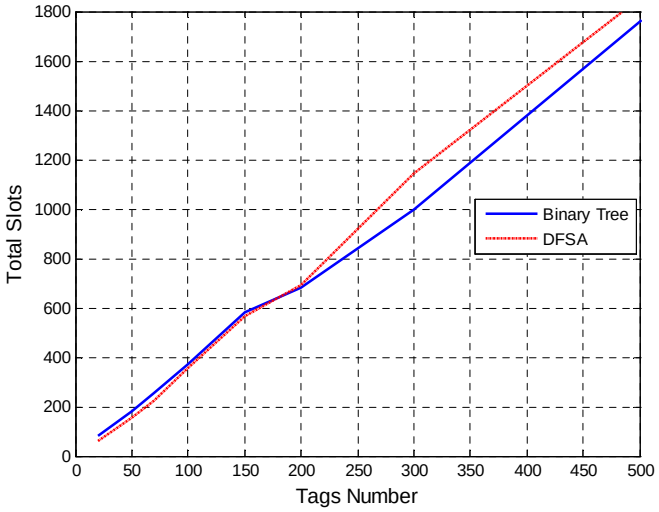

Figure6 –Comparation of Binary Tree and DFSA algorithms

In figure 6, the blue solid line represents the Binary Tree algorithm; the red dotted line represents the DFSA algorithm. According to the simulation results, when the number of tags is small, the DFSA algorithm is better than the Binary Tree algorithm; when the number of tags is large, the Binary Tree algorithm is more efficient than the DFSA algorithm. The ALOHA algorithm is easier to implement than the Binary Tree algorithm, so an advanced DFSA algorithm is proposed.

4.4 Advanced DFSA algorithm

The DFSA algorithm is less efficient when the number of tags is large, then split the tags into several groups the number of tags in each group will be smaller, and then reader can identify tags use DFSA algorithm in each group.

The procedure of grouping is as follows:

First tags generate an N-bit random number which represents the group number, so the total number of groups is 2^N. Tags which choose the same number list in the same group.

In the DFSA algorithm in Gen2 protocol, each tag has a RNG and a slot counter. Tags generate a 16-bit random number(RN16) using the RNG, and extract Q-bit subsets from

an RN16 to preload the tag's slot counter. So it is easy to add a slot counter to extract N-bit subsets from an RN16 as the group number.

Then reader will identify tags using DFSA algorithm group by group.

Several results in the simulation of the advanced DFSA algorithm are listed in Table 5(The results shown are the average of ten times simulation). It shows the number of slots used to identify all the tags by using different numbers of groups and tags.

Table 5: Simulation results of advanced DFSA algorithm

Number of groups	Number of tags						
	50	100	200	300	500	1000	2000
2	164	302	662	1152			
4	167	299	646	1029	1792	3626	
8		318	643	993	1710	3501	
16			657	964	1682	3397	7325
32				1015	1792	3401	6970
64							6527

According to the simulation results, it takes less number of slots to identify all tags when the number of tags in each group is between 20 and 50. But if the number of tags in each group is too small, it will waste some of the slots and drop the efficiency of the algorithm. So it is important to choose a right grouping number. The simulation results of the advanced algorithm using the right grouping number are illustrated in figure 7(The results shown are the average of ten times simulation).

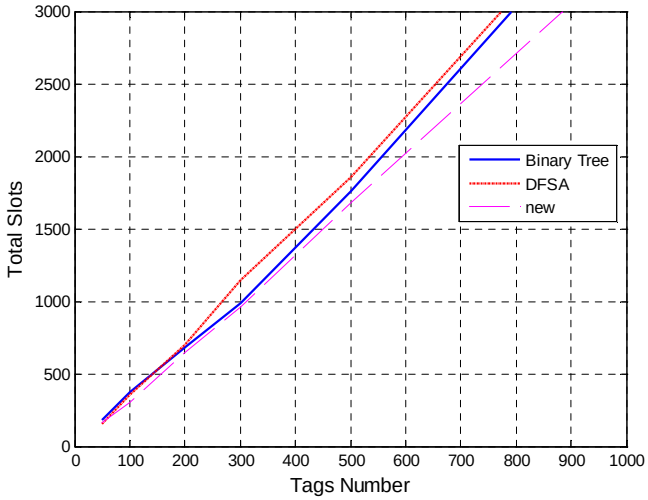

Figure7 –Comparation of Binary Tree, DFSA and advanced DFSA algorithms

In the figure, the blue solid line represents Binary Tree algorithm in Class1 protocol; the red dotted line represents the DFSA algorithm in Gen2 protocol; the pink dash-dotted line represents the advanced DFSA algorithm.

According to the results, the advanced DFSA algorithm is better than the other two algorithms.

5. Conclusion

In this paper, several anti-collision algorithms and the simulation results of the algorithms are listed. In the simulation, different parameters of the algorithms are used to examine the performance of the algorithm and several results are presented. Finally an advanced DFSA algorithm is proposed. According to the simulation results, the advanced DFSA algorithm is better than DFSA and Binary Tree algorithms when use the right grouping number.

REFERENCES

[1] EPC Global. 860MHz~930MHz Class I Radio Frequency Identification Tag Radio Frequency & Logical Communication Interface Specification Candidate Recommendation, Version 1.0.1

[2] EPC Global. EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz~960 MHz Version 1.0.9

[3] Jae-Ryong Cha, Jae-Hyun Kim; "Dynamic Framed Slotted ALOHA Algorithms using Fast Tag Estimation Method for RFID System". Consumer Communications and Networking Conference, 2006. CCNC 2006. 2006 3rd IEEEVolume 2, 8-10 Jan. 2006 Page(s):768 - 772 Summary: Not available.....

[4] Floerkemeier, C.; Wille, M.; "Comparison of Transmission Schemes for Framed ALOHA based RFID Protocols". Applications and the Internet Workshops, 2006. SAINT Workshops 2006. International Symposium on23-27 Jan. 2006 Page(s):4 pp.

[5] Su-Ryun Lee; Sung-Don Joo; Chae-Woo Lee; "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification". Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on 17-21 July 2005 Page(s):166 – 172

[6] Leian Liu; Zhenhua Xie; Jingtian Xi; Shengli Lai. <<An improved anti-collision algorithm in RFID system>>. Mobile Technology, Applications and Systems, 2005 2nd International Conference on 15-17 Nov. 2005 Page(s):5 pp.