ACSIJ Advances in Computer Science: an International Journal, Vol. 2, Issue 4, No.5 , September 2013
ISSN : 2322-5157
www.ACSIJ.org

Reverse Query Tree approach to cope with Id distribution problem in Tree-based tag anti-collision protocols of RFID

Milad HajMirzaei

Department of Computer Engineering , Science And Research Branch
Islamic Azad University, Yazd, IRAN
Hmirzaei.m@gmail.com

Abstract

Tag collision is one of the most important issues in RFID systems and many tag anti-collision protocols were proposed in literature. But some kind of these protocols like Tree-based protocols (specifically Query tree) which its performance depends on tag id length and construction, have some issues like id distribution. In this paper we discuss about Query tree protocol which may influenced by id distribution. Then we propose a novel type of this protocol called Reverse Query tree to solve it.

Keywords: *RFID, Tag collision*, *Tree-based protocols.*

1. Introduction

Since the advent of RFID (Radio Frequency Identification) in 1948 [1] this technology was first used in WWII by the Allies armed forces to distinguish friendly from enemy aircraft and tanks, called IFF (Identify Friend or Foe) [2]. Today RFID has a significant role in fields of : supply chain management, agriculture, military, healthcare, Pharmaceuticals, retail and so on.

This technology use communicated radio frequency to retrieve data. The main RFID components are :Reader including an antenna which is the device used to read or write data to RFID tags. Tag is a device with an integrated circuit on which the reader acts. The tags can earn its energy from signals which has received from reader, which called passive tag or by its own battery supply (active tag). There is another tag which called semipassive tag that uses battery supply to power on and received signal energy from reader to transmit data. This technology is used to track inventory, object identification and more. Data write on the tags and attach to objects to rapid and automatically read data.

A typical RFID system contain of some tags and one or some readers and a computer to interrogate information. As soon as computer`s request, reader send query to receive tags information which are in its interrogation zone. Then transfer the results to computer for processing. One of the RFID issues is tag collision. When more than one tag transmits its ID to reader simultaneously, the collision occurs. Almost the received signals are corrupted. RFID suffers from incorrect received signals due to collision. It's reported that in typical RFID deployments, the tag read rate is usually about 60-70% [3]. To address this issue some algorithms have been proposed, called Tag anti-collision. These algorithms allow all of tags in the interrogation zone of the reader to be identified successfully. We can classify them to 3 groups. Aloha-based algorithms, Tree-based algorithms and Counter-based algorithms. In this paper we will discuss about Tree-based algorithms, specifically Query Tree (QT) scheme. The QT performance depends of ID distribution and ID length. In section 4 we will explain this issue, then propose a novel approach to address the this.

The rest of this paper is organized as follows: section 2 overviews tag anti-collision algorithms. Section 3 we discuss about tag memory content then we will explain about ID distribution issue in section 4. In section 5 we will propose an approach to solve this problem and then we simulate them in section 6. Section 7 is conclusion.

2. Tag Anti-Collision Algorithms

To identify tags within the interrogation zone, a reader sends a request to ask tags to send back their IDs. When multiple tags within the reader’s interrogation zone respond to the request simultaneously, collision occurs and the reader cannot identify any tag properly. This is called the tag collision problem. Therefore, tag collisions are resolved by the reader utilizing techniques collectively known as anti-collisions schemes. Several tag anti-collision protocols are proposed for reducing tag collisions. They can be categorized into three classes: ALOHA-based, tree-based, and counter-based protocols.

2.1 Aloha- Based protocols

ALOHA-based tag anti-collision protocols [4-7] are based on a backoff mechanism that operates in a probabilistic manner. They try to stagger the response times of tags in the interrogation zone. In general, ALOHA-based protocols are simple and have fair performance. However, they have the tag starvation problem that a tag may never be identified because its responses always collide with others.

2.2 Counter-Based protocols

Counter-based protocols [8,9] do not have the tag starvation problem. The basic idea of the two classes of protocols is to repeatedly split the tags encountering collisions into subgroups until there is only one tag in a subgroup to be identified successfully.

2.3 Tree-Based protocols

The basic idea of the tree-based tag anti-collision protocol is to repeatedly split the tags encountering collisions into subgroups according to tag IDs until there is only one tag in a subgroup to be identified successfully. In general, the tree-based protocol has longer identification time latency than that of the ALOHA-based protocol, but it does not have the tag starvation problem. A further drawback of the tree-based protocol is that its performance is affected by the length or the distribution of tag IDs. Below we will introduce main Tree-based protocol called Query tree (QT) protocol.

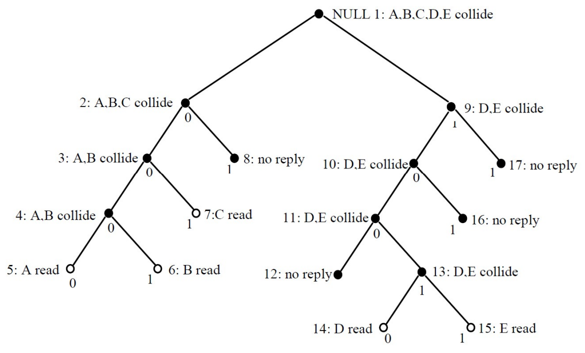

Fig. 1 example of QT protocol for five tags. tags are read in white circles. A=0000001101, B=0001000011, C=0010001010, D=1001001000, E=1001100001

2.3.1 Query Tree (QT)

In this approach [10] a reader first broadcast a request bit string S to tags. The tags which their prefix IDs match with S, respond to the reader by sending back the remaining bits of their IDs. comparison start from most significant bits of ID. If only one tag responds to reader, the tag is identified correctly otherwise more than one tags respond simultaneously to reader, the collision occur. In this case the reader sends a longer bit string that has one bit more than the last string. Usually reader appends 0 or 1 to S string that is S0 or S1 (almost first use 0). Tags divide to two subgroups: tags start with S0 and S1. It repeats until only one tag match with S string to identify correctly. This approach delay depends on the ID length and ID distribution. For instance we have five tags A, B, C, D and E. we can observe result of this scheme in figure 1 and in Table 1 with more details. In this example each tag ID length is 10 bits and tag IDs was chosen randomly. With the assumption of random IDs we did not consider ID distribution in this example.

3. Tag IDentification (TID)

in practice the tag IDs bits are not randomly chosen like what we did in QT example. In this section we use [11] to show and briefly describe TID structure.
The contents of the TID memory bank of a Gen 2 Tag, as a bit string $b_0b_1\dots b_{N-1}$, where the number of bits N is at least 48. the Tag Identification memory bank Shall contain an 8 bit ISO/IEC 15963 allocation class identifier of E2h (11100010) at memory locations 00h to 07h . TID memory locations 08h to 13h Shall contain a 12 bit Tag mask designer identifier (MDID) obtainable from EPCglobal. TID memory locations 14h to 1Fh Shall contain a 12-bit vendor-defined Tag model number (TMN). bits $b_{32}\dots b_{34}$ as a 3-bit unsigned integer V. If V equals zero, that means this TID bank contents does not contain a serial number. Otherwise, we can calculate the length of the serial number $L = 48 + 16(V - 1)$. Consider bits $b_{48}b_{49}\dots b_{48+L-1}$ as an L-bit unsigned integer. This is the serial number. In such cases which we have two tags having the same MDID and TMN, the serial number must be different and unique.

4. ID Distribution Issue

Some tag anti-collision protocols which their performance depends on tag ID may be influenced by ID distribution such as Query tree which has described before. To quantify the similarity of IDs, we define an identical bit as the length of the identical prefix all tag IDs have. The tag ID is depicted by $x_1x_2\dots x_mx_{m+1}\dots x_{96}$ (xi is a binary digit, $(1<m<96)$ and all tag IDs have the same $x_1x_2\dots x_m$ if the identical bit is m and each tag has a 96 bit ID.

For instance we have 2 tags A=0000000101 and B=0000001011 with identical bit m=6 and each tag ID length is n=10. If we use QT protocol to identify the tags, the QT succeeded to identify them in depth of m+1=7 in QT tree. This example is shown in figure 2. In this example 7 collisions occur before identifying two tags correctly.

Table 1: QT for five tags. R:Respond, NR: No Respond, A=0000001101, B=0001000011, C=0010001010, D=1001001000, E=1001100001

No.	Query	A	B	C	D	E	Result
1	NULL	R	R	R	R	R	Collision
2	0	R	R	R	NR	NR	Collision
3	00	R	R	NR	NR	NR	Collision
4	000	R	R	NR	NR	NR	Collision
5	0000	R	NR	NR	NR	NR	Read A
6	0001	NR	R	NR	NR	NR	Read B
7	001	NR	NR	R	NR	NR	Read C
8	01	NR	NR	NR	NR	NR	No reply
9	1	NR	NR	NR	R	R	Collision
10	10	NR	NR	NR	R	R	Collision
11	100	NR	NR	NR	R	R	Collision
12	1000	NR	NR	NR	NR	NR	No reply
13	1001	NR	NR	NR	R	R	Collision
14	10010	NR	NR	NR	R	NR	Read D
15	10011	NR	NR	NR	NR	R	Read E

5. An Approach to solve ID distribution issue

We understand from the above example the performance of QT protocol directly depends on identical bit m. consider we have 100 tags that m bought from same vendor with the same type, so this means we have 100 tags with identical bit m. to identify the whole tags through QT protocol the competition among tags commence after level m+1. It is due to identical bits are first m bits and the different bits are last n-m bits (n is ID length and tags start to send its ID from most significant bits). to solve this issue we can change the QT protocol to force the tags to start sending its IDs from least significant bit (LSB) instead of most significant bit (MSB). In this way competition among tags start from the first bits and it doesn't wait until the protocol pass the m identical bit. We call this approach Reverse query Tree (RQT). We use RQT for previous example of figure 2 to show its efficiency in ID distribution situation. RQT read TID from LSB to MSB, in other words RQT invert the TID. So RQT change A=0000000101 to A=1010000000, B=0000001011 to B=1101000000, then perform like QT to identify tags. We can observe the RQT protocol performance in figure 3. In this example 2 collisions occur before RQT identify two tags correctly.

6. Evaluation

We used C# to study the performance of RQT and QT. this evaluation is based on the number of collisions occur in each protocol for different identical bit. We use 100 tags with identical bit m=0,8..., 64. TID length is 96 bits. First m bits in ID are identical and rest of bits were chosen randomly. We also assume a noise free channel and packet loss are due to collision only. Each simulation is repeated 10 times. The evaluation result is shown in figure 4.

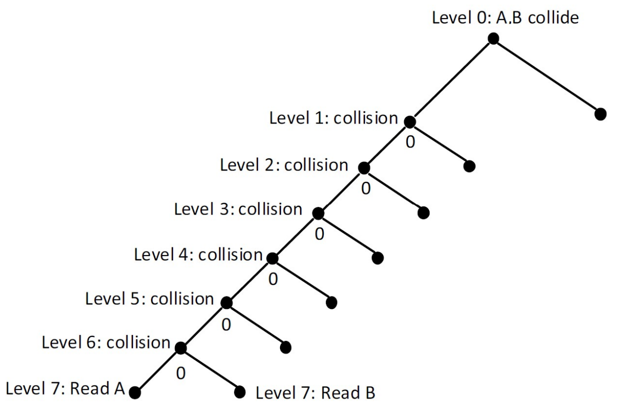

Fig. 2 Example of QT protocol for two tags. TID include 6 identical bits

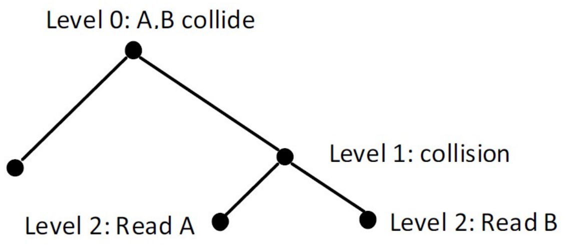

Fig. 3 Example of Reverse QT for two tags. TID include 6 identical bits

7. Conclusion

as mentioned before one of drawback of QT is ID distribution. When we have some tags with no identical bits as a prefix in TID, QT has its best throughput, like the point in figure 4 with zero identical bits. By increasing identical bits, number of collisions increase as we expected. It is due to dependency to identical bits. Also we can see RQT has a normal trend in all point of our simulation because RQT does not depend on identical bits.

Acknowledgments

The authors would like to thank Dr Pouya Derakhshan Barjoei who provide insightful and constructive comments on the paper.

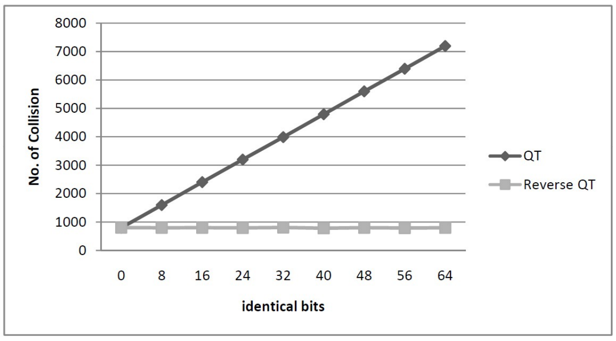

Fig. 4 Evaluating the performance of QT and RQT protocol with different identical bits

References

[1] H. Stockman , “Communication by Means of Reflected Power”, Proc. IRE 35 1948, 1196–1204
[2] G. Roussos , Networked RFID systems software and services , Springer-Verlag London Limited ,pp . 1-10 ,2008
[3] S. R. Jeffery, M. J. Franklin, & M. Gaorfalakis, (2008). “An adaptive RFID middleware for supporting metaphysical data independence” The VLDB Journal, 17(3), 265–289.
[4] D. Krebs and M. J. Liard. White Paper: Global Markets and Applications for Radio Frequency Identification. Venture Development Corporation, 2001.
[5] J. R. Cha and J. H. Kim. Novel anti-collision algorithms for fast object identification in RFID systems. In Proc. of the 11th International Conference on Parallel and Distributed systems—Workshops (ICPADS’05), pp. 63–67, Fuduoka, Japan, 2005.
[6] G. Khandelwal et al. ASAP: A MAC protocol for dense and time constrained RFID systems. In Proc. of IEEE International Conference on Communications (ICC’06), Istanbul, Turkey, 2006.
[7] S. Lee, S. D. Joo, and C. W. Lee. An enhanced dynamic framed slotted aloha algorithm for RFID tag identification. In Proc. of Mobiquitous, pp. 166–172, 2005.
[8] M.-K. Yeh and J.-R. Jiang. Adaptive k-way splitting and pre-signaling for RFID tag anti-collision. In Proc. of the 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON’07), Taipei, Taiwan, 2007.
[9] J. Myung and W. Lee. Adaptive splitting protocols for RFID tag collision arbitration. In Proc. of MobiHoc 2006, pp. 202–213, Florence, Italy, 2006.
[10] C. Law, K. Lee, and K.Y. Siu, E_cient Memoryless Protocol for Tag Identi_cation, Discrete Algorithms and Methods for MOBILE Computing and Comm 2000, 75-84.
[11] GS1 EPC Tag Data Standard 1.6. Ratified standard. 9 september 2011. http://www.gs1.org/gsmp/kc/epcglobal/tds