QT-CBP: A New RFID Tag Anti-Collision Algorithm using Collision Bit Positioning★

Hyunji Lee[1], Jongdeok Kim[2]

Dept. of Computer Science and Engineering, Pusan National University.
[1]eastleap@pusan.ac.kr, [2]kimjd@pusan.ac.kr

Abstract. The ability to recognize many tags simultaneously is crucial for many advanced RFID-based applications. The tag anti-collision algorithm of an RFID system, which arbitrates collisions on the air interface among tags in the same reading range of a reader, makes a great influence on the speed and the reliability in multiple tag recognition. This paper presents a new memoryless tag anti-collision algorithm, QT-CBP (Query Tree with Collision Bit Positioning), which is designed based on QT (Query Tree) algorithm. QT-CBP is likely to make more concise tree traversal than QT by extracting and making use of detailed information on a collision condition, such as the number of collision bits and their positions. Basically QT-CBP is an enhanced algorithm for readers running QT, so no change is required at tags. Simulation study shows that QT-CBP outperforms QT, especially on the condition where tags have similar tag IDs by having the same company or product ID prefixes.

1 Introduction

Radio Frequency IDentification (RFID) is an automatic identification technology that a reader recognizes objects through wireless communications with tags attached to the objects. RFID systems have recently begun to find greater use in industrial automation and in supply chain management. The ability to recognize many tags simultaneously is crucial for many advanced RFID-based applications in these domains. However, multiple tags in the same reading range of a reader may interfere with each other, which make the reader hard to recognize the tags. RFID tag anti-collision algorithms are developed to cope with this problem and the performance of multiple tag recognition is greatly influenced by the algorithm applied [1-8].

RFID tag anti-collision algorithms can be categorized into tree-based algorithms and aloha-based algorithms. While aloha-based algorithms can be faster in identification than tree-based ones, they have a serious problem that a tag may not be identified for a long time so called tag starvation. A tree-based algorithm can be categorized into either "memoryless" or "memoryful" whether it requires memory at

★ This work was supported by the Regional Research Centers Program (Research Center for Logistics Information Technology), granted by the Korean Ministry of Education & Human Resources Development.

tag. As memoryless tag is relatively cheap and easy to implement, memoryless algorithms are widely used in building cost effective RFID systems [8].

This paper presents a new memoryless tree-based tag anti-collision algorithm, QT-CBP (Query Tree with Collision Bit Positioning). QT-CBP is designed based on QT (Query Tree), a representative memoryless tree-based tag anti-collision algorithm developed at MIT's Auto-ID Center [1]. When a collision occurs, QT interprets it as a simple boolean condition, but QT-CBP extracts more information about the collision, such as the number of collision bits and their positions. By making use of this information, QT-CBP can resolve collisions with fewer queries and fewer collisions than QT.

The improvement gets clear if tags under interrogation have similar tag codes. Note that most of RFID code schemes, including the Electronic Product Code™ (EPC™) standard from the EPCglobal, adopt hierarchical structure [9]. Multiple tags to be identified simultaneously in the real world are likely to have similar tag codes, because they are likely to have the same prefix for its company and product IDs, which increases the efficiency of QT-CBP. Another good point of QT-CBP is that it does not require any change at tags, as it is basically an enhanced algorithm for readers running QT. The performance of QT-CBP is evaluated through simulation and the results show that QT-CBP reduces overall identification delay and the number of collisions than QT.

The rest of this paper is organized as follows. In Section 2, we briefly introduce the existing tag anti-collision algorithms by categorizing them and make a detail review on the QT algorithm. Section 3 presents our QT-CBP algorithm in detail. The performance of QT-CBP is evaluated through simulation and analysis and addressed in Section 4. The conclusion of the paper is drawn in Section 5.

2 Related Studies

2.1 Classification of Anti-Collision Algorithms

An RFID tag anti-collision algorithm can be categorized into either an aloha-based algorithm or a tree-based algorithm. Aloha-based algorithms reduce the probability of tag collision by making tags randomly select their response time. Though aloha-based algorithms may be faster than tree-based algorithms, their performances are stochastic and they cannot perfectly prevent collisions which bring about a serious problem called tag starvation that a tag may not be identified for a long time. Aloha-based algorithms can be further classified into either Bit-slot type [5] or ID-slot type [6][7]. While ID-slot type transfers whole tagID, Bit-slot type transfers only a bit during the response time.

A tree-based algorithm makes a binary tree by splitting the group of colliding tags into two subgroups and traverse the tree until the reader recognize the IDs of tags without collision. Though tree-based algorithms have relatively long identification delay, they do not cause tag starvation. A tree-based algorithm can be further classified into either "memoryless" or "memoryful" whether it requires memory at tag. In memoryless algorithms [1][2], the responses from tags are decided based only on the current query from the reader. However, in memoryful algorithms [3][4], tags are able to store their current status of tree traversal and respond to a reader's query based on not only the query but also their status stored in memory. Though, more efficient tree traversal can be accomplished by making use of memory at tag, memoryful algorithm requires more complex tags which are difficult to implement and hard to be cost effective. The classification of RFID tag anti-collision algorithms is summarized at Table 1.

Table 1. Classification of tag anti-collision algorithms

Anti-Collision Algorithm		
Tree-based Algorithm	Memoryless	Query Tree [1] Tree Walking [2]
	Memoryful	Splitting tree [3] Bit-arbitration [4]
Aloha-based Algorithm	Bit-Slot	Bit-Slot [5]
	ID-Slot	I-Code [6] STAC [7]

2.2 Query Tree Algorithm

The QT algorithm is a representative memoryless tree-based tag anti-collision algorithm developed at MIT's Auto-ID Center. The QT algorithm consists of rounds. In each round, a reader transmits a query to tags, and then tags respond it with their IDs. The reader makes its query by popping a query stored in the query queue. Each query contains k-bits long prefix string which each tag compares with the prefix of its tagID. If it matches, the tag sends its whole tagID as a response or simply ignores the query and makes no response otherwise. When the query queue is empty, the reader makes a special query including the empty prefix string <ε>. Any tag receiving the query including the empty prefix string <ε> should response.

If there is only one response for a query, the reader can successfully recognize the tag. However, if there are more than one responses, responses collide and the reader cannot recognizes the tag. In this collision case, the reader creates two queries by appending '0' (zero) and '1' (one) to the previous query and stores them into the query queue. In case of no tags matches the prefix, there is no tag response. And the reader does noting and begins the next round. The algorithm repeats the above procedure until all queries in the queue are popped (i.e., empty).

Fig. 1 illustrates the process of QT algorithm with four tags which have IDs of 000, 001, 101 and 110. Each column represents a round. The first row 'Reader' shows the queries sent from the reader, the second row 'response' shows the aggregated responses from the tags, from the third to the sixth rows show the responses of each tags, the seventh row 'Q' shows stored queries in the query queue, and the eighth row 'M' shows identified tags and their IDs.

Reader	ε	0	1	00	01	10	11	000	001
response	Collision	Collision	Collision	Collision	No Response	101	110	000	001
Tag1 (000)	000	000		000				000	
Tag1 (001)	001	001		001					001
Tag3 (101)	101		101			101			
Tag4 (110)	110		110				110		
Q={ε}	0 1	1 00 01	00 01 10 11	01 10 11 000 001	10 11 000 001	11 000 001	000 001	001	
Memory M						101	101 110	101 110 000	101 110 000 001

Fig. 1. Process illustration of QT algorithm for 4 tags with 3 bits long tag IDs, 000, 001, 101 and 110

3 QT-CBP Protocol

When a collision occurs, QT interprets it as a simple boolean condition. The QT might be designed based on an assumption that a reader can not extract any valid information but collision itself when a collision occurs. However, we assert that more detailed information about the collision can be extracted. For example, in 900MHz EPC Class 0 RFID systems, one of most widely used RFID systems, each tag response is defined by two sub-carrier frequencies, one for binary 0, and the other for binary 1. As 0 and 1 are responded through different frequencies, a reader can identify them at the same time [10]. Though a reader cannot differentiate one 0 (from a single tag) from multiple 0s (from multiple tags), it does not matter for QT-CBP.

From the above observation, we assume that a reader can extract more detailed information about the collision, such as the number of collision bits and the positions of colliding bits. Based on this assumption, we designed a new memoryless tree-based tag anti-collision algorithm, QT-CBP. Without changing tag operation, a QT-CBP reader makes use of information on collision bits and their positions to remove redundant queries which just make another useless collision. As a result, the QT-CBP can reduce overall identification delay and the number of collisions than QT. Figure 2 illustrates the pseudo codes of QT-CBP.

```
The QT-CBP Protocol
Reader has a query stack S and a TagId memory M.
Let ωk be the k'th bits of a bit string
Reader
 Begin
   Initially S = < ε >, M = <  >
   while(stack is not empty)
     Pop a query q from S;
     Broadcast q;
     Switch (response result)
       Case "only one response":
           Save the responded tagID r in M
           Break;
       Case "more than one response":
           Get the aggregated response R
           R = $\omega_1\omega_2\omega_3...\omega_{k-1}X...$; X -> collision bit
         count the collision(X) bits -> Nc
         resolve the position of the first colliding bit -> k
         If (Nc = 1)
               Get two new tagIDs r1, r2 from R
            $r_1=\omega_1\omega_2\omega_3...\omega_{k-1}0...$, $r_2=\omega_1\omega_2\omega_3...\omega_{i-1}1...$
            save r1, r2 in M
         else
           Push two new queries q1, q2 to S
            $q_1=\omega_1\omega_2\omega_3...\omega_{k-1}0$, $q_2=\omega_1\omega_2\omega_3...\omega_{i-1}1$
           Break;
       Case "no response": // Do nothing
           Break;
   end while
 end
```

```
Tag
 Has a TagID r = $\omega_1\omega_2\omega_3...\omega_{|tagID Length|}$
 begin
     Wait (query q from the reader)
     if(q=ε or q=$\omega_1\omega_2\omega_3...\omega_{|q|}$)
       send r to the reader
 end
```

Fig. 2. A pseudo code illustration of QT-CBP

Figure 3 illustrates the process of QT-CBP for the same configuration of Figure 1. Note that while QT uses a queue to store the next queries, QT-CBP uses a stack. While QT requires four queries, which are 0, 00, 000 and 001, to resolve two tags, 000, 001, QT-CBP requires just one query, 0, to resolve them. The overall number of rounds required to resolve all tags reduced from nine to five.

Reader	ε	1	11	10	0
Response	Collision (XXX)	Collision (1XX)	110	101	(00X) 000 001
Tag1 (000)	000				000
Tag2 (001)	001				001
Tag3 (101)	101	101		101	
Tag4 (110)	110	110	110		
S={ε}	0 1	0 10 11	0 10	0	
Memory M			110	110 101	110 101 000 001

Fig. 3. Process illustration of QT-CBP algorithm for 4 tags with 3 bits long tag IDs, 000, 001, 101 and 110

4 Simulation for Performance Evaluation

The Electronic Product Code™ (EPC™) is an identification scheme for universally identifying physical objects via Radio Frequency Identification (RFID) tags. EPCglobal has released the Tag Data Standards (TDS) specification – recently TDS version 1.3, which includes a General Identifier (GID) and several serialized version of the EAN.UCC (European Article Numbering - Uniform Code Council) legacy encoding schemes. Figure 4 shows general form of tag data structure as representative scheme – the GID-96 encoding scheme [9].

	Header	General Manager Number	Object Class	Serial Number
GID - 96	8	28	24	36
	00110101 (Binary value)	268,435,455 (Max. decimal Value)	16,777,215 (Max. decimal value)	68,719,467,735 (Max. decimal value)

Fig. 4. EPCGlobal's GID-96 Tag Sturcture

The GID-96 encoding scheme is used in modeling sample tag data of our simulation for performance evaluation. The GID-96 scheme consists of four fields, 'Header', 'General Manager Number', 'Object Class' and 'Serial Number'. The Header field defines the overall length and format of the following fields and it is fixed to 00110101 for the GID-96 scheme. The General Manager Number identifies an organizational entity (essentially a company, manager or other organization) that is responsible for maintaining the numbers in subsequent fields, Object Class and Serial Number. EPCglobal assigns the General Manager Number to an entity, and ensures that each General Manager Number is unique. The Object Class is used by an EPC managing entity to identify a class or "type" of thing. These object class numbers, of course, must be unique within each General Manager Number domain. Finally, the Serial Number code, or serial number, is unique within each object class. In other words, the managing entity is responsible for assigning unique, non-repeating serial numbers for every instance within each object class.

Because of the hierarchical structure of the standard encoding scheme, we assert that multiple tags to be identified simultaneously in the real world are likely to have similar tag IDs, as they are likely to have the same General Manager Number or even the same Object Class number. We applied the above assertion to our simulation. Sample tags are generated to reflect the above characteristics by sharing the same Object class number or sequentially assigned Serial Number. We carry out simulations for three different tag generation cases. In each case, we increase the number of tags to be identified from 100 to 1000.

To evaluate the performance of QT and QT-CBP, we measured the total length of queries in bits and the number of rounds required to identify all the sample tags given.

Case 1: Five Different Object Classes with Sequential Serial Codes. In this simulation, tags for five different object classes are generated. Tags belong to the same object class have sequentially assigned serial codes. The followings are examples of generated sample tags. The underlined parts are Object Class number.

000011110001 , 000011110002, ...
000011101001 , 000011101002, ...
011100001021 , 011100001022, ...

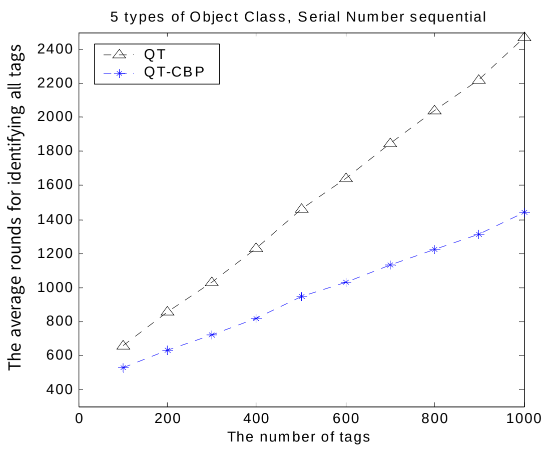

Fig. 5. The number of rounds required – Case 1

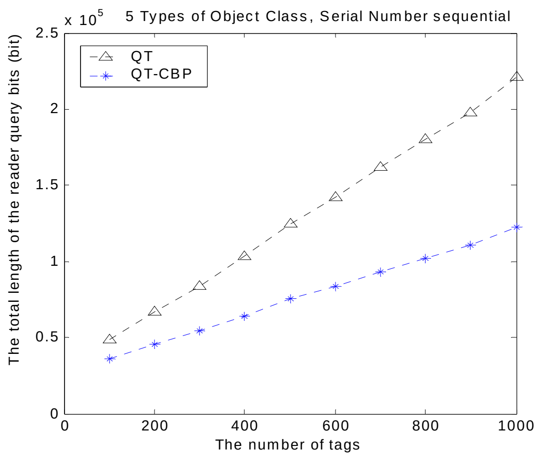

Fig. 6. The total length of queries in bits – Case 1

Case 2: The same Object Class with Random Serial Codes. In this simulation, the generated tags are different only in the Serial Code field. The 36-bit long Serial Code values are randomly assigned.

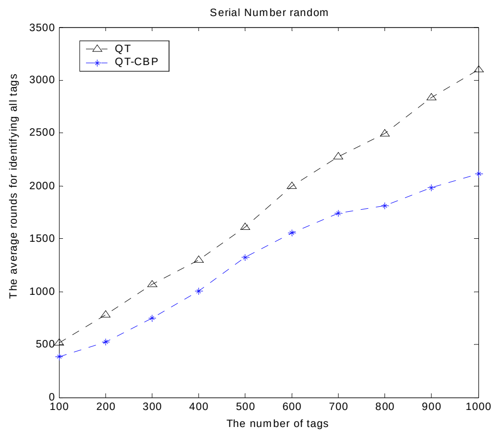

Fig. 7. The number of rounds required –Case 2

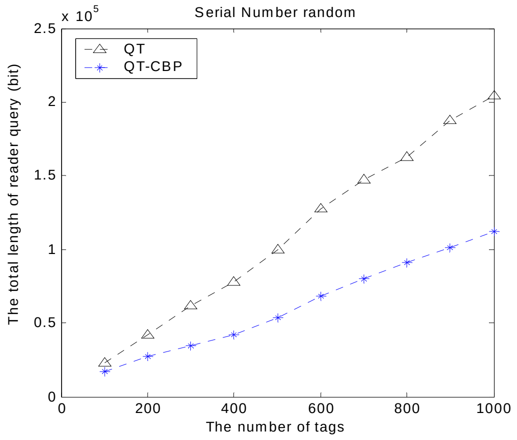

Fig. 8. The total length of queries in bits – Case 2

Case 3: The same Object Class with Sequential Serial Codes. In this simulation, the generated tags are different only in the Serial Code field and their Serial Codes are sequentially assigned.

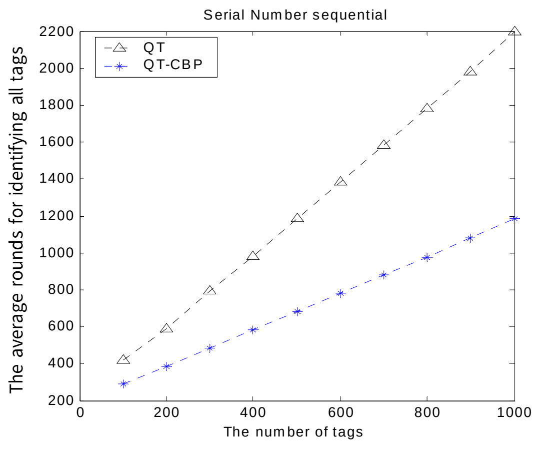

Fig. 9. The number of rounds required – Case 3

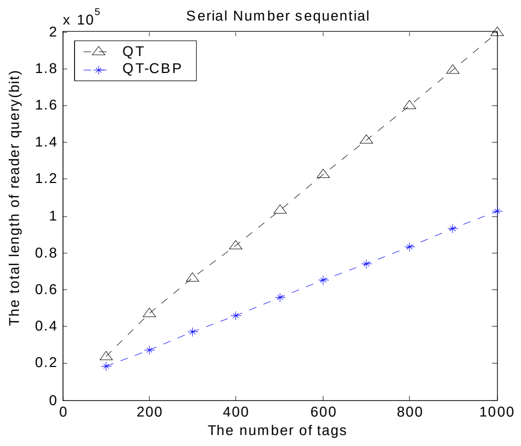

Fig. 10. The total length of queries in bits – Case 3

The previous simulation results show that QT-CBP is more efficient than QT. Note that the performance gap between QT and QT-CBP gets wider as the number of tags increases, and the most noticeable gap among the three cases is found in the case 3. This is because QT-CBP becomes more efficient as more tags have similar IDs.

5 Conclusions

In this paper, we suggest a new memoryless tree-based RFID tag anti-collision algorithm, QT-CBP, based on the QT algorithm developed at MIT's Auto-ID center. When a collision is detected, QT-CBP analyzes the number of collision bits and their positions to make more efficient tree traversal. Without changing tag operation, the QT-CBP identifies multiple tags with fewer reader queries and fewer collisions than QT, which makes faster identification possible. The performance of QT-CBP is evaluated through simulation and the results show that QT-CBP outperforms QT, especially on the condition where tags have similar tag IDs by having the same company or product ID prefixes.

References

1. Ching Law, Kayi Lee, and Kai-Yeung Sju, "Efficient Momoryless Protocol for Tag Identification", Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication, pp. 75-84, Boston, MA, August 2000.
2. A. Juels, R. Rivest, and M. Szydlo, "The Blocker Tag: Selective Tag Blocking of RFID Tags for Consumer Pri-vacy", Proceedings of the 10th ACM Conference on Com-puter and Communication Security, pp. 103~111, 2003.
3. Don R. Hush and Cliff Wood, "Analysis of Tree Algo-rithm for RFID Arbitration", Proceedings of IEEE Interna-tional Symposium on Information Theory, pp. 107~116, 1998.
4. Marcel Jacomet, Adrian Ehrsam, Urs Gehrig, "Contact-less identification device with anti-collision algorithm", Proceedings of IEEE Conference on Circuits, Systems, Computers and Communications, Athens, July, 1999.
5. Changsoon Kim, Kyunglang Park, Hiecheol Kim, Shindug Kim, "An Efficient Stochastic Anti-Collision Al-gorithm using Bit-slot Mechanism", PDP'2004, July 2004.
6. Harald Vogt, "Efficient Object Identification with Pas-sive RFID Tags", Proceedings of International Conference on Pervasive Computing, Zurich, 2002.
7. Auto-ID Center, "13.56 MHz ISM Banc Class 1 Radio Frequency Identification Tag Interface Specification, Ver-sion 1.0", Auto-ID Center, May, 2003.
8. Jihoon Myung and Wonjun Lee, "An Adaptive Mem-oryless Tag Anti-Collision Protocol for RFID Networks", Proceeding of the 24th IEEE Annual Conference on Com-puter Communications (INFOCOM 2005), Miami, Florida, March 2005.
9. Auto-ID Center, "EPCTM Tag Data Standards Version 1.3", Auto-ID Center. September 2005.
10. Auto-ID Center, "Draft protocol specification for a 900MHz Class 0 Radio Frequency Identification Tag", Auto-ID Center. February 2003.