Query Tree Algorithm for RFID Tag with Binary-Coded Decimal EPC

Ching-Nung Yang, *Senior Member, IEEE,* Li-Jen Hu, and Jia-Bin Lai

***Abstract*—A tag-collision problem in Radio Frequency Identification (RFID) system is the event that a reader cannot identify a tag if many tags respond to a reader at the same time. Although binary Electronic Product Code (EPC) is the most natural for a computer, most people are accustomed to the decimal system. In RFID applications, we need to convert binary EPC to decimal numbers. Since converting binary-coded decimal (BCD) data into the decimal numbers is much less complex than converting binary data into decimal numbers. This motivates us to represent EPC by BCD. However, using BCD-based EPC delivers two problems: (i) Is the existing query tree algorithm suitable for identifying BCD-based EPC? (ii) How do we design a new query tree algorithm to enhance the tag-identification efficiency? In this work, we solved the problems.**

***Index Terms*—RFID, EPC, tag collision, query tree.**

I. Introduction

A Tag-collision problem in Radio Frequency Identification (RFID) system is the identification problem when multiple tags respond to a reader simultaneously and the reader cannot differentiate these tags correctly. Tag collisions will degrade identification efficiency, and this unreliable identification will compromise the usefulness of RFID system. Up to date, several technologies on tag collision were proposed. There are two major types of anti-collision algorithms- one is ALOHA-based algorithm and the other is tree-based algorithm. ALOHA-based algorithm [1] reduces the tag collision, but has the starvation problem (a tag cannot be identified for a long time). Tree-based algorithm can solve this starvation problem. The well-known tree-based algorithms are binary tree (BT) and query tree (QT). QT algorithm does not need the additional memory and thus is referred to as the memory-less protocol.

In QT, a reader sends a prefix of EPC to query tags, and the tags matching the prefix respond. We extend the prefixes until only one tag responds. Some elegant QTs are briefly described in the following. Chiang *et al.* [2] proposed a prefix-randomized QT. A reader first scans the neighboring tags to determine which M-ary tree is suitably used for querying tags. After finishing queries by M-ary tree, a reader then uses binary QT (BQT) for inquiries. In adaptive query splitting algorithm [3], the authors used extra candidate queue to store the prefix bits of responded tags to speed up the identification process. A hybrid QT [4] used 4-ary QT and the slotted back-off mechanism to avoid collision. Choi et al. [5] used RN16 (a 16-bit random number) as tag's temporary ID. However, the short length of RN16 is not enough in real environments. Yang and He [6] further modified Choi *et al.*'s RN16-based QT to solve the problem of using short RN16. In an enhanced QT [7], the length of prefix code is adjusted dynamically according to the length of tag's ID. Cho *et al.* [8] proposed new QT to identify tags with consecutive serial number. In [9], two M-ary trees were combined as the unified QT to improve the identification efficiency. The authors adopted Manchester code in QT to find the location of different bit in responded strings, and thus the reader can skip the unnecessary queries [10].

All the above QTs were designed for tags with binary EPC. As we know, a 96-bit EPC code is a group of 96 bits, and has 2^{96} combinations of 1's and 0's to represent the tag's ID. A binary system is most natural for computer, and it is readily represented in today's system. However, most people are accustomed to the decimal system. A so-called binary-coded decimal (BCD) coding expresses each decimal digit (0 through 9) to a four-bit BCD code. A number with m decimal digits require $4m$ bit in BCD. For example, for 96 bits, we have 10^{24} combinations when using BCD representation. For RFID application, we need to convert 96-bit EPC to decimal numbers in back-end database to show the company ID, the product ID, and the serial number in decimal numbers. BCD numbers are decimal numbers and not binary numbers, although they use bits in their representation. Thus, converting BCD data into the decimal numbers is much less complex than converting binary data into decimal numbers. The problem of conversion will be more important in a large database management system. By the above observation, we can use BCD to represent EPC to speed up the conversion. In this work, we design new QT to identify tags with BCD-based EPC.

II. Motivation

Most people are accustomed to the decimal system. Therefore, when applying RFID, we need to convert EPC to user-friendly decimal numbers. BCD coding is a way to express each decimal digit to a four-bit BCD code (one of ten values, 0-9) individually, while a straight binary code takes the complete decimal number and express it in binary. Obviously, converting BCD data into the decimal numbers in computers is much less complex than converting binary data into decimal numbers. This motivates us to represent EPC by BCD in a tag to speed up the EPC to decimal conversion in back-end database. The ease of EPC conversion is especially important in a large database management system. Although BCD-based EPC determines from the count 0-$10^{24}-1$, which is lesser than $2^{96}-1$ (96-bit EPC), the possible values of 10^{24} are large enough for most applications. However, using BCD to represent EPC in RFID system will deliver the following

Manuscript received June 6, 2012. The associate editor coordinating the review of this letter and approving it for publication was Y.-D. Lin.

The authors are with the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan (C. N. Yang is the corresponding author: cnyang@mail.ndhu.edu.tw).

Digital Object Identifier 10.1109/LCOMM.2012.090312.121213

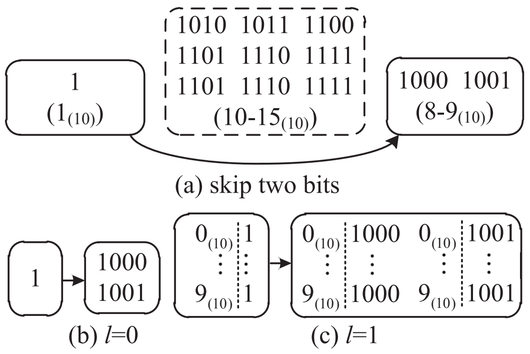

Fig. 1. The concept of Approach 1: (a) skip two bits by adding ($q000$) and ($q001$) in a queue (b) $l = 0$ (c) $l = 1$.

two problems: (1) Is the existing QT suitable for identifying BCD-based EPC? (2) How do we design new QT to enhance the tag-identification efficiency? Obviously, BCD-based EPC has some binary combinations, which do not occur in binary EPC. Therefore, BQT is not suitable for identifying tags with BCD-based EPC. In this work, we propose two approaches to resolve the identification of tags with random and consecutive serial numbers, respectively.

III. The Proposed QTAs

A. Approach 1

In BQT algorithm, the reader asks the tags whether any of their IDs contains a prefix same to the query strings q. If two or more tags answer (i.e. a collision is detected), the reader then appends bit 0 and 1 to generate the longer prefixes ($q0$) and ($q1$) in a queue. We repeat the query procedure until all tags are uniquely identified. Consider the identification of tags with BCD-based EPC by BQT. Suppose that a collision is detected and that a querying bit string q has the length $4l + 1,\ l \geq 0$, and its last bit is 1. In this situation, we can skip two bits and add ($q000$) and ($q001$) instead of ($q0$) and ($q1$) in a queue to reduce the interrogation cycles. As shown in Fig. 1(a), since BCD is composed of four-bit code groups ($0\text{-}9_{(10)}$), we can skip $10\text{-}15_{(10)}$. Figs. 1(b) and (c) show two cases for $l = 0$ and $l = 1$.

B. Approach 2

Since EPC is represented by BCD, we can apply 10-ary query tree (10-QT) to achieve the optimum performance. In 10-QT algorithm, we query a four-bit string q ($0\text{-}9_{(10)}$) each time. When tags collide, we need to add ($q0000$), ($q0001$), ($q0010$), ($q0011$), ($q0100$), ($q0101$), ($q0110$), ($q0111$), ($q1000$), and ($q1001$) in a queue. As we know, using the large M-ary tree implies the longer prefix. At this time, tags may have the same prefix with the small probability and thus reduce collisions, but increases the unnecessary inquiries. The invalid inquiries can be significantly reduced when tags IDs are consecutive. Consider a possible application scenario of RFID in [8]. A publisher, after printing, tens or hundreds of books with the same title are usually packaged in a box for delivery and distribution. These books in a box will have a common prefix (the same company ID and product ID), and their serial numbers are consecutive. Approach 2 works well for identifying tags with consecutive serial numbers.

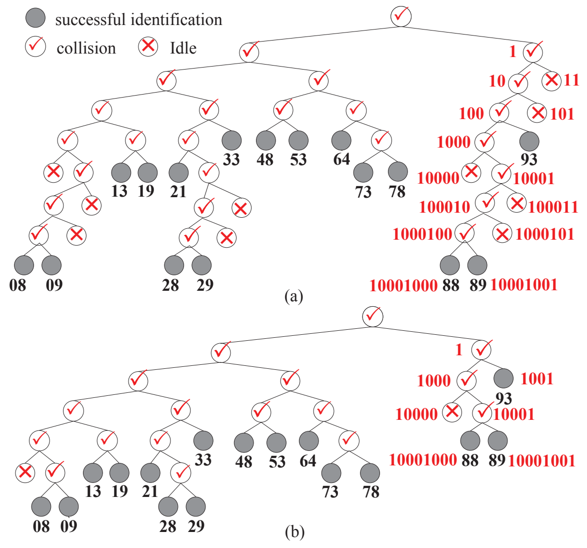

Fig. 2. Identification of tags with random serial number using: (a) BQT (b) Approach 1.

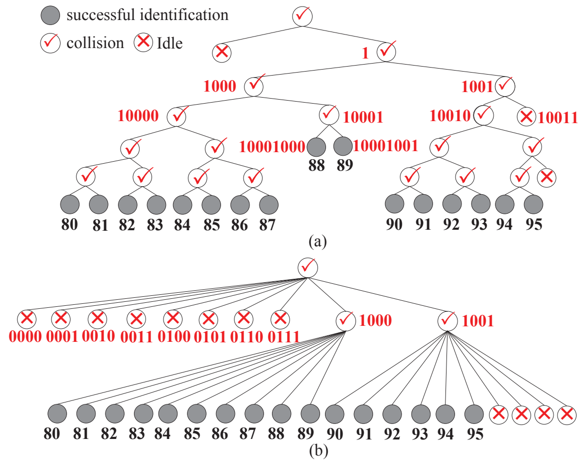

Fig. 3. Identification of tags with consecutive serial number using: (a) Approach 1 (b) Approach 2.

C. Examples

Two simple examples are conducted to briefly show how Approach 1 and Approach 2 enhance the identification. Two scenarios, tags with random and consecutive serial numbers, are shown in Example 1 and Example 2.

Example 1. Suppose that sixteen tags have tags ID in decimal format: 08, 09, 13, 19, 21, 28, 29, 33, 48, 53, 64, 73, 78, 88, 89 and 93, which are represented by two BCD codes. Fig. 2 shows identification of these tags by BQT and Approach 1. There are 51 total interrogation cycles for BOT, which includes 25 collision cycles, 10 idle cycles and 16 successful cycles. Approach 1 save 8 collisions and 8 idle cycles, and only need 35 interrogation cycles. For example, for identifying 88, 89, and 93, the query strings: (10), (11), (100), (101), (100010), (100011), (1000100), and (1000101), can be skipped in Approach 1. For this case, Approach 2 needs 61 interrogation cycles (6 collisions, 39 idle cycles, 16 successful cycles), which is even less ineffective than BQT.

Example 2. Suppose that sixteen tags have consecutive tags IDs in decimal format (80-95$_{(10)}$), which are represented by two BCD codes. As shown in Fig. 3, Approach 1 and Approach 2 have 37 interrogation cycles (18 collisions, 3 idle cycles, 16 successful cycles), and 31 interrogation cycles (3 collisions, 12 idle cycles, 16 successful cycles), respectively. Since Approach 2 uses 10-QT, it reduces the number of collisions from 18(Approach 1) to 3. On the other hand, BQT has 49 interrogation cycles (24 collisions, 9 idle cycles, 16 successful cycles).

All detail communication procedures between reader and tags in examples can be found in the supplementary manuscript on the website (http://cis.csie.ndhu.edu.tw/cnyang/bcd.pdf).

IV. Performance Evaluation

A. Experimental Results

Two experiments demonstrate that our approaches works well in both scenarios that tags' serial numbers are random and consecutive, respectively. The number of tags is n, from 100 to 10,000. For each n, we repeat the same experiment ten times to calculate the average number of collision cycles N_C, the average number of idle cycles N_I. The average number of total interrogation cycles is $N_T = n+N_C+N_I$. In we consider the RFID warehouse distribution [9]. It is reasonable to assume that the EPC data of most items from the same warehouse will be very similar since the items are manufactured by the same company, and are stacked together in a large warehouse. As we know, EPC embraces four sections: header (H: 8 bits), GMN (G: 28 bits), object class (O: 24 bits), and serial number (S: 36 bits). Here, we test tags with the identical l_f=60 bits (H+G+O) cascaded with the random (Experiment A) and consecutive (Experiment B) serial number l_s=36 bits (S).

Experiment A. Three algorithms are tested: BQT, Approach 1, and Approach 2. In this experiment, all tags have the identical l_f=60 bits and the random serial number l_s=36 bits in BCD format. Approach 1 can skip some queries by adding (q000) and (q001) in a queue, such that it can reduce the total interrogation cycles when compared with BQT. For example, for n=100, BQT has N_T=475 while Approach 1 saves 26 cycle and has N_T=449. On the other hand, Approach 2 using 10-QT has the less collisions N_C=60 when compared to N_C=237 (BQT) and N_C=224 (Approach 1). However, Approach 2 has the large idle cycles N_I=438. Finally, it has N_T=598.

Experiment B. Redo Experiment A, but use tags with identical l_f=60 bits and consecutive serial number l_s=36 bits. We generate the consecutive serial number by using $(seed + 1) \sim (seed + n)$, where $seed$ is randomly chosen from [0, 999]. Approach 2 can significantly reduce the idle cycles for the consecutive serial number, and still has the less collisions. For n=100, Approach 2 has N_C=35, while BQT and Approach 1 have N_C=215 and 189, respectively. Although using 10-QT may increase the idle cycles, Approach 2 still has the less idle cycles N_I=215 due to the consecutive serial number. Finally, Approach 2 has N_T=350, but BQT and Approach 1 have N_T=431 and 379.

The average interrogation cycles using BQT, Approach 1 and Approach 2 are shown in Fig. 4. The simulation results confirm that Approach 1 and Approach 2 outperform BQT for tags with random and consecutive numbers, respectively.

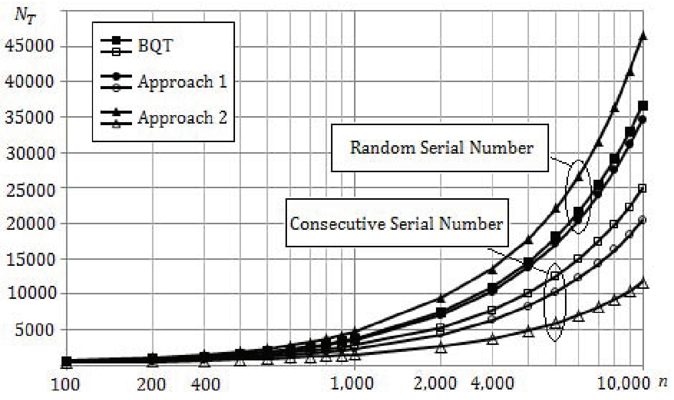

Fig. 4. Average number of interrogation cycles using BQT, Approach 1 and Approach 2 for random and consecutive serial numbers.

B. Performance Analysis for Approach 1 and Approach 2

In this section we analyze the number of total interrogation cycles for our approaches. Suppose that the number of total interrogation cycles of BQT, Approach 1, and Approach 2 are $N_T^{(B)}$, $N_T^{(1)}$, and $N_T^{(2)}$, respectively, and n tags are represented by m BCD codes, where $m = \log_{10} n$.

Lemma 1. The difference $N_T^{(B)} - N_T^{(1)}$ are $0.22n + 5.78$ and $0.44n + 5.56$ for tags with random and consecutive serial number, respectively.

Proof. Suppose that q is a query bit string with the length $4l+1$ and its last bit is 1. Here, we consider two cases: (i) the fixed prefix l_f=60 bits and the random serial number l_s=36 bits (ii) the fixed prefix l_f=60 bits and the consecutive serial number l_s=36 bits. As shown in Fig. 5(a), the prefix l_f has 15 BCD codes ($B_1 - B_{15}$) and the first bit in B_i has the value of b_i=0 and 1 with the probability 0.8 and 0.2, respectively. When b_i=1, we can save 4 cycles with half probability (note: we can save 4 cycles for collision but cannot save any cycles when it idles). Therefore, the average reduced number of cycles for the fixed prefix l_f is

$$0.5 \times \sum_{i=0}^{15} \left(0.2^i \times 0.8^{(15-i)} \times \begin{pmatrix} 15 \\ i \end{pmatrix} \times 4i \right) = 6. \quad (1)$$

Case (i): The random l_s=36 bits is shown in Fig. 5(b). There are 10^l possible combinations for q_l, $0 \le l \le (m-1)$, which collide with half probability. Thus, the average reduced number of interrogation cycles using Approach 1 is

$$0.5 \times \sum_{i=0}^{m-1} \left(4 \times 10^i\right) = 0.22n - 0.22. \quad (2)$$

From (1) and (2), we have $N_T^{(B)} - N_T^{(1)} = 0.22n + 5.78$.

Case (ii): The consecutive l_s=36 bits is shown in Fig. 5(c). The string q_l, $(9-m) \le l \le 8$, will always collide, because the serial number is consecutive. Thus, we have

$$N_T^{(B)} - N_T^{(1)} = 6 + \sum_{i=0}^{m} \left(4 \times 10^i\right) = 0.44n + 5.56. \quad (3)$$

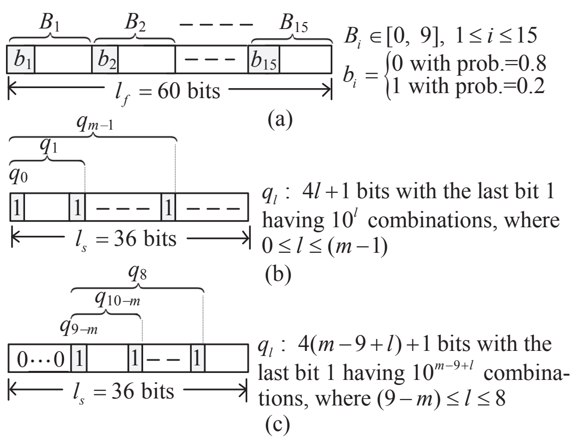
B_1 B_2 B_{15}
b_1 b_2 b_{15}
$l_f = 60$ bits
$B_i \in [0, 9],\ 1 \le i \le 15$
$b_i = \begin{cases} 0 \text{ with prob.}=0.8 \\ 1 \text{ with prob.}=0.2 \end{cases}$
(a)
q_{m-1} q_1 q_0
$l_s = 36$ bits
q_l : $4l+1$ bits with the last bit 1 having 10^l combinations, where $0 \le l \le (m-1)$
(b)
q_8 q_{10-m} q_{9-m}
$0\cdots0$
$l_s = 36$ bits
q_l : $4(m-9+l)+1$ bits with the last bit 1 having 10^{m-9+l} combinations, where $(9-m) \le l \le 8$
(c)

Fig. 5. Diagrammatic representations of l_f and l_s: (a) B_1-B_{15} (b) q_l of random serial number (c) q_l of random serial number.

Lemma 2. The values of $N_T^{(B)} - N_T^{(2)}$ and $N_T^{(1)} - N_T^{(2)}$ are $n^{1.2} - 0.11n + 2\log_{10} n - 48.78$ and $n^{1.2} - 0.55n + 2\log_{10} n - 54.34$ for tags with consecutive serial number.

Proof. For the fixed prefix l_f=60 bits, we have 60 collisions and 60 idle cycles for BQT, and 15 collisions and $(10-1) \times 15$=135 idle cycles for Approach 2. As shown in Fig. 5(c), for the previous $(9-m) \times 4$ zero bits in the consecutive serial number l_s=36 bits, we have $(9-m) \times 4$ collisions and $(9-m) \times 4$ idle cycles for BQT, and $(9-m)$ collisions and $(10-1) \times (9-m)$= idle cycles for Approach 2. The numbers of collisions caused by other m consecutive BCD codes for BQT and Approach 2 are $(1+2+2^2+...+2^{4m-1}) = n^{1.2} - 1$ and $(1+10+10^2+...+10^{m-1}) = (n-1)/9$. From the above, we can determine and $N_T^{(B)}$ and $N_T^{(2)}$ as follows.

$$\begin{aligned} N_T^{(B)} &= 60 + 60 + (9-m) \times 4 + (9-m) \\ &\times 4 + n^{1.2} - 1 = n^{1.2} - 8\log_{10} n + 191. \end{aligned} \tag{4}$$

$$\begin{aligned} N_T^{(2)} &= 15 + 9 \times 15 + (9-m) \times 1 + (9-m) \\ &\times 9(n-1)/9 = 0.11n - 10log_{10}n + 239.78. \end{aligned} \tag{5}$$

By (3), (4), and (5), we have

$$\begin{aligned} N_T^{(B)} - N_T^{(2)} &= n^{1.2} - 0.11n + 2\log_{10}^{n} - 48.78, \\ N_T^{(1)} - N_T^{(2)} &= n^{1.2} - 0.55n + 2\log_{10}^{n} - 54.34. \end{aligned} \tag{6}$$

From Lemma 2, we have $N_T^{(B)} - N_T^{(2)} > 0$ for $n \ge 26$ and $N_T^{(1)} - N_T^{(2)} > 0$ for $n \ge 35$, and thus $N_T^{(2)} < N_T^{(B)}$, $N_T^{(2)} < N_T^{(1)}$ for the large n.

V. Conclusion

It seems that no one studies the identification of tags with BCD-based EPC. BCD-based EPC can speed up the conversion of EPC to decimal numbers in back-end database, and this ease of EPC conversion is especially important in a large database management system. In this work, we take the lead in studying new QT to efficiently identify tags with BCD-based EPC. Two approaches are proposed to resolve the identification of tags with random and consecutive serial numbers, respectively. Theoretical analyses and experimental

TABLE I
N_C, N_I and N_T for Tags with Random Serial Number

n	BQT			Approach 1			Approach 2		
	N_C	N_I	N_T	N_C	N_I	N_T	N_C	N_I	N_T
100	237	138	475	224	125	449	60	438	598
200	426	227	853	405	206	811	109	78	1097
300	598	299	1197	565	266	1131	151	1058	1509
400	778	379	1557	737	338	1475	192	1331	1923
500	960	461	1921	909	410	1819	235	1612	2347
600	1138	539	2277	1077	478	2155	280	1919	2799
700	1327	628	2655	1254	555	2509	328	2253	3281
800	1513	714	3027	1431	632	2863	378	2610	3788
900	1707	808	3415	1613	714	3227	427	2945	4272
1000	1889	890	3779	1781	782	3563	476	3286	4762
5000	9076	4077	18153	8585	3586	17171	2225	15024	22249
10000	18320	8321	36641	17283	7294	34587	4643	31788	46431

TABLE II
N_C, N_I and N_T for Tags with Consecutive Serial Number

n	BQT			Approach 1			Approach 2		
	N_C	N_I	N_T	N_C	N_I	N_T	N_C	N_I	N_T
100	215	116	431	189	90	379	35	215	350
200	336	137	673	289	90	579	46	213	459
300	458	159	917	387	88	755	57	211	568
400	582	183	1165	490	91	981	68	215	683
500	703	204	1407	589	90	1179	79	212	791
600	825	226	1651	688	89	1377	90	214	904
700	948	249	1897	788	89	1577	102	215	1017
800	1069	270	2139	887	88	1775	113	214	1127
900	1191	292	2383	988	89	1977	124	213	1237
1000	1314	315	2519	1088	89	2177	135	213	1348
5000	6199	1200	12399	5084	85	10169	579	208	5787
10000	12314	2315	24629	10088	89	20177	1135	214	11349

results reveal that our approaches have better performance than BQT.

References

[1] S. R. Lee, S. D. Joo, and C. W. Lee, "An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification," in *Proc. 2005 IEEE International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 166–172.

[2] K. W. Chiang, C. Hua, and T. S. Yum, "Prefix-randomized query-tree protocol for RFID system," in *Proc. 2006 IEEE International Conference on Communication*, pp. 1653–1657.

[3] J. Myung, W. Lee, J. Srivastava, and T. K. Shih, "Tag-splitting: adaptive collision arbitration protocols for RFID tag identification," *IEEE Trans. Parallel Distrib. Syst.*, pp. 763–775, June 2007.

[4] J. Ryu, H. Lee, Y. Seok, T. Kwon, and Y. Chioi, "A hybrid query tree protocol for tag collision arbitration in RFID system," in *Proc. 2007 IEEE International Conference on Communications*, pp. 5981–5986.

[5] J. H. Choi, D. Lee, and H. Lee, "Query tree-based reservation for efficient RFID tag anti-collision," *IEEE Commun. Lett.*, vol. 11, pp. 85–87, Jan. 2007.

[6] C. N. Yang and J. Y. He, "An effective 16-bit random number aided query tree algorithm for RFID tag anti-collision," *IEEE Commun. Lett.*, vol. 15, pp. 539–541, May 2011.

[7] C. H. Hsu, C. H. Yu, Y. P. Huang, and K. J. Ha, "An enhanced query tree (EQT) protocol for memoryless tag anti-collision in RFID systems," in *Proc. 2008 IEEE International Conference on Future Generation Communication and Networking*, pp. 427–432.

[8] J. K. Cho, J. D. Shin, and S. K. Kim, "RFID tag anti-collision protocol: query tree with reversed IDs," in *Proc. 2008 International Conference on Advanced Communication Technology*, pp. 225–230.

[9] P. Pupunwiwat and B. Stantic, "Unified q-ary tree for RFID tag anti-collision resolution," in *Proc. 2009 Australasian Database Conference*, pp. 49–58.

[10] X. L. Jia, Q. Y. Feng, and C. Z. Ma, "An efficient anti-collision protocol for RFID tag identification," *IEEE Commun. Lett.*, vol. 14, pp. 1014–1016, Nov. 2010.