Tweaked Binary Tree Algorithm to Cope with Capture Effect and Detection Error in RFID Systems

Chuyen T. Nguyen*, Anh Tuan H. Bui*, Vuong V. Mai† and Anh T. Pham†
* School of Electronics and Telecommunications, Hanoi University of Science and Technology, Vietnam
E-mail: chuyen.nguyenthanh@hust.edu.vn, buihoanganhtuank56@gmail.com
† Computer Communications Lab., University of Aizu, Japan
E-mail: {d8161107, pham}@u-aizu.ac.jp

***Abstract*—This paper proposes a new RFID binary tree-based identification protocol, namely Tweaked Binary Tree (TBT), to cope with hidden tag problem caused by capture effect and detection error phenomena. In TBT, the whole identification process is divided into multiple binary tree cycles, and the hidden tags in a cycle are checked and re-transmitted in the first slot of the next one. The average number of slots for a successful detection of a tag, and the tag loss rate, defined as a ratio between the number of missing tags and the whole tag cardinality, are theoretically analyzed. Computer simulations are also performed to validate the theoretical analysis. We also confirm the superiority of the proposed method in comparison with a conventional General Binary Tree (GBT) one.**

I. Introduction

Owing to the low cost, low power consumption and efficiency, Radio Frequency Identification (RFID) technology has been widely implemented in many applications of identifying objects automatically such as supply chain, medical tracking, and security [1]-[3]. In general, an identification process is initialized when RFID readers send a request to tags. Each tag, after receiving the request, sets its counter by a random number, and responds its identity (ID) to readers as the counter is zero. When multiple tags reply to a reader simultaneously, signal collision happens and in this case, tags's replied packets are usually corrupted or lost [4]. To overcome this challenge, a number of anti-collision algorithms/protocols has been proposed, which can be classified into two main practical approaches: Aloha-based and tree-based.

On one hand, Aloha-based protocols permit tags to respond to the reader in a frame of time slots, randomly [5]-[8]. The protocols are well known in EPC-global standards and Philips smart label IC data sheet [9],[10]. On the other hand, tree-based algorithms split colliding tags into two groups, and tags in only one group reply to readers in the next time slots [11]-[15]. This splitting process is repeated until no collisions are detected. There are two types of the tree-based algorithms, which are binary tree and query tree. While the binary tree-based algorithms resolve colliding tags with a splitting probability of 0.5, the query tree-based ones compare a transmitted binary string with prefix of IDs to select tags [9].

The unreliable communication between readers and tags due to impacts of wireless channel impairments [16]-[18], however, makes the algorithms inefficient. Indeed, under the impact of channel impairments, the reader might wrongly detect a slot with a tag's transmission as in an empty state (no tags transmit) if its received signal-to-noise ratio (SNR) is below the reader's sensitivity threshold. In another case, the reader might still detect a tag involved in collision if its received Signal-to-Interference plus Noise Ratio (SINR) is higher than the threshold. The sooner is called detection error (D.E.) or tag missing while the later is the capture effect (C.E.).

Over the years, there has been a number of studies for both Aloha-based [19]-[21] and tree-based algorithms [22],[23] to cope with the C.E. Also, the studies in [24] and [25] suggest a simple solution to deal with the D.E., in which tags are read multiple times and thus, a probability of missing tags is significantly decreased. However, the C.E. and D.E. are not considered simultaneously. Recently, Nguyen *et al*. [26],[27] studies both the phenomena, also for Aloha-based algorithms. To the best of our knowledge, a study of tree-based algorithms dealing with both the D.E. and C.E. has not been investigated so far.

In this paper, we therefore study tree-based anti-collision algorithms under effects of both the C.E. and D.E. In particular, we first introduce a conventional general binary tree (GBT) method, which was proposed to cope with the C.E. only. Then, we newly propose a binary tree-based algorithm, namely, Tweaked Binary Tree (TBT) to find undetected tags hidden due to both the phenomena. Our algorithm is organized into several binary cycles, in which the tags hidden by either the capture effect or the detection error in one cycle are checked and recognized in next cycles. The average number of slots for a successful detection of a tag, and the tag loss rate, defined as a ratio between the number of missing tags and the whole tag cardinality, are theoretically analyzed and evaluated via computer simulations. The results are also compared with that of the GBT method to show the effectiveness of the proposed method.

The rest of the paper is organized as follows. In section II,

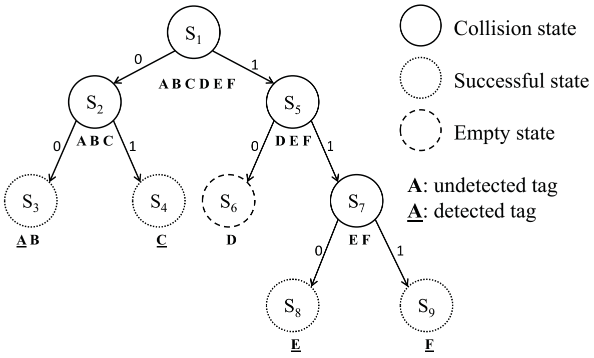

Fig. 1. Binary tree with detection error and capture effect

we describe a general RFID system model. The conventional method GBT, and the proposed algorithm TBT are presented in section III. The mathematical analysis of the proposed TBT is explained in Section IV. Section V shows computer simulation results, and conclusions are drawn in section VI.

II. System Model

Our considered model consists of a reader and n tags. In order to cope with the collision problem, binary tree-based anti-collision protocols/algorithms are employed. In particular, tags involving in a collision are randomly divided into two subsets. While tags in one subset transmit their signal to the reader in the next time slot/node, the others have to wait for their transmissions until the first subset is completely identified. This process is finished when all the tags are detected.

Each time slot/node is supposed to be recognized by the reader as in one of three states, which is *empty* or *successful*, or *collision*. In particular, a slot is *empty* if there is no transmissions during that time, while the slot in which the reader successfully decodes a tag's ID is in *successful* state. On the other hand, when multiple tags transmit simultaneously, this slot might be detected as *collision* by a power threshold detection scheme or Cyclic Redundancy Check (CRC). Besides, we assume that the reader and each tag have a reader counter (RC) and a tag counter (TC), respectively, which are both initially set to zero. The reader uses RC to finish the reading process (when $RC = -1$), while each tag can determine how many more time slots to wait until its transmission (transmits when $TC = 0$). After being detected and acknowledged by the reader, each tag sets its TC to be -1 and does not respond to the reader in next time slots.

Due to fading effects, the D.E. and the C.E. phenomena might happen in a node with certain probabilities, which we denote by β, and α, respectively. Here, we ignore effects of path-loss which could be valid for indoor RFID applications with a flat Rayleigh fading model, for example [28]. Therefore, the C.E. probability α and the D.E. probability β are assumed to be identical for all nodes with multiple transmissions and one transmission, respectively. On the other hand, due to the diversity effects, the received SNR at the reader in case of multiple transmissions is assumed to be large enough so that the D.E. does not happen. *Collision* slots are, thus, supposed to be easily detected from *empty* and *successful* ones, which has been also adopted by works in [27].

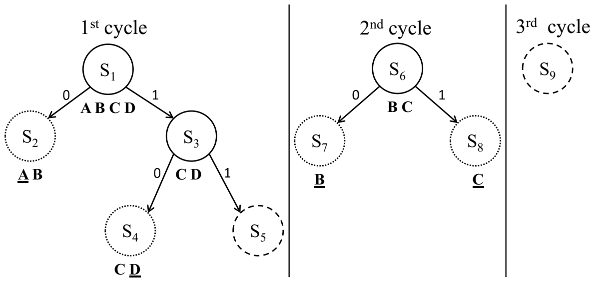

Fig. 2. General binary tree with capture effect

Figure 1 shows a simple example of the binary tree-based algorithm to identify 6 tags, namely A, B, C, D, E, and F, under impacts of wireless channels. We can see that, due to the C.E., tag A is detected at node S_3, although tag B also transmits during this time. In this case, the reader adopts S_3 as in a *successful* state, and thus, tag B is lost. Also, tag D is not recognized at node S_6 because of the D.E.. Our purpose is to propose a new binary tree-based identification algorithm coping with both the C.E. and D.E..

III. Proposed Method

A. Conventional general binary tree [23]

To cope with the C.E., the general binary tree (GBT) divides the identification process into several cycles where each cycle is corresponding to a binary tree. The undetected tags hidden by the C.E. in the current cycle are found in the first time slot of the next one.

To implement the algorithm, the reader is required to maintain a special Boolean parameter i.e., *Extension Flag (EF)* to check whether a slot is reserved to find hidden tags. Specifically, the parameter is set to *false* at the beginning of a cycle and turns *true* upon encountering a successful slot. In this case, the reader keeps its *RC* unchanged, which implies a reservation of a time slot in the next cycle.

On the other hand, the reader broadcasts the identified ID (IID), its *RC* value, and *Successful* state when successfully detecting a tag. Transmitting tags with $TC= 0$ compare their IDs with the IID. If a tag's ID matches, its TC becomes -1, otherwise it becomes RC value to re-transmit in the reserved slot.

Figure 2 describes an example of the GBT with the C.E.. Tags B and C hidden by the C.E. in slots S_2 and S_4, respectively, re-transmit in the first time slot of next cycle (i.e. slot S_6). Table I explains the operation of the whole identification process in details in each time slot. GBT is proven to cope well with the C.E., however, the D.E. is not considered in the algorithm and thus, hidden tags by this phenomenon are not studied also.

Slot	Parameters						Feedback
	RC	*EF*	TC_A	TC_B	TC_C	TC_D	
1	0	0	0	0	0	0	Collision
2	1	0	0	0*	1	1	Successful
3	1	1	-1	1	0	0	Collision
4	2	1	-1	2	0*	0	Successful
5	1	1	-1	1	1	-1	Empty
6	0	0	-1	0	0	-1	Collision
7	1	0	-1	0	1	-1	Successful
8	1	1	-1	-1	0	-1	Successful
9	0	0	-1	-1	-1	-1	Empty

* denote tags hidden by C.E.

TABLE I
GBT IDENTIFICATION PROCESS

B. Proposed tweaked binary tree (TBT)

In this section we describe a newly proposed Binary Tree-based algorithm, which we call Tweaked Binary Tree (TBT), to cope with both the C.E. and D.E.. The algorithm also divides the identification process into multiple Binary Tree cycles, in which tags hidden by either the C.E. or the D.E. in any cycle are expected to re-transmit in the first slot of the very next cycle.

In order to do this, we assume that the reader maintains two parameters: a boolean parameter *Extension Flag* (*EF*) and another *Extra Cycle* (*EC*). The former is used to check whether a slot is reserved for hidden tags. Specifically, *EF* is set to be *false* at the beginning of a cycle and turns *true* upon encountering either *Successful* or *Empty* slot. If such slots are detected, while *EF* = *false*, *RC* remains unchanged. This implies a reservation of a slot in the next cycle. On the other hand, if *EF* = *true*, *RC* is decreased by 1 for each *Empty* or *Successful* slot, while increased by 1 for a *Collision* slot. When a cycle is terminated and *RC*= 0, a new identification cycle will be performed.

The parameter *EC* is stored by the reader to avoid an infinite loop of cycles. In particular, at the beginning of a cycle (*RC*= 0), if no responses is detected, while *EC* = 0, *RC* becomes -1 and the identification process is finished. Otherwise (*EC*> 0), *RC* is kept at 0, and *EC* is decreased by 1. In other words, *EC* is the number of slots the reader uses to check the last tag hidden by the D.E., if any, before terminating the identification process. Without this parameter, *RC* is always equal to 0 since *EF* = *false*.

Tags whose *TCs* equal to 0 transmit their IDs upon receiving the request message from the reader. They, then, receive a feedback message that includes information of the IID, state of the slot, and the *RC*. In this case, a *Collision* state requires transmitting tags to randomly add either 0 or 1 to their *TCs* while the others increase their *TCs* by 1. In case of *Successful* state, transmitting tags compare their IDs with the IID. If a tag's ID matches the IID, its $TC = -1$, otherwise it becomes *RC* value. Other tags decrease their *TC* by 1. Finally, *Empty* state asks transmitting tags to set their *TCs* to be *RC* value, whereas the others decrease their *TCs* by 1. These settings help hidden tags to re-transmit in the first slot of the next cycle. We summarize the performance of the reader and tags via pseudocode in Table II.

Reader

```
RC = 0, EC=1
Send start command
While(RC≥0)
If(RC=0)
    EF = false
Listen to signal:
If no signal
    If(RC>0)
        If(EF=false)
            EF = true
        Else
            RC = RC - 1
    Else
        If(EC>0)
            EC = EC - 1
        Else
            RC = RC - 1
    Respond: (Empty,RC)
Else
    Try to decode ID from signals:
    If an ID is decoded
        If(EF = false)
            EF = true
        Else
            RC = RC - 1
        Respond: (Successful,IID,RC)
    Else
        RC = RC + 1
        Respond: (Collision)
```

Tag

```
TC = 0
Received start command
While(TC ≥0)
If(TC=0)
    Send its ID
    Received feedback f(IID,RC)
    If(f=Successful)
        If ID = IID
            TC = TC - 1
        Else
            TC = RC
    Else
        If(f=Collision)
            TC = TC + rand(0,1)
        Else
            TC = RC
Else
    Received feedback f(IID,RC)
    If(f=Collision)
        TC = TC + 1
    Else
        TC = TC - 1
```

TABLE II
TBT PSEUDOCODE FOR READER AND TAG

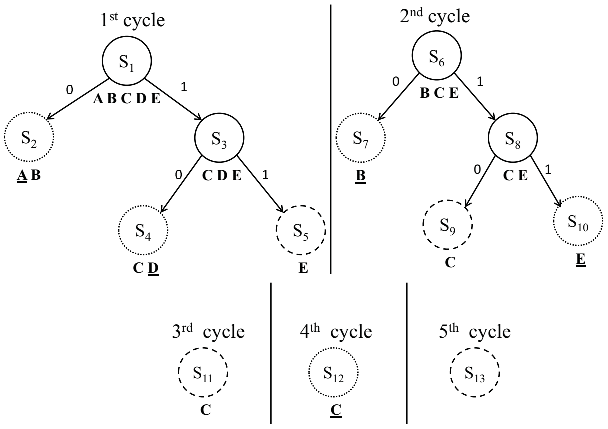

Fig. 3. Tree presentation of TBT

Figure 3 shows an example of the performance of TBT, where the reader tries to detect tags A, B, C, D, and E. In slot S_2, although both tags A and B transmit their IDs, tag A is recognized due to the C.E.. The reader sets *EF* to be *true* and broadcasts a feedback of (*Successful*, IID = ID_A, *RC*). Tag B, then, knows that it has been hidden, and thus, sets its counter to be *RC* to re-transmit in the first slot of the next cycle (i.e. slot S_6). Besides, tag A knows that it has been identified, and its counter becomes -1. Similarly, in slot

Slot	Parameters								Feedback
	RC	EF	EC	TC_A	TC_B	TC_C	TC_D	TC_E	
1	0	0	1	0	0	0	0	0	Collision
2	1	0	1	0	0*	1	1	1	Successful
3	1	1	1	-1	1	0	0	0	Collision
4	2	1	1	-1	2	0*	0	1	Successful
5	1	1	1	-1	1	1	-1	0**	Empty
6	0	0	1	-1	0	0	-1	0	Collision
7	1	0	1	-1	0	1	-1	1	Successful
8	1	1	1	-1	-1	0	-1	0	Collision
9	2	1	1	-1	-1	0**	-1	1	Empty
10	1	1	1	-1	-1	1	-1	0	Successful
11	0	0	1	-1	-1	0**	-1	-1	Empty
12	0	0	0	-1	-1	0	-1	-1	Successful
13	0	0	0	-1	-1	-1	-1	-1	Empty

* denotes tags hidden by C.E.

** denotes tags hidden by D.E.

TABLE III

TBT IDENTIFICATION PROCESS

S_4, only tag D is identified, while tag C re-transmits in slot S_6. On the other hand, in slot S_5, tag E is hidden because of the D.E.. In this case, tag E knows this fact since the reader's feedback is (*Empty*, *RC*). Therefore, tag E also sets its counter to be *RC* to re-respond in slot S_6.

In the second and third cycles, tag C is still not detected in slots S_9 and S_{11}, respectively, because of the D.E.. In slot S_{11}, EC= 1 and thus, there is another opportunity for tag C to re-transmit in slot S_{12} (RC= 0). In this slot, it is identified successfully. However, the reader does not know if slot S_{12} is experiencing the C.E.. Therefore, slot S_{13} is reserved and in this case, *RC* becomes -1 since $EC = 0$, which terminates the identification process. Table III presents detailed information about parameters in each slot.

IV. Mathematical Analysis

In this section, we mathematically analyze the performance of TBT algorithm. Two main performance metrics: the average number of slots for a successful detection of a tag and the tag loss rate denoted by η and γ, respectively, are considered. Here, the ratio of the number of missing tags to the total number of tags is regarded as the loss rate.

We consider a slot with n unrecognized tags, and let $L(n)$ denote the number of consumed slots in a TBT cycle to find n tags. When the C.E. happens (with a probability of α), it takes only one slot at the current cycle since a tag is successfully detected. Otherwise, n tags collide (with a probability of $1-\alpha$), and are randomly divided into two subsets, which, respectively, contain i and $n-i$ tags. Therefore, $L(n)$ can be written as follows

$$\begin{aligned} L(n) &= \alpha + (1-\alpha)\,2^{-n}\sum_{i=0}^{n}\binom{n}{i}[1+L(i)+L(n-i)] \\ &= \frac{1+2^{1-n}(1-\alpha)\sum_{i=0}^{n-1}\binom{n}{i}L(i)}{1-2^{1-n}(1-\alpha)}, \end{aligned} \tag{1}$$

where $L(0) = 1$ and $L(1) = 1$.

We can see from (1) that in order to calculate the total number of consumed slots in the identification process, which we denote by L, we need to find u_j and $L(u_j)$. Here, u_j is defined as the number of unrecognized tags at the beginning of the j-th cycle for $j = 1, 2, ..., m$, and m refers to the last cycle. In this case, the number of recognized tags during the j-th cycle, which we denote by r_j, must be known also. We have

$$u_j = n - \sum_{k=1}^{j-1} r_k, \tag{2}$$

$$r_j = L_1(u_j), \tag{3}$$

where $L_1(u_j)$ is defined as the number of successful slots in a TBT cycle with u_j unrecognized tags. $L_1(u_j)$ (or $L_1(n)$ for convenience) can be calculated as the same way as (1)

$$L_1(n) = \alpha + (1-\alpha)\,2^{-n}\sum_{i=0}^{n}\binom{n}{i}[L_1(i)+L_1(n-i)], \tag{4}$$

where $L_1(0) = 0$ and $L_1(1) = 1-\beta$. By using (2), (3) and (4) alternatively, we obtain $r_1, u_2, r_2, ..., u_m$ and r_m ($u_1 = n$).

Finally, L can be calculated as

$$L = \sum_{j=1}^{m} L(u_j) + \epsilon, \tag{5}$$

where ϵ is the average number of extra slots to deal with the last missing tag due to the D.E.. As mentioned in Sect.III-B, TBT uses *EC* slots to check for the missing tag after an empty slot is detected, while RC= 0 (in this slot, if EC= 0, the identification process will be terminated). Therefore, $\epsilon \in [(EC+1)(EC+2)]$. The average number of slots for a successful detection of a tag is written as

$$\eta = \frac{L}{n}. \tag{6}$$

Regarding to the tag loss rate γ, the number of missing tags after the whole identification process cannot exceed one. The main reason is that all hidden tags in one cycle will re-transmit in the first slot of the next one, while we assume that the D.E. does not happen at slots with multiple tags. In this case, the probability of losing the last tag with $EC = 0$ would be less than β. Also, choosing $EC = 1$ would make that probability drop below β^2, which is negligible with small values of β. Therefore, we will skip the analysis of γ, and verify this result by computer simulations.

V. Numerical Results

In this section, the average number of slots to detect a tag (η) and the tag loss rate (γ) of the proposed TBT algorithm are evaluated via computational simulations with 10,000 runs. The results are also compared with those of GBT under the same conditions. In all cases, we assume that $n = 200$ and EC= 1 for simplicity.

In Fig. 4, we plot theoretical and simulation results of the number of TBT slots used to detect 200 tags with different

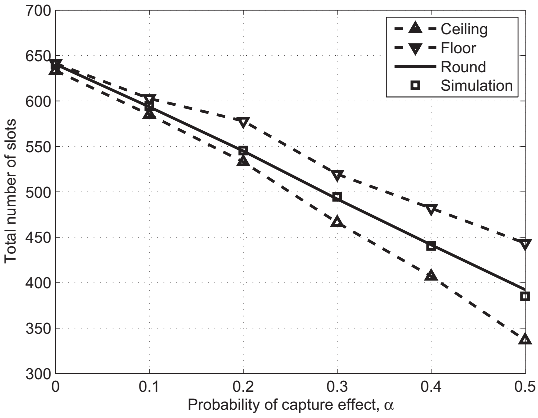

Fig. 4. The number of TBT slots used to detect 200 tags, given $\beta = 0.1$

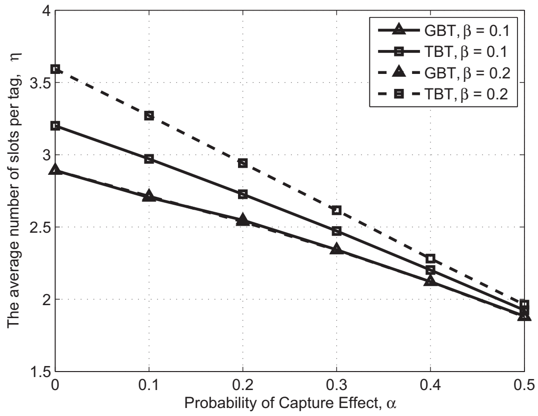

Fig. 5. The average number of slots η w.r.t. α, given $\beta = 0.1\,(0.2)$

values of α, given $\beta = 0.1$. Since the number of recognized tags in the j-th cycle (r_j) must be an integer, we use three functions in (3), namely, ceiling $\lceil r_j \rceil$, floor $\lfloor r_j \rfloor$, and round $[r_j]$ (the nearest integer of r_j) to obtain the theoretical results. We can see that the simulation result lies between theoretical ones using ceiling and floor functions, and it matches with the one using the round function. This is because, ceiling function causes an overestimation of number of recognized tags, while floor function causes an underestimation, which shows an lower bound and upper bound of the identification delay (the number of consumed slots, in other words), respectively. Also, as α increases, although the number of TBT cycles also increases, the number of consumed slots in each cycle drops significantly, which leads to a lower overall identification delay.

In Fig. 5 we plot the average number of slots per tag with respect to different values of α, given $\beta = 0.1\,(0.2)$, of all algorithms. We can see that when α increases, the period of identification in each cycle is decreased thanks to the C.E., and thus, the number of consumed slots is decreased in all methods. In case α is large enough, the performance of TBT can be comparable to that of GBT since the number of slots

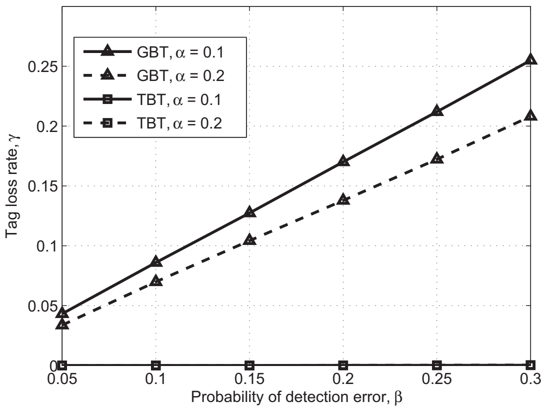

Fig. 6. The tag loss rate γ w.r.t. β, given $\alpha = 0.1$ (0.2)

with one tag's transmission is significantly reduced and thus, the effect of the D.E. on algorithms is negligible. On the other hand, TBT takes more slots to detect a tag when β becomes larger since more tags are hidden by the D.E.. We also observe that β does not affect the performance of GBT since the D.E. is not considered in this algorithm.

The tag loss rate of all methods with respect to different values of β is shown in Fig. 6, given $\alpha = 0.1\,(0.2)$. As β increases, more hidden tags are ignored in GBT, and thus, the loss rate increases. On the other hand, TBT keeps an outstanding tag loss rate regardless of values of β and α, which approximates 0%. This result validates our analysis, where we explained that, the maximum number of missing tags after the whole TBT identification process is 1. The probability of losing this last tag with $EC = 1$ is also less than β^2.

VI. CONCLUSIONS

In this paper, we studied Binary Tree-based anti-collision schemes in RFID systems and proposed TBT algorithm to cope with both the C.E. and D.E. phenomena. The whole TBT identification process was divided into multiple Binary Tree cycles, where tags hidden by the phenomena in a cycle re-transmitted in the first slot of the next cycle. The performance of the TBT was theoretically analyzed via two parameters which were the average number of slots for a successful detection of a tag and the tag loss rate. Computer simulations were also performed to validate our analysis. The results showed that the tag loss rate of TBT was much smaller than that of the conventional GBT, and approximated 0% regardless of values of α and β. The average number of slots for a successful detection of a tag of the proposed method, although was higher than that of GBT since more slots had to be used to deal with the D.E., they could be comparable when α was large enough. In future works, we intend to study the Query tree-based anti-collision algorithms under the C.E. and D.E..

REFERENCES

[1] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification, Wiley & Sons, 2003.

[2] R. Angeles, “RFID Technologies: Supply-Chain Applications and Implementations Issues,” *Information Systems Management*, vol. 22, no. 1, pp. 51-65, 2005.

[3] D. Molnar and D. Wagner, “Privacy and Security in Library RFID: Issues, Practices, and Architectures,” *Proceeding of the 11th ACM conference on Computer and Communication Security*, pp. 210-219, 2004.

[4] M. Bolíc, D. Simplot-Ryl and I. Stojmenovíc, RFID Systems: Research Trends and Challenges, Wiley & Sons, 2010.

[5] L. G. Roberts, “ALOHA Packet System with and without Slots and Capture,” *ACM SIGCOMM Computer Communication Review*, vol. 5, pp. 28-42, 1975.

[6] W-T Chen, “An Accurate Tag Estimate Method for Improving the Performance of an RFID Anticollision Algorithm Based on Dynamic Frame Length ALOHA,” *IEEE Transaction on Automation Science and Engineering*, vol. 6, no. 1, Jan., 2009.

[7] L. Zhu and T.-S. P. Yum, “The Optimal Reading Strategy for EPC Gen-2 RFID Anti-Collision Systems,” *IEEE Transactions on Communications*, vol. 58, no. 9, pp. 2725-2733, Sep., 2010.

[8] Y. Maguire and R. Pappu, “An optimal Q-Algorithm for the ISO 18000-6C RFID Protocol,” *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 1, pp. 16-24, Jan., 2009.

[9] L. Zhu and T.-S. P. Yum, “A Critical Survey and Analysis of RFID Anti-Collision Mechanisms,” *IEEE Communications Magazine*, vol. 49, no. 5, pp. 214-221, 2011.

[10] Philips Semiconductors, “I-CODE Smart Label RFID Tags,” http://www.semiconductors.philips.com/acrobat_download/other /identification/SL092030.pdf.

[11] “EPCglobal radio frequency identity protocols class-1 generation-2 uhf rfid protocol for communications at 860mhz-960mhz version 1.2.0,” Tech. Rep., 2008.

[12] D. Hush and C. Wood, “Analysis of Tree Algorithm for RFID Arbitration,” *Proceedings of the IEEE International Symposium on Information Theory*, pp. 107, Aug., 1998.

[13] M. Jacomet, A. Ehrsam and U. Gehrig, “Contactless Identification Device with Anticollision Algorihm,” *Proceedings of IEEE Conference on Circuits, System, Computers and Communications*, pp. 269-273, July, 1999.

[14] J. Choi and W. Lee, “Comparative Evaluation of Probabilistic and Deterministic Tag Anti-collision Protocols for RFID Networks,” *Proceedings of the EUCWorkshops*, pp. 538-549, 2007.

[15] M. A. Bonuccelli, F. Lonetti and F. Martelli, “Instant Collision Resolution for Tag Identification in RFID Networks,” *Elsevier, Ad Hoc Networks*, vol. 5, no. 8, pp. 1220- 1232, 2007.

[16] C. He and Z. J. Wang, “Closed-Form BER Analysis of Non-Coherent FSK in MISO Double Rayleigh Fading/RFID Channel,” *IEEE Communications Letters*, vol. 15, no. 8, pp. 848-850, Aug., 2011.

[17] W.-K. Sze, W.-C. Lau and O.-C. Yue, “Fast RFID Counting under Unreliable Radio Channel,” *Proceedings of the IEEE Internation Conference in Communications*, pp. 1-5, June, 2009.

[18] R. Jacobsen, K. F. Nielsen, P. Popovski and T. Larsen, Reliable Identification of RFID Tags Using Multiple Independent Reader Sessions, *Proceedings of the IEEE International Conference on RFID*, pp. 64-71, Apr., 2009.

[19] B.-D., Victoria M. and V.-A., Javier, “Analysis of the Identification Process in Active RFID Systems with Capture Effect,” *Proceedings of European Workshop on Smart Objects: Systems, Technologies and Applications*, pp. 1-6, 2010.

[20] W. J. Shin and J. G. Kim, “A Capture-Aware Access Control Method for Enhanced RFID Anti-collision performance,” *IEEE Communications Letters*, vol. 13, no. 5, pp. 354-356, May, 2009.

[21] B. Li and J. Wang, “Efficient Anti-Collision Algorithm Utilizing the Capture Effect for ISO 18000-6C RFID Protocol,” *IEEE Communications Letters*, vol. 15, no. 3, pp. 352-354, 2011.

[22] V. K. Y. Wu and R. H. Campbell, “Using Generalized Query Tree to cope with the Capture Effect in RFID Singulation,” *Proceedings of Consumer Communications and Networking Conference CCNC*, pp. 1-5, 2009.

[23] Y.-C. Lai and L.-Y. Hsiao, “General Binary Tree Protocol for Coping with the Capture Effect in RFID Tag Identification,” *IEEE Communications Letters*, vol. 14, no. 3, pp. 208-210, 2010.

[24] P. Popovski, K. Fyhn, R. Jacobsen, and T. Larsen, Robust Statistical Methods for Detection of Missing RFID Tags, *IEEE Wireless Communications*, vol. 18, no. 4, pp. 74-80, 2011.

[25] K. Fyhn, R. M. Jacobsen, P. Popovski and T. Larsen, Fast Capture-Recapture Approach for Mitigating the Problem of Missing RFID Tags, *IEEE Transactions on Mobile Computing*, vol. 11, no. 3, pp. 518-528, 2012.

[26] C. T. Nguyen, K. Hayashi, M. Kaneko and H. Sakai, “RFID Tag Cardinality Estimation under Capture Effect and Detection Errors,” *Proceedings of the Asia Pacific Wireless Communications Symposium*, Kyoto, Aug., 2012.

[27] C. T. Nguyen, K. Hayashi, M. Kaneko and H. Sakai, “Maximum Likelihood Approach for RFID Tag Cardinality Estimation under Capture Effect and Detection Errors,” *IEICE transactions on communications*, vol. E96B, no. 5, pp. 1122-1129, 2013.

[28] M. Islam, S. A. Samad, M. A. Hannan, and A. Hussain, “Software defined radio for RFID signal in Rayleigh fading channel,” *Proc. IEEE Region 10 Conference TENCON*, pp. 13681372, Oct. 2010.