Effective Tag Identification Algorithm for Dynamic Radio Frequency Identification Systems

Cheng-Huang Chang, Chen-Chuan Wu, Chiu-Kuo Liang

Department of Computer Science and Information Engineering, Chung Hua University, Taiwan
{e10602008, e10102006, ckliang}@chu.edu.tw

Abstract

In radio frequency identification (RFID) systems, the rapid identification of all tags located within the range of a reader is a major research topic. For rapid identification, the development of an effective anticollision protocol between the reader and tags is essential. Numerous anticollision protocols have been proposed, but these anticollision algorithms tend to be used in a static environment where the tags available within the range of a reader do not change during an identification process. By contrast, in the dynamic environment of RFID systems, such as in inventory management, some identified tags are removed and new tags are included. The reader in such an environment must regularly reidentify the tags. The previously proposed dynamic identification method cannot fully utilize the results from a previous identification process, and that results in collisions and idles occurring again in the new identification process. Therefore, such an identification process is inefficient. In this study, an efficient protocol termed the leaf node bit collision detection–based query tree (LBQT) algorithm is proposed. The proposed LBQT algorithm can reduce unnecessary queries for unchanged tags and rapidly identify the new tags. Experimental results indicate that the proposed algorithm outperforms previous protocols without depending on various densities of tag distributions.

Keywords: RFID systems, Tag anticollision algorithms, Dynamic identification

1 Introduction

Radio frequency identification (RFID) is a modern technology that is widely used in industrial applications, such as object and people tracking, supply chain management, vehicle positioning [1-3], and inventory management [4]. Traditional identification systems, such as barcodes, are inefficient at automatic identification and data collection because of their read rate, visibility, and contact limitations. By contrast, RFID systems can provide rapid and reliable communication without establishing physical visibility or contact between readers and tags. Because of these advantageous features, the current RFID technology transcends the object identification function and is being used for localization [5-6] and sensing applications [7].

One of the areas of research in this field is the reduction of identification processing time for a particular number of tags within the recognition range of an RFID reader. To achieve rapid tag identification, anticollision protocols are required. A collision may occur when multiple tags simultaneously respond to the inquiry of the reader. Therefore, anticollision protocols generally aim to reduce collisions during the tag identification process. In general, collisions can be categorized into two types: reader collisions and tag collisions. When two or more neighboring readers simultaneously inquire about a tag, reader collisions occur. Thus, the tag cannot accurately provide its unique identification code (ID) to the inquiring readers. The reader collision can be easily eliminated by detecting the collisions and communicating with other readers. Tag collisions occur when more than one tag concurrently respond to a reader, which causes the reader to identify no tag. In RFID systems with low-cost passive tags, a tag can only respond to the inquiry of readers. Therefore, tag anticollision protocols are essential for the efficient identification of tag IDs in RFID systems.

The relevant literature provides numerous anticollision research results. These anticollision protocols can be classified into two primary categories, namely an Aloha-based anticollision scheme [8-10] and a tree-based scheme [11-19]. Although the Aloha-based protocols can reduce the probability of tag collisions, they exhibit a tag starvation problem whereby a particular tag may not be identified for a long time. By contrast, tree-based protocols, such as the binary tree splitting protocol [18] and query tree (QT) algorithm [11], do not cause the starvation problem, but they have a relatively long identification delay. Therefore, we consider tree-based protocols and seek to reduce the identification delay.

In this study to develop a practical RFID system, an efficient tag identification protocol was created for the

*Corresponding Author: Chiu-Kuo Liang; E-mail: ckliang@chu.edu.tw
DOI: 10.3966/160792642019122007019

dynamic environment of a warehouse inventory management system. In the target dynamic environment, all items should be tracked by a mobile reader (Figure 1). The reader can identify tags within its communication range by moving in a straight direction. The reader identification process comprises the following two phases: (1) an initial static phase and (2) a dynamic phase. The initial static phase is the first phase of the process when the reader is restarted and no tags have been identified. The dynamic phase indicates the reader identification process of identifying the changed tags in the range of the reader. In the dynamic environment, the reader identification process can be considered the execution of the initial static identification process followed by several dynamic identification processes until no tag is changed.

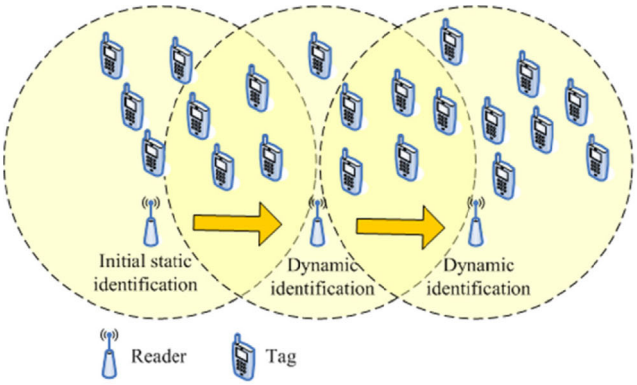


Figure 1. Mobile reader for tag identification

For tree-based anticollision schemes, most studies have focused on improving tag identification performance in a static environment in which both the reader and tags are static. An adaptive memoryless QT (MQT) [12] first considers mobile tag identification. In [17], an enhanced bit collision detection–based query tree (EBQT) was proposed for reducing identification delay and communication overhead in mobile identification scenarios. In this study, a leaf node–based bit collision detection query tree (LBQT) algorithm was proposed for improving the performance of tag identification. To evaluate the performance of the proposed technique, the proposed LBQT scheme was implemented with the previously proposed method (i.e., the EBQT protocol). The experimental results indicate that the proposed technique achieves significantly improved performance in most circumstances.

The remainder of this paper is organized as follows: Section 2 discusses relevant literature. In Section 3, the proposed algorithm (i.e., the LBQT protocol) is presented. Performance comparisons and an analysis of the proposed technique are provided in Section 4. Finally, Section 5 presents the conclusions.

2 Related Work

The binary QT protocol [11] is considered a milestone in the development of binary tree–based algorithms for passive tags. Although the QT protocol ensures reliable performance, it requires a long time to complete the identification process. To reduce the processing time, several enhanced QT protocols have been proposed, namely the adaptive memoryless QT (MQT) [12], bit collision detection-based QT (BQT) [13], anticipative inquiry scheme (AIS) [14], prefix-randomized QT (PRQT) [15], hybrid QT (HQT) [16], and enhanced BQT (EBQT) [17]. Of these protocols, only MQT and EBQT perform identification in dynamic environments in which some tags are eliminated from the recognition region of the reader and some new tags are moved into the region of the reader. In the following, only the MQT and EBQT algorithms are presented and used as benchmarks for comparison with our work.

2.1 Adaptive Memoryless QT Protocol

Fundamentally, the MQT protocol for accelerated identification uses information from previous processes. In the initial identification phase, the MQT protocol uses the QT protocol to identify all tags, which means that the reader transmits a query and tags respond with their IDs. The query includes a bit string of length k. The tag responds if the first k-bit string of its ID matches the query string of the reader. After the first identification process, the MQT protocol places all leaf nodes in the query tree into a special queue termed the candidate queue (CQ). The leaf nodes in the query tree are the successfully identified nodes or idle nodes.

In the subsequent dynamic identification phase, the reader updates the query queue using CQ and dequeues until the query queue is empty. In the dynamic recognition process, the reader places the leaf nodes into CQ for the next process.

Table 1 illustrates the operation of the MQT for five tags with an ID length of 4 bits in the initial static identification phase. The tag IDs are “0000,” “0001,” “0100,” “0101,” and “0110,” respectively. Initially, the reader sends an empty query string to tags and all tags respond simultaneously, which results a collision. The reader adds two query strings, namely “0” and “1”, into query queue. Then, the reader fetches the query string “0” from the query queue, broadcasts it to all tags, and waits for the tag responses. At this stage, more than one tag responds simultaneously, resulting in collisions. The reader extends query string attached “0” or “1” and adds into the query queue. Next, the reader sends query string “1” to all tags and no tags respond. The reader adds the query string “1” into CQ for next dynamic identification process. The reader continues the process until the query queue is empty.

Table 1. Detailed steps of the MQT protocol in the initial static phase

Steps	Prefix	Results	Query Queue	Add to CQ
1	Empty	Collision	0, 1	
2	0	Collision	1, 00, 01	
3	1	Idle	00, 01	1
4	00	Collision	01, 000, 001	
5	01	Collision	000, 001, 010,011	
6	000	Collision	001, 010, 011, 0000, 0001	
7	001	Idle	010, 011, 0000, 0001	001
8	010	Collision	011, 0000, 0001, 0100, 0101	
9	011	Identified	0000, 0001, 0100,0101	011
10	0000	Identified	0001, 0100,0101	0000
11	0001	Identified	0100, 0101	0001
12	0100	Identified	0101	0100
13	0101	Identified		0101

Next suppose that after the initial identification process, two tags, “0000” and “0001,” leave and two tags, “0010” and “1110,” arrive. Table 2 presents the detailed steps of the identification process in the dynamic phase. In this phase, the reader updates the query queue by CQ which contains “1”, “001”, “011”, “0000”, “0001”, “0100”, and “0101” query strings and dequeues until the query queue is empty. The identification process is the same as in the initial phase and it takes only seven steps to identify five tags.

Table 2. Detailed steps of the MQT protocol in the dynamic phase

Steps	Prefix	Results	Query Queue	Add to CQ
INIT			1,001,011,0000,0001, 0100,0101	
1	1	Identified	001,011,0000,0001, 0100,0101	1
2	001	Identified	011,0000,0001,0100, 0101	001
3	011	Identified	0000,0001,0100,0101	011
4	0000	Idle	0001,0100,0101	0000
5	0001	Idle	0100, 0101	0001
6	0100	Identified	0101	0100
7	0101	Identified		0101

2.2 Enhanced Bit Collision Detection-based Query Tree Protocol

Although the MQT protocol utilizes information in the previous process, its performance significantly depends on the difference between two consecutive processes. A large difference may not reduce the time delay for the identification process. The EBQT protocol uses the number of identified tags in the previous process to shorten the future process under a condition where the number of tags does not fluctuate for each process. A level jumping strategy is used. If an approximate number of tags is known to a reader, prefixes in a query queue can be expanded to an appropriate level for reducing collisions. For example, if five tags are identified in the previous process, the prefix in the query queue may start from the second level instead of the root level (Figure 2).

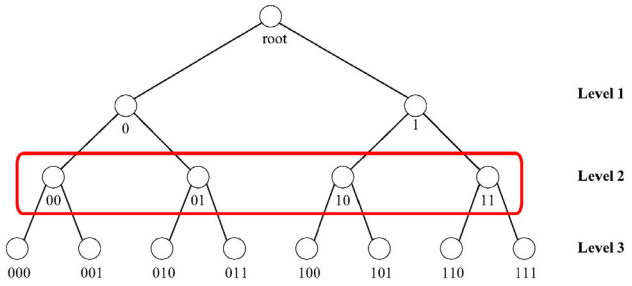


Figure 2. Level jumping in EBQT protocol

The detailed process of the EBQT protocol is as follows: In the initial identification phase, the EBQT protocol uses BQT protocol [13] to identify all tags. In the BQT protocol, the reader initializes a query string by using “**...*” with the length of tag IDs and adds the string into an empty queue Q. Each “*” is one-bit wild masking for the values “0” or “1.” At the initial stage, the reader fetches a query string from Q, broadcasts it to all tags, and each tag sends back its ID to the reader. All tags respond simultaneously, resulting in collisions. When collisions occur, the reader detects individual bit collisions through a hybrid Manchester coding scheme [13] and updates the query string. At a particular bit position, if all responses have values of “0” or “1, no collision occurs at this bit and the bit of the query string can be updated using the corresponding values “0” or “1.” If a collision occurs at a particular bit position, the bit at this position string remains “*”. After the query string is updated, the reader considers the following three possible situations: (1) Multiple “*” exist in the updated query string. In this situation, the reader replaces the first “*” with “0” and “1” to create two new query strings and adds them into queue Q. (2) A single “*” is present in the updated query string. The reader replaces this “*” with “0” and “1” and marks them as two successfully identified tags without extra queries. (3) No “*” exists in the updated query string. Here, the reader recognizes the ID and considers the tag a successfully identified tag. The reader then fetches a subsequent query string from Q and repeats the identification process until queue Q is empty. In the BQT protocol, the reader sends a query string with a length of tag IDs to all tags. Each bit position of the query string may include “0”, “1”, or “*” wild mask. These tags match the bit position of values “0” and “1” and they must respond with their respective values for the position corresponding to the wild mask “*”. For example, if the reader sends a query string “0*1**” to all tags, the tags with the first and third bits of “0” and “1” respond with their respective values of the second, fourth, and fifth bits to the reader.

In the EBQT protocol, an ID counter (I_c) is established to count the number of identified tags. After the initial identification, the reader begins to generate prefixes for the next processes. The prefixes can be generated according to the starting level, which is computed by the following equation:

$$L = \lfloor \log_2 I_c \rfloor \quad \textbf{(1)}$$

The reader generates prefixes from "$\underbrace{00...0}_{L}$*...*" to "$\underbrace{11...1}_{L}$*...*" and adds them into the query queue. I_c is reset to 0 and the BQT protocol is performed for the subsequent process.

Although the EBQT protocol uses the number of identified tags in the previous process to reduce the time required for the identification process by implementing the appropriate level in the subsequent process, some idle nodes are requeried, particularly for imbalanced tag ID distribution. Therefore, the requeried idle nodes are wasted. Table 3 illustrates the detailed operation of the EBQT algorithm for identifying five tags in the initial static phase using the same example as in the previous section.

Table 3. Detailed steps of EBQT protocol in initial static phase

Steps	Prefix	Update string	Results	Add to Q
INIT				****
1	****	0***	Collision	00**, 01**
2	00**	000*	Identified	01**
3	01**	01**	Collision	010*, 011*
4	010*	010*	Identified	011*
5	011*	0110	Identified	

After the initial identification, the reader begins to generate prefixes for next dynamic phase according to the ID counter I_c. In this example, the starting level $L = \lfloor \log_2 5 \rfloor = 2$. The reader generates prefixes "00**", "01**", "10**", and "11**" and adds them into the query queue. Table 4 illustrates the detailed steps of the EBQT algorithm in the dynamic phase.

Table 4. Detailed steps of EBQT protocol in dynamic phase

Steps	Prefix	Update string	Results	Add to Q
INIT				00**, 01**, 10**, 11**
1	00**	0010	Identified	01**, 10**, 11**
2	01**	01**	Collision	10**, 11**, 010*, 011*
3	10**		Idle	11**, 010*, 011*
4	11**	1100	Identified	010*, 011*
5	010*	010*	Identified	011*
6	011*	0110	Identified	

3 Proposed LBQT Protocol

In the MQT protocol, the identified or idle nodes are put into the CQ for the subsequent process. However, with the increasing tag change rate, the previously identified tags may not exist. This finding indicates that the identified tags become idle nodes and are read into the CQ. Therefore, the MQT protocol uses several queries for the disappeared tags. Although the EBQT protocol utilizes the number of identified tags from the previous identification process to estimate the appropriate level to initiate a query process for the subsequent dynamic identification, it does not consider tag ID distribution. Therefore, for imbalanced distribution, numerous idle nodes are probably present in the query process. Figure 3 illustrates the inefficiency of the EBQT protocol by showing the eight identified tags that are available in the previous identification process. Then, according to the EBQT protocol, the nodes in level 3 are included in the query queue for the subsequent dynamic identification process. However, if no new arrival tags are available, four possible idle nodes are required to pose a query during the subsequent dynamic phase.

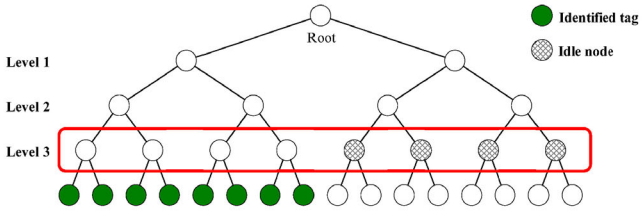


Figure 3. Illustration of imbalanced case of EBQT protocol

In this study, an efficient protocol was proposed to reduce the identification process in the dynamic phase. The proposed protocol is termed the LBQT protocol. The proposed LBQT protocol comprises the following three parts:

(1) The tag identification process in the EBQT protocol is more efficient than that in the MQT protocol because the EBQT protocol uses the BQT protocol and not the QT protocol used in the MQT protocol. The tag identification procedure in the BQT protocol skips all possible idle queries during the identification process, which is advantageous. In this study, the BQT and QT protocols were combined and an improved BQT protocol was developed to more efficiently identify tags than using the BQT protocol alone.

(2) The BQT protocol can efficiently identify tags because it can avoid idle nodes in the identification process. However, it is not useful in a dynamic environment because it may have some new arrival tags in the previous idle nodes. Therefore, in the proposed LBQT protocol, the possible idle nodes must be identified, and these nodes should be put into the

CQ for the subsequent dynamic phase. Therefore, an idle node recovery procedure was developed to trace the skipped idle nodes and add them into a particular queue for the subsequent identification process.

(3) To reduce the identification time for newly arrived tags in the dynamic phase, a level jumping technique was developed to eliminate those possibly unnecessary query nodes.

3.1 Improved BQT Protocol

In the BQT protocol, the reader sends a query bit string with a length of tag IDs in which each bit consists of "0", "1", or "*". The "*" bit indicates the unknown bit for responding to tags. Tags with specific bits that match the corresponding bits in the query bit string of the reader respond with their values for the unknown bits. To reduce the query time in the proposed improved BQT protocol, the consecutive tailing bits of "*" values are replaced with an empty string. For example, in the first round, the reader must send a masking code "**...*" to tags in the BQT protocol. By contrast, in the improved BQT protocol, the reader sends an empty string to tags. In the BQT protocol, the reader must send a masking code "01*1**" to tags, whereas in the improved BQT protocol, the reader sends "01*1" to tags. Tags matching the query string send a response comprising unknown bits and remaining bits.

3.2 Idle Node Recovery Procedure

In the proposed LBQT protocol, when the reader sends a query string to tags, the reader masks the collided bits using the masking code "*". The bits before the first masking code "*" in the returned bits represent the status of tag distributions. For example, if the returned bits are identified as "0***", the reader can realize that all tags have the same first bit "0" in their IDs. In addition, the reader can recognize that no tags have "1" as their first bit. Thus, if the reader sends a query code "1", no tag responds. Therefore, an idle node is found and added into the CQ for the subsequent process. Figure 4 presents the idle node recovery procedure. Figure 4 indicates that a reader sends "0***" to tags and receives a returned code "001*". According to the procedures, two idle nodes "01" and "000" are recovered and added into the CQ.

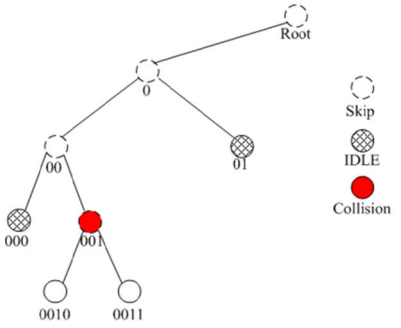


Figure 4. Illustration of idle node recovery procedure

3.3 Enhanced Level Jumping Technique

In the MQT protocol, the identified or idle prefix strings are added into the CQ for the subsequent identification process. However, the query time of the prefix strings in the CQ may be wasted when several tags are eliminated in the subsequent identification process. To improve the query time in the subsequent process, two queues, namely CQ and idle queue (IQ) were used instead of using only the CQ in the MQT process. In the proposed LBQT protocol, the purpose of the CQ is similar to that in the MQT protocol, where it is used for storing the identified prefix strings during the identification process. However, the idle prefix stings are not added into the CQ. By contrast, they are stored in the IQ. In the subsequent identification process, the prefix strings in the CQ are first queried and followed by queries of the idle strings in the IQ. The queries in the CQ are used for the previous tags, whereas the queries in the IQ are for the subsequent new tags. Therefore, if the number of newly arriving tags is known, the query time for the idle strings in the IQ is improved by jumping to the appropriate level for scanning the prefixes. Let T_c and F_c denote the number of tags identified in the previous identification process and in the current identification process, respectively. An idle threshold level (L_{idle}) is obtained using the following equation:

$$L_{idle} = \lfloor \log_2(T_c - F_c) \rfloor \quad \textbf{(2)}$$

The reader then eliminates the nodes from the IQ with a bit length less than L_{idle} and begins the subsequent process by updating the query queue with the modified IQ and dequeues until the query queue is empty.

To facilitate the understanding of the proposed algorithm, the same example from the previous section is used, and the detailed operation is explained as follows. Table 5 illustrates the detailed operation of communication between the reader and tags of the example in the initial static phase. As evident in Table 5, the proposed LBQT puts the identified nodes and idle nodes into the CQ and IQ, respectively, for the subsequent process. For example, initially, the reader sends an empty prefix to tags, and all tags respond to the query of the reader at the same time. After receiving the responses, the reader understands the distribution of tag ID as "0***", which indicates that the first bit in the ID of any tag is not "1". Therefore, "1" is an idle node in the query tree and is put into the IQ for the subsequent process.

After the initial static phase, the CQ comprises 3 prefixes "00", "010", and "011", whereas the IQ comprises 2 prefixes, "1" and "001". Moreover, the reader understands that five identified tags are present in the previous process (i.e., $T_c = 5$). Four tags, namely "0000", "0001", "0100", and "0101", are removed and four tags, namely "1000", "1010", "1011", and "1110"

Table 5. Detailed steps of LBQT protocol in initial static phase

Step	Prefix	Update string	Results	Query Queue	Add to CQ	Add to IQ
1	Empty	0***	Collision	00, 01		1
2	00	000*	Identified	01	00	001
3	01	01**	Collision	010, 011		
4	010	010*	Identified	011	010	
5	011	0110	Identified		011	

are moved in before the subsequent identification process.

In the subsequent identification process, the reader first puts all prefixes in the CQ into the query queue. In the example, the query queue has 3 prefixes, "00", "010", and "011", in the beginning of the subsequent identification process. After the query process, the reader identifies all tags that can be identified by the prefixes in the CQ. In the example, only one tag can be identified for the prefixes in the CQ: $F_c = 1$. The idle threshold level is represented as $L_{idle} = \lfloor \log_2(5 - 1) \rfloor = 2$. This indicates that the prefixes may need to be adjusted in the IQ to improve the identification process for new arrival tags. In this example, the IQ comprises "1" and "001" strings. By using the proposed enhanced level jumping technique, those prefixes with a bit length less than L_{idle} are extended to the prefixes with a bit length of L_{idle}. Therefore, the modified IQ comprises "10", "11", and "001" prefixes. The reader puts all prefixes in the IQ into the query queue and finishes the query until the query queue is empty.

Table 6 shows the detailed operation of the proposed LBQT protocol in the dynamic phase, where two tags are removed and a new tag arrives.

Table 6. Detailed steps of LBQT protocol in dynamic phase

Step	Prefix	Update string	Results	Query Queue	Add to CQ	Add to IQ
INIT				00, 010,011		
1	00		Idle	010, 011		00
2	010		Idle	011		010
3	011	0110	Identified	10,11,001	011	
4	10	10**	Collision	11,001,100, 101		
5	11	1110	Identified	001,100,101	11	
6	001		Idle	100,101		001
7	100	1000	Identified	101	100	
8	101	101*	Identified		101	

4 Performance Evaluation

To evaluate the performance of the proposed technique, the LBQT protocol was implemented with the EBQT protocol. A set of simulation experiments was conducted for the proposed algorithms. All experiments were performed on a computer equipped with a 3.0-GHz central processing unit and 4-GB memory in C# on the .NET platform. Each experiment was repeated 20 times, and the recorded data were averaged for the runs into the final results. In the experiments, the IDs of all tags were 16 bits long and the number of tags was set from $2^{16} \times 10\% = 6554$ to $2^{16} \times 50\% = 32768$. Two different distributions of tag IDs and balanced and imbalanced distributions were considered. Balanced distribution indicates that all tags were uniformly distributed in the left subtree and right subtree of the query tree. By contrast, imbalanced distribution indicates that the tags were uniformly distributed only in one of the left subtree or right subtree and no tag was present in the other subtree. Moreover, in our experiments, the tag changing rate was assumed to be fixed for both moving-out and moving-in tags between two consecutive processes. This means that the same number of identified tags moved out as new tags moved in. For balanced distribution, the moved-out and moved-in tags were randomly selected. For imbalanced distribution, the moved-out tags were randomly selected from one subtree and moved-in tags were randomly generated in another subtree.

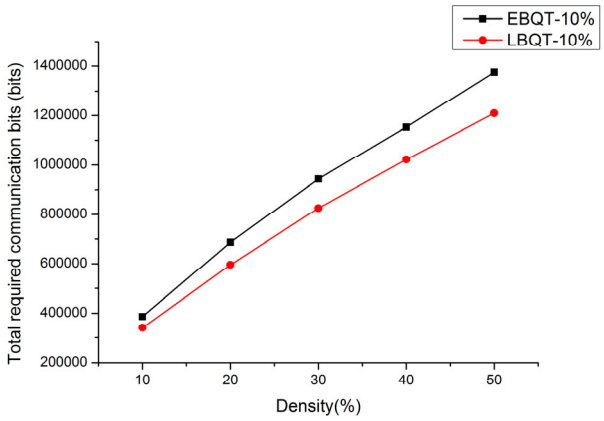
4.1 Balanced Distribution

In this experiment, the performance of the proposed LBQT protocol was evaluated when tag distribution is balanced. All tags were randomly generated in a uniform distribution to both left and right subtrees.

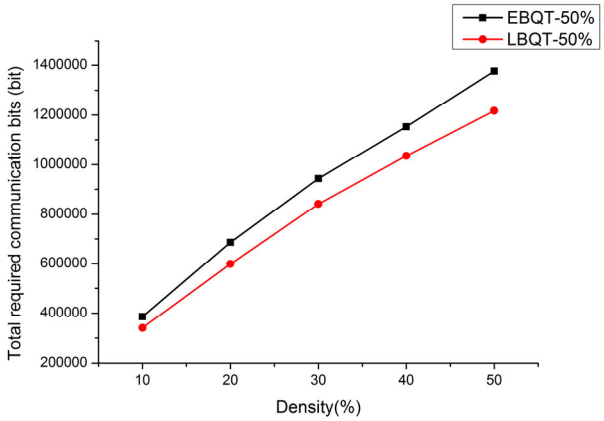
Figure 5 shows the time comparison results of the experiment examining the effect of the number of tags on the total bits required for communication between the reader and tags to complete tag identification in the EBQT and LBQT protocols. The experiment was performed by executing an initial static phase and a dynamic phase with tag changing rates of 10% and 50%, respectively, for each protocol.

Figure 5 indicates that with an increasing number of tags, each protocol proportionally increased because of the increasing number of collisions. However, the proposed LBQT protocol required fewer bits or less time to complete the identification process. Furthermore, the total required bits for both protocols at a tag changing rate of 50% was almost the same as that at a tag changing rate of 10%. This indicates that both protocols can effectively complete the identification process regardless of the tag changing rate in balanced distribution.

Figure 6 shows the time comparison results of the experiment that examined the effect of the changing rate of tags on the total required communication bits for tag identification in the EBQT and LBQT protocols. This experiment was performed by executing an initial static phase and a dynamic phase for each protocol with tag densities of 25% and 50%.

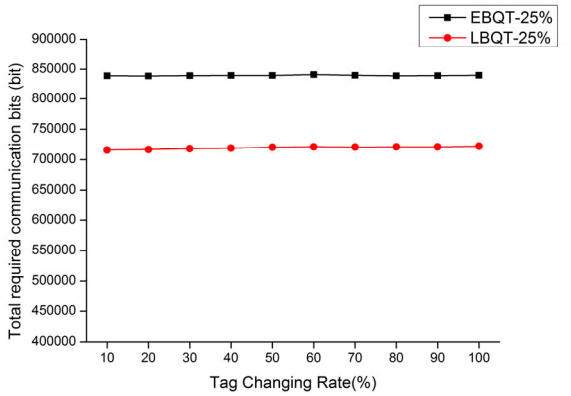


(a) Tag changing rate = 10%

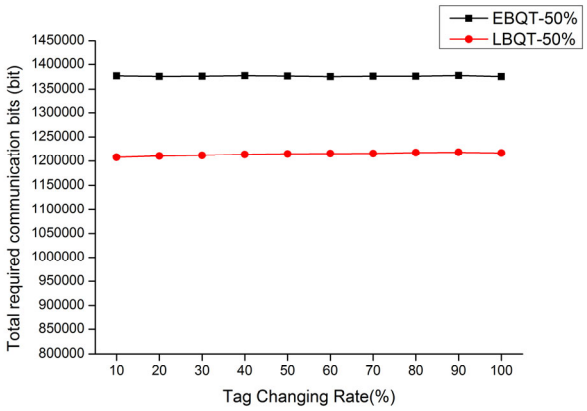


(b) Tag changing rate = 50%

Figure 5. Total required communication bits for completing a static phase and a dynamic phase with different tag changing rates at different densities



(a) Tag density = 25%



(b) Tag density = 50%

Figure 6. Total required bits for completing a static phase and a dynamic phase at different changing rates with tag densities of 25% and 50%, respectively

Figure 6 indicates that both protocols exhibited acceptable performance at the increasing tag changing rates. This indicates that both protocols can efficiently identify the newly arrived tags without spending more time. However, the proposed LBQT protocol outperformed the EBQT protocol in terms of total required communication bits to complete the identification process.

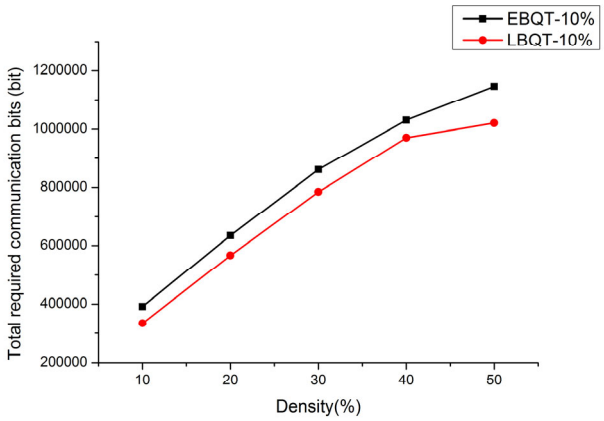
4.2 Imbalanced Distribution

In this experiment, the performance of the proposed LBQT protocol was evaluated for imbalanced tag distribution. Initially, all tags were randomly generated with uniform distribution in one subtree of the query tree. In the dynamic phase, all the newly arrived tags were generated in the other subtree.

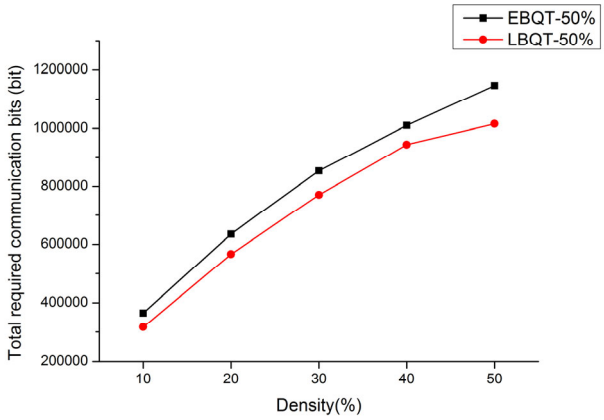
Figure 7 shows the time comparison results of the experiment that examined the effect of the number of tags on the required communication for the tag identification in the EBQT and LBQT protocols. The experiment was performed by executing an initial static phase and a dynamic phase at tag changing rates of 10% and 50%, respectively, for each protocol.

Figure 7 indicates that with an increasing number of tags, each protocol increased proportionally. The proposed LBQT protocol outperformed the EBQT protocol for tag changing rates of 10% and 50%. With an increasing tag changing rate, the LBQT protocol was superior to the EBQT protocol. For example, the performance of the LBQT protocol was, on average, 10.2% and 28.3% superior to that of the EBQT protocol when the tag changing rates were 10% and 50%, respectively. This result indicates that the LBQT protocol can effectively eliminate the unnecessary query nodes for a large number of newly arrived tags in imbalanced distribution.

Figure 8 shows the time comparison results of the experiment that examined the effect that the changing rate of tags had on the total required communication bits for tag identification in the EBQT and LBQT protocols under imbalanced distribution. This experiment was performed by executing an initial static phase and a dynamic phase for each protocol with tag densities of 25% and 50%.

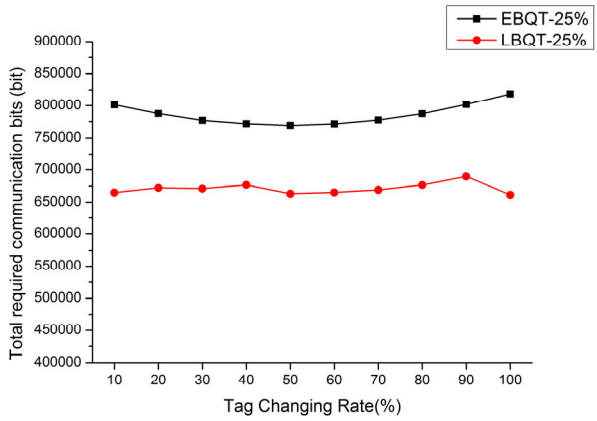


(a) Tag changing rate = 10%

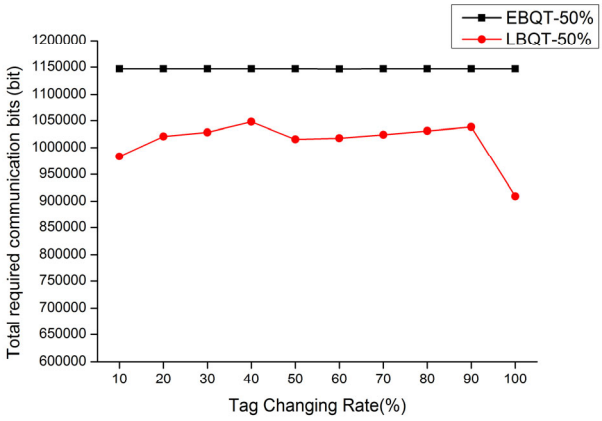


(b) Tag changing rate = 50%

Figure 7. Total required communication bits for completing a static phase and a dynamic phase at different tag changing rates with different densities of imbalance distribution



(a) Tag density = 25%



(b) Tag density = 50%

Figure 8. Total required bits for completing a static phase and a dynamic phase at different changing rates with tag densities of 25% and 50%, respectively, under imbalanced distribution

Figure 8 indicates that the proposed LBQT protocol used more queries for the newly arrived tags when the tag changing rate increased from 10% to 40%. However, it reduced more unnecessary query nodes when the tag changing rate increased from 40% to 100%. This result indicates that when the tag changing rate is small, the level jumping technique cannot acceptably eliminate the unnecessary query nodes because only a slight difference is evident between the two consecutive identification phases. However, with an increasing tag changing rate, the difference continually increased. Therefore, a large amount of both unnecessary query nodes and the number of transmission bits between reader and tags can be reduced when level jumping technique is applied. In Figure 8(b), when tag density is 50%, there are $2^{16} \times 50\% = 32768$ tags distributed among the whole left subtree or right subtree of the binary tree in the initial phase. When the tag changing rate increased to 100%, after the initial identification process, all tags in one subtree are removed and there are 32768 newly arrived tags distributed in other subtree. In this case, $T_c = 32768$, $F_c = 0$ and the idle threshold level $L_{idle} = \lfloor \log_2(32768 - 0) \rfloor = 15$. This means that the level jumping technique is applied and the tag identification process will jump from level 14 to level 15 when the tag changing rate increased from 90% to 100%. Therefore, all of the query nodes in level 14 can be omitted and the number of transmission bits between reader and tags can be greatly reduced. By contrast, with the increasing density, the performance of the EBQT protocol was nearly independent of the tag changing rate because it jumped to the same level for scanning regardless of the tag changing rate.

5 Conclusions

The development of a highly efficient tag

identification process in a dynamic RFID system is crucial and challenging. Many collisions may occur during the dynamic tag identification process because of many unknown new tag arrivals. An identification protocol such as EBQT can reduce the identification time, but there are still many idle cycles. In this study, an efficient protocol called the LBQT protocol was proposed, which can reduce the time delay in the dynamic identification phase more efficiently. The proposed LBQT protocol improves the BQT protocol in terms of the identification process and saves communication overhead, adapts the approach of adding leaf nodes into the CQ for the subsequent process, develops an idle node recovery procedure to restore the idle nodes into the IQ, and improves the level jumping technique on the nodes in the IQ by using the idle threshold level to reduce the identification process time for newly arrived tags. Therefore, the protocol outperforms the previous EBQT protocol. Experimental results indicated that the LBQT protocol is more efficient for the dynamic environment of an RFID system.

References

[1] C. C. Hsu, J. H. Chen, A Novel Sensor-Assisted RFID-based Indoor Tracking System for the Elderly Living Alone, *Sensors*, Vol. 11, No. 11, pp. 10094-10113, October, 2011.

[2] A. Shirehjini, A. Yassine, S. Shirmohammadi, Equipment Location in Hospitals Using RFID-based Positioning System, *IEEE Transactions on Information Technology in Biomedicine*, Vol. 16, No. 6, pp. 1058-1069, November, 2012.

[3] J. Wang, D. Ni, K. Li, RFID-based Vehicle Positioning and Its Applications in Connected Vehicles, *Sensors*, Vol. 14, No. 3, pp. 4225-4238, March, 2014.

[4] X. Zhu, S. K. Mukhopadhyay, H. Kuraya, A Review of RFID Technology and Its Managerial Applications in Different Industries, *Journal of Engineering and Technology Management*, Vol. 29, No. 1, pp. 152-167, January-March, 2012.

[5] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, E. Aboutanios, Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications, *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 2, pp. 1327-1346, November, 2016.

[6] P. K. Yoon, S. Zihajehzadeh, B. S. Kang, E. J. Park, Robust Biomechanical Model-based 3D Indoor Localization and Tracking Method Using UWB and IMU, *IEEE Sensors Journal*, Vol. 17, No. 4, pp. 1084-1096, February, 2017.

[7] C. Occhiuzzi, S. Caizzone, G. Marrocco, Passive UHF RFID Antennas for Sensing Applications: Principles, Methods, and Classifications, *IEEE Antennas and Propagation Magazine*, Vol. 55, No. 6, pp. 14-34, December, 2013.

[8] J. Park, M. Chung, T. J. Lee, Identification of RFID Tags in Framed-Slotted ALOHA with Robust Estimation and Binary Selection, *IEEE Communications Letters*, Vol. 11, No. 5, pp. 452-454, May, 2007.

[9] H. Wu, Y. Zeng, J. Feng, Y. Gu, Binary Tree Slotted ALOHA for Passive RFID Tag Anticollision, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 1, pp. 19-31, January, 2013.

[10] B. Zhen, M. Kobayashi, M. Shimizui, Framed Aloha for Multiple RFID Objects Identification, *IEICE Transactions on Communications*, Vol. E88-B, No. 3, pp. 991-999, March, 2005.

[11] C. Law, K. Lee, K. Y. Siu, Efficient Memoryless Protocol for Tag Identification, *The 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Boston, MA, USA, 2000, pp. 75-84.

[12] J. Myung, W. J. Lee, T. K. Shih, An Adaptive Memoryless Protocol for RFID Tag Collision Arbitration, *IEEE Transactions on Multimedia*, Vol. 8, No. 5, pp. 1096-1101, October, 2006.

[13] H. S. Gou, H. C. Jeong, Y. H. Yoo, A Bit Collision Detection based Query Tree Protocol for Anti-Collision in RFID System, *2010 IEEE 6th International Conference on Wireless and Mobile Computing, Networking and Communications*, Niagara Falls, ON, Canada, 2010, pp. 421-428.

[14] C. H. Hsu, B. Y. Chen, C. T. Yang, Anticipative Inquiry Scheme for Efficient RFID Tag Identification, *The IEEE 3rd Conference on Multimedia and Ubiquitous Engineering*, Qingdao, China, 2009, pp. 232-237.

[15] K. W. Chiang, C. Q. Hua, T. S. Peter, Prefix-Randomized Query Tree Protocol for RFID System, *2006 IEEE International Conference on Communications (ICC-06)*, Istanbul, Turkey, 2006, pp. 1653-1657.

[16] J. Ryu, H. Lee, Y. Seok, T. Kwon, Y. Cho, A Hybrid Query Tree Protocol for Tag Collision Arbitration in RFID systems, *2007 IEEE International Conference on Communications (ICC-07)*, Glasgow, UK, 2007, pp. 5981-5986.

[17] H. S. Gou, Y. H. Yoo, Bit Collision Detection Based Query Tree Protocol for Anti-Collision in RFID System, *International Journal of Innovative Computing, Information and Control*, Vol. 8, No. 5(A), pp. 3081-3102, May, 2012.

[18] J. Myung, W. Lee, J. Srivastava, Adaptive Binary Splitting for Efficient RFID Tag Anti-Collision, *IEEE Communications Letters*, Vol. 10, No. 3, pp. 144-146, March, 2006.

[19] C. K. Liang, Y. C. Chien, A Pre-Detection Based Anti-Collision Algorithm with Adjustable Slot Size Scheme for Tag Identification, *Sensors & Transducers*, Vol. 189, No. 6, pp. 61-70, June, 2015.

Biographies

Cheng-Huang Chang received the MS degree in computer science and information engineering from Chung Hua University in 2019. His research interests include wireless networks and RFID systems.

Chen-Chuan Wu received the MS degree in computer science and information engineering from Chung Hua University in 2014. His research interests include RFID systems, wireless networks, and sensor networks.

Chiu-Kuo Liang received his PhD in Computer Science from the National Tsing Hua University in Taiwan, R.O.C. He is currently an associate professor of the Department of Computer Science and Information Engineering at Chung Hua University. His research interests include wireless mobile computing, sensor networks, and parallel processing.