Sensors and Materials, Vol. 33, No. 8 (2021) 2549–2564
MYU Tokyo

S & M 2638

Hybrid Predetection Technique for Efficient Tag Identification in Radio-frequency Identification Systems

Kuo-Sen Hsu and Chiu-Kuo Liang*

Department of Computer Science and Information Engineering, Chung Hua University,
No. 707, WuFu Road, Section 2, Hsinchu 30012, Taiwan

(Received March 22, 2021; accepted June 10, 2021)

Keywords: anticollision algorithm, predetection technique, radio-frequency identification (RFID) systems

Radio-frequency identification (RFID) is a noncontact automatic identification technology that has been widely used in various modern industries. Due to the continuous improvement in sensing capabilities, energy autonomy, affordability, and usability, RFID is one of the key enabling technologies in realizing the Internet of Things (IoT). With their non-line-of-sight wireless transmission, wireless power, and sensing capability, lightweight RFID sensor tags can be connected to people or objects, which is crucial for future IoT applications such as healthcare, logistics, and manufacturing. The RFID approach with the simultaneous reading of multiple tag features can quickly identify connected "things" for further data communication and integration. Reducing the identification processing time for many tags located inside the interrogation range of the reader is essential. Therefore, an effective protocol must be developed to prevent collisions and reduce useless queries between the reader and tags to achieve rapid identification. Predetection tree-based algorithms, which are based on certain small predetection timeslots that tags can respond to, can eliminate unnecessary idle slots and prevent collision slots. In this paper, an efficient predetection-based anticollision algorithm has been proposed to achieve outstanding tag identification performance. The numbers of collision slots and idle slots are reduced by exploiting a novel predetection mechanism and slot size adjustment technique. Simulation results indicate that the proposed scheme reduces the tag identification time by approximately 30–60% compared with previously proposed predetection-based protocols. Moreover, the numbers of query cycles, collisions, and slots are smaller in the proposed scheme than in the previously proposed protocols.

1. Introduction

Radio-frequency identification (RFID) is a noncontact automatic identification technology that is widely used in modern industrial applications such as object tracking and identification, wireless sensor networks, inventory management, supply chain management, wireless sensor networks, indoor positioning, and the Internet of Things (IoT).[1–11] Conventional identification

*Corresponding author: e-mail: ckliang@chu.edu.tw
https://doi.org/10.18494/SAM.2021.3381

ISSN 0914-4935
https://myukk.org/

systems, such as QR codes and barcodes, cannot effectively perform automatic identification and data collection due to tag-reader contact limitations, tag visibility issues, and their low data reading rate. Nowadays, RFID systems can provide reliable and rapid communication without requiring direct contact between the reader and tags. Because of these characteristics, the functions of RFID technology not only go beyond object identification, but can also be used for localization and sensing applications.[6] Furthermore, due to the RF energy supply, components with sensing capability and chips can be integrated into RFID tags for simultaneous sensing and identification purposes. These systems are referred to as computational RFID (CRFID) systems.[7] CRFID systems allow programs to run on an embedded computer that is only powered by RF energy. Passive RFID sensors obtain RF energy from RF radiation to power the circuit that performs the sensing task and saves the sensing information in the RFID chip, which can be retrieved by RFID readers.

Due to the wide coverage and mobility of RFID interrogators, the acquisition of information from "things" with passive RFID sensor tags is no longer restricted to specific locations. As a result, such an RFID system with a large number of "things" can be implemented in many real-world applications. For example, wearable and wireless sensing devices do not limit a patient's movements and allow efficient and continuous medical monitoring of the patient, improving their quality of life. As a result, the healthcare industry is being revolutionized by RFID sensors in a way that benefits both patients and medical providers.[8,9] Furthermore, agriculture will need effective sensors to provide efficient solutions in the future.[10,11] Several studies have shown that it is necessary to significantly increase worldwide food production by 2050. The key to achieving efficient food production lies in the purposeful and thoughtful use of sensors and technology. Therefore, an effective RFID sensor tag identification algorithm is needed in the aforementioned applications.

Reducing the identification processing time is essential for a large-scale RFID system that consists of numerous tags within the range of RFID readers. A collision occurs if multiple tags respond to a reader's inquiry simultaneously. Thus, tag anticollision protocols are essential for the efficient identification of IDs through inexpensive passive tags in RFID systems.

Numerous studies on anticollision protocols have been conducted. Anticollision protocols may be classified into Aloha-based and tree-based protocols.[12–21] Although the probability of tag collisions is lower in Aloha-based protocols, they incur the tag starvation problem, which means that a particular tag may remain unidentified for a long time.[12–14] By contrast, tree-based protocols, such as the binary tree splitting protocol and query tree algorithm, ensure the identification of all tags.[15–21] However, they have a relatively long identification delay. In this study, we adopted tree-based protocols with the aim of reducing the identification delay.

In our previous studies, we have addressed the main design issue in the use of a predetection-based mechanism to reduce unnecessary idle slots and avoid collision slots.[17,21] In a predetection-based protocol, the reader allocates certain small timeslots for tags to respond to when determining the tag ID distribution. Once the tag ID distribution has been determined, only existing tags can respond in the corresponding timeslots during the following tag response cycle. Thus, all idle slots are eliminated and collisions are reduced. As a result, the previously proposed predetection-based scheme outperforms many other tree-based schemes, such as the

HQT and H^{2}QT protocols.[21] However, some collisions that cannot be detected existed in our previous scheme.

In this study, we employed the results found in previous studies to develop a novel collision detection scheme for use in the predetection cycle. The proposed identification scheme can (1) reduce the number of collision cycles in a tag response cycle to the maximum possible extent, (2) reduce the communication overhead between the reader and tags during the predetection process, and (3) complete the tag identification process in the shortest possible time.

The paper is organized as follows. Section 2 presents the concept and function of the predetection scheme. In Sect. 3, we introduce our tag identification technique, called the hybrid predetection-based query tree (HPQT) algorithm. Section 4 presents the simulation results of the proposed technique and compares its performance with that of previously proposed protocols. Finally, in Sect. 5, we conclude this study.

2. Predetection Scheme

To understand the main concept of the proposed anticollision algorithm, we must first determine the tag ID distribution to the maximum possible extent. When the tag ID distribution is determined, the reader sends the distribution information to the tags so that the tags can select an appropriate timeslot in which to respond during the tag response cycle. Then, different tags respond in different timeslots, which helps prevent collisions. Moreover, because the reader allocates the exact number of timeslots in which tags should respond, no slot is empty.

To implement our concept, we use a predetection-based technique to determine the distribution of tags. Note that the communication between passive tags and readers follows the *request–respond* mode, which results in several iterations of reader request and tag response operations during the tag identification process. In our predetection scheme, after the reader request, the tag response consists of three phases of operations: predetection, broadcast, and tag response phases. These phases are illustrated in Fig. 1.

The functions of these three phases are described as follows:

(1) *Predetection phase*: The reader allocates certain small timeslots for the tags, and each timeslot is responsible for collecting tag ID distribution information for a particular group. Furthermore, each small timeslot is assigned a unique label represented by a series of bits; this label identifies a particular group of tag IDs. Once a tag matches the prefix string, the tag

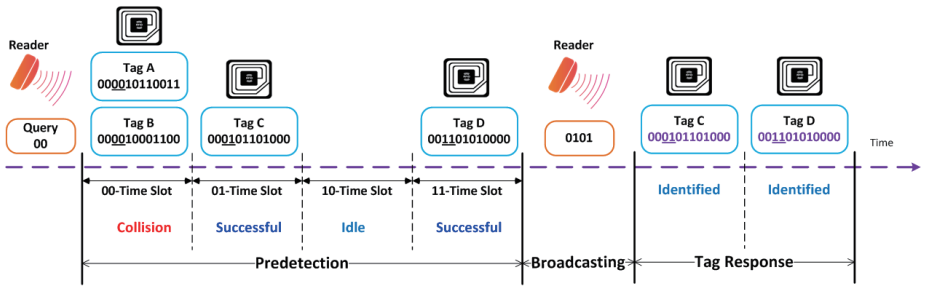

Fig. 1. (Color online) Tag response cycle in predetection-based algorithm.

sends a message to an appropriate small timeslot that has the same label as the following bits of its ID. Figure 1 shows four small allocated timeslots in the predetection phase. These slots are labeled 00, 01, 10, and 11. Tags that match the prefix string send a message to the 00, 01, 10, and 11 slots according to the subsequent two bits. After receiving tag responses, the reader recognizes the status of each timeslot as *collision*, *idle*, or *successful*. When only one tag responds, a *successful* state is generated; when no tag responds, an *idle* state is generated; and when more than one tag attempts to respond to the same small timeslot, a *collision* state is recognized. Once the state of each timeslot has been determined, the reader can realize the tag distribution for a particular group, which helps it determine whether the timeslot must receive the tag ID during the tag response phase.

(2) *Broadcast phase*: After the predetection phase, the reader encodes the status of each small timeslot into bit 0 or 1 and sends the encoded result to the tags so that the tags with IDs matching the prefixes can determine in which timeslot to respond to their IDs during the tag response phase. In general, bit 1 indicates an allocated timeslot in which tags should respond, and bit 0 indicates that no timeslot has been allocated in which tags should respond.

(3) *Tag response phase*: On receiving the bit string from the reader during the broadcast phase, each tag is aware that either a timeslot is allocated for responding with its ID or there is no need to respond during the phase. Thus, a sequence of tag responses can be obtained by the reader accordingly, and idle slots and collisions can be avoided.

3. Proposed Anticollision Scheme

The main function of the predetection phase in the aforementioned predetection-based scheme is to reveal the tag distribution, which helps to reduce collision slots and eliminate idle slots. In our previously proposed scheme, the predetection-based query tree (PDBQT) protocol,[9] the reader allocates four small timeslots—–that is, 00, 01, 10, and 11—–in the predetection phase for the tags to respond. According to the quaternary query tree mechanism, tags respond to the timeslot determined from the next two bits of their tag IDs if their tag IDs match the reader's query prefix string. A tag only responds to a four-bit random number instead of the entire tag ID, as shown in Fig. 2. After receiving responses, the reader recognizes the status of each small timeslot—–*idle*, *successful*, or *collision*. Some collisions are not recognized correctly because the

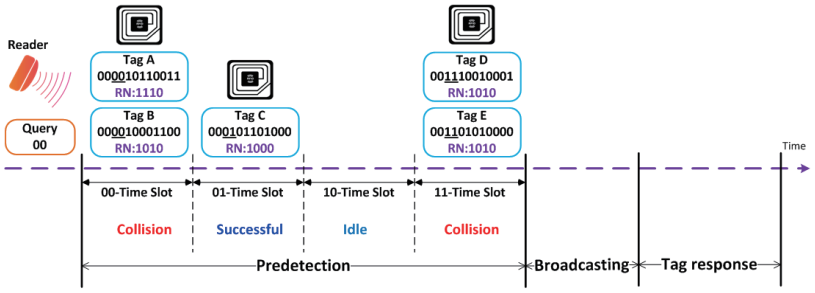

Fig. 2. (Color online) Tag responding to a four-bit random number in the PDBQT predetection phase.

same four-bit random number is sent by more than one tag; however, the number of collision slots can be substantially reduced in the tag response cycle. The drawback of the PDBQT protocol is the communication overhead of tags in the predetection phase; each tag must send back a four-bit random number to the reader. To reduce the communication overhead, we thus proposed another scheme, the efficient predetection-based query tree (EPDQT) protocol,[12] which differs from the PDBQT protocol in the predetection phase. In the EPDQT protocol, each tag that matches the reader's query string responds to the following bit, called the tag bit, determined from the next two bits of the tag ID and sends the tag bit to the timeslot determined from the next two bits, as shown in Fig. 3. Consequently, the communication overhead during the predetection phase can be substantially reduced. However, in the EPDQT protocol, some collisions cannot be detected in the predetection phase if more than one tag responds with the same tag bit to the same timeslot determined from the next two bits of the tag ID, thus generating collision slots in the tag response phase.

In this paper, a novel collision detection scheme is proposed that not only reduces the communication overhead but also increases the collision detection rate. The proposed scheme focuses on the tag response during the predetection phase. Unlike the PDBQT protocol, in the proposed scheme, a tag sends two bits––a tag bit and a random number bit––to a small timeslot, as illustrated in Fig. 4. The tag bit is obtained from the tag ID, and the random bit is a one-bit random number generated by the tag itself.

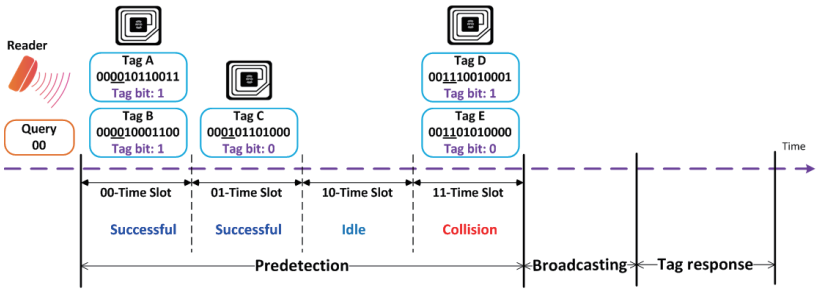

Fig. 3. (Color online) Tag responding to one tag bit in the EPDQT predetection phase.

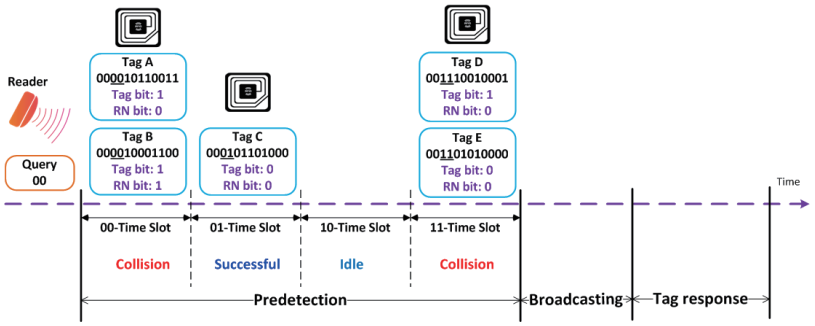

Fig. 4. (Color online) Tags responding to one-bit tag ID and one-bit random number in proposed predetection phase.

The procedure of our proposed predetection scheme can be described as follows. In our proposed anticollision algorithm, the tag response cycle is divided into three phases: predetection, broadcast, and tag response, after the reader sends a request. Before the tag response, the reader allocates 2^n small timeslots to collect the tag distribution information of different groups, where n is a parameter that can be defined by the algorithm's users. After a tag receives the prefix string sent by the reader, it will respond to the reader if its tag ID matches the received prefix string. The tag selects one of the 2^n timeslots according to the n bits following the received prefix string of its tag ID. Then, the responding tag sends the $(n + 1)$th bit following the received prefix string of its tag ID, combined with a random bit generated by the tag itself, to the reader. For example, if $n = 2$ and the tag ID is 10011001, then after receiving a two-bit prefix string, say 10, sent by the reader, the tag is aware that it should respond to the timeslot 01 because the two bits after the received prefix string are 0 and 1. Then, the tag takes the third bit, which is 1, as the tag bit; generates a binary random number, say 0, as the random bit; and sends the combined results, 10, to the timeslot in the predetection phase.

Once the predetection phase is completed, the reader is aware of the status of each timeslot in the predetection phase. By using the hybrid Manchester coding technique,[10] the reader can determine whether each received bit is ϕ (i.e., *idle*), 0, 1, or × (i.e., *collision*). Table 1 presents the results that can be obtained by a reader after receiving the response bits from tags during each small timeslot in the predetection phase.

Depending on the results obtained by the reader, the status of each small timeslot is recognized as *idle*, 0-*success*, 1-*success*, *tag-collision*, *random-collision*, or *tag-random-collision*. The *idle* state indicates that no tag is present in a particular group. The 0-*success* and 1-*success* states indicate that the reader receives 0 or 1, respectively, for the tag bit and the same random bits. The *tag-collision* state indicates that more than one tag responds to different tag bits and the same random bit to the reader. Similarly, the *random-collision* state indicates that more than one tag matches the prefix string, and the same tag bit but different random bits are sent to the reader. Finally, the *tag-random-collision* state indicates that more than one tag matches the prefix string, and different tag bits and random bits are sent to the reader. As in the previously proposed predetection-based scheme, in the proposed scheme, no timeslot is required

Table 1
Possible reader results in predetection phase of proposed scheme.

Tag bit	Random bit	Results	Broadcasted bits	Slots allocated
ϕ	ϕ	*idle*	00	0
0	0	0-*success*	10	1
0	1	0-*success*	10	1
1	0	1-*success*	01	1
1	1	1-*success*	01	1
0	×	*random-collision*	00	0
1	×	*random-collision*	00	0
×	0	*tag-collision*	11	2
×	1	*tag-collision*	11	2
×	×	*tag-random-collision*	00	0

Note: The ϕ symbols shown in the tag bit and random bit columns indicate no response from the tags.

in the tag response for the *idle* state. For both the 0-*success* and 1-*success* states, the reader allocates a timeslot in the tag response phase because only one tag may match the prefix string. For the *tag-random-collision* state, no timeslot is allocated because more than one tag matches the prefix string, and some additional queries are required to identify these colliding tags. For the *tag-collision* state, the reader realizes that two tag IDs match the prefix string and thus allocates two timeslots to identify these tags. By contrast, in the *random-collision* state, the reader realizes that more than one tag matches the prefix string and has the same next bit. Thus, no timeslot is required in this case. Accordingly, each state is encoded into a two-bit broadcasted string, and each bit in the broadcasted string indicates whether a timeslot is required in the tag response phase. Bit 1 indicates that a timeslot is allocated, whereas bit 0 indicates otherwise.

As each tag receives the broadcasted 2^{n+1}-bit string, it checks the next $(n + 1)$-bit string after the received prefix string of its tag ID to determine whether the corresponding bit in the broadcasted string is 0 or 1. Each tag responds to its tag ID in the tag response phase only if its corresponding bit in the broadcasted string is 1. Moreover, the exact timeslot for tags to respond is determined by counting the number of 1s in the received broadcasted string from the starting bit to the corresponding bit. For example, if $n = 2$, the tag ID is 10010101, and the prefix string is 10, then after receiving the broadcasted string 00111001, the tag checks the value of the next three-bit string, which is 010 or 2, after the received prefix string bits of its tag ID to determine whether the corresponding bit (i.e., the third bit from the left of the broadcasted string) is 1 or 0.

Let us present an example to clarify our discussion. Table 2 depicts the detailed operation of identifying nine tags with 15-bit tag IDs in our proposed scheme, where $n = 2$. The tag IDs are tag A: 001010000000000, tag B: 001101010000101, tag C: 010101100011110, tag D: 100010111111010, tag E: 101101000010001, tag F: 110111010001110, tag G: 110010010101010, tag H: 111001001010111, and tag I: 111110111001000. First, the reader sends an empty query string to the tags and allocates four small timeslots in the predetection phase (represented as 00, 01, 10, and 11) in which the tags should respond simultaneously. For the small timeslot 00, only tags A and B have the same label as the first two bits of the tag ID. Tags A and B respond with the subsequent bit of the first two bits of their IDs, which are bit 0 for both tags A and B, as the tag bit along with one random bit. Assume that the random bits of tags A and B are different. Then, the reader realizes that the small timeslot 00 is in the *random-collision* state, which is encoded as 00 for the broadcasted string, and the reader combines the timeslot 00 with the tag bit 1 to include a new query prefix 001 in the queue for further processing. For the small timeslot 01, only tag C responds to the tag bit and the random bit to the timeslot. The reader realizes the timeslot is in the 0-*success* state, which results in the broadcasted string 10. The small timeslots 10 and 11 are processed in a similar manner. After the predetection phase, the reader sends the broadcasted string, which is 00101100, to the tags. Then, all tags count the locations of 1s in the broadcasted string. The first bit 1 in the broadcasted string is the second bit from the left, which can be represented as 010 in binary. Thus, the tag matching the first three bits of 010, which is tag C, responds with its tag ID in the first timeslot in the tag response phase. After the tag response phase, the reader removes a query prefix from the queue as the new query string and repeats the identification process. The identification process continues until the prefix queue is empty. After four query cycles, all tags have been identified as shown in Table 2.

Table 2
Detailed operations of the proposed HPQT scheme for identifying nine tags with $n = 2$.

Query cycle	Prefix string	Phase	Time slots	Tags that respond	Tag bit	Random bit	Results	Broadcasted string	Queue
1	ϕ	Predetection	00	A, B	1	X	*random-collision*	00	001
			01	C	0	1	0-*success*	10	
			10	D, E	X	0	*tag-collision*	11	
			11	F, G, H, I	X	X	*tag-random-collision*	00	110, 111
		Broadcast						00101100	
		Tag response	010	C			identified		
			100	D			identified		
			101	E			identified		
2	001	Predetection	00	ϕ			*idle*	00	
			01	A	0	1	0-*success*	10	
			10	B	1	0	10-*success*	01	
			11	ϕ			*idle*	00	
		Broadcast						00100100	
		Tag response	010	A			identified		
			101	B			identified		
3	110	Predetection	00	ϕ			*idle*	00	
			01	G	0	1	0-*success*	10	
			10	ϕ			*idle*	00	
			11	F	1	1	1-*success*	01	
		Broadcast						00100001	
		Tag response	010	G			identified		
			111	F			identified		
4	111	Predetection	00	H	1	0	1-*success*	01	
			01	ϕ			*idle*	00	
			10	ϕ			*idle*	00	
			11	I	0	0	0-*success*	10	
		Broadcast						01000010	
		Tag response	001	H			identified		
			110	I			identified		

Note: The ϕ symbols shown in the prefix string and tags that respond columns indicate an empty query string and no response from tags, respectively.

4. Performance Evaluation

To verify the effectiveness of our proposed technique, we compare the performance of the HPQT protocol with that of the PDBQT and EPDQT protocols. We perform several experiments to verify the performance of the HPQT protocol. Each experiment is performed 30 times and the average is taken as the final result.

The simulation environment accords with the related regulations of the EPCglobal C1 G2 standard as follows.[22] An RFID system that consists of one reader and 5000–50000 non-duplicate tags within the reading range is employed. The length of the tag ID is 96 bits. Two distributions of tag IDs are considered, namely, uniform random and sequential distributions. In the sequential distribution, the tag IDs are in groups and consecutive. Let g denote the maximum

group size when the sequential distribution is used as a percentage of the number of tags to be identified. In our simulation, three different values of g are considered, namely, 10, 20, and 50%. The data transfer rate between the reader and tags is set to 128 kbit/s. The length of the reader command is 22 bits. The transmission waiting time from the reader to the tags and from the tags to the reader is set to 20 μs. For simplicity, we consider an ideal transmission channel between the reader and the tags and ignore the capture effect and path-loss effects.

The focus of the simulations is to determine the performance of the algorithms for different numbers of tags, with the performance reflected by the number of queries required, the delay time, the total number of transmission bits required, and the slot system efficiency. The slot system efficiency is defined as the ratio of the number of tags to the total number of slots taken to complete the tag identification.

4.1 Number of queries required for different numbers of tags

Figures 5–8 show the results for the total number of queries required to identify tags for both the uniform random and sequential distributions of various schemes. Figure 5 indicates that the EPDQT protocol outperforms both the HPQT and PDBQT protocols in terms of the number of queries required to identify tags when the uniform random distribution is used. The PDBQT protocol performs the worst among the three protocols. The reason for these results is relatively clear. The predetection phase in the PDBQT protocol eliminates all idle slots and detects most collision slots. Therefore, more query cycles are required to resolve collisions in the PDBQT protocol. By contrast, the EPDQT protocol allocates two slots in the tag response phase for the collision slots detected in the predetection phase. Thus, many tags are identified in the tag response phase, resulting in a considerable reduction in the number of queries. However, some collision slots still exist in the tag response phase. Our proposed HPQT protocol aims to reduce the collisions that occur in the tag response phase by detecting as many collisions as possible in the predetection phase. As illustrated in Fig. 5, the collision avoidance performance of the proposed HPQT protocol is similar to that of the PDBQT protocol.

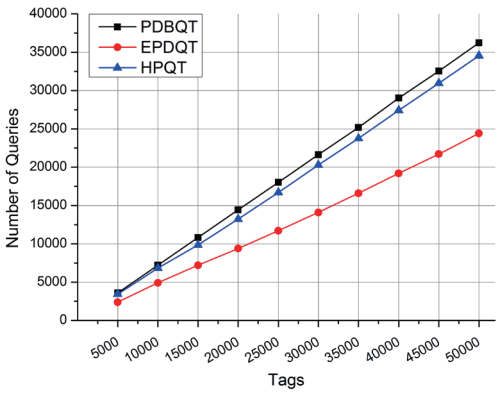

Fig. 5. (Color online) Total number of queries required for a uniform distribution.

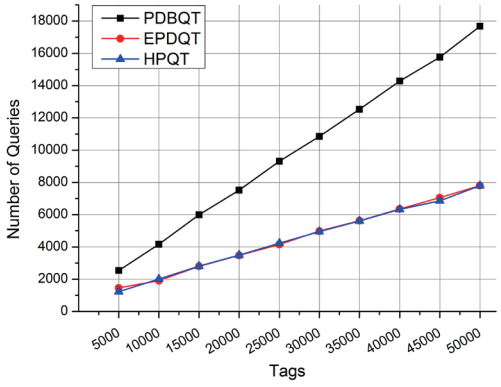

Fig. 6. (Color online) Total number of queries required for a group distribution with g = 10%.

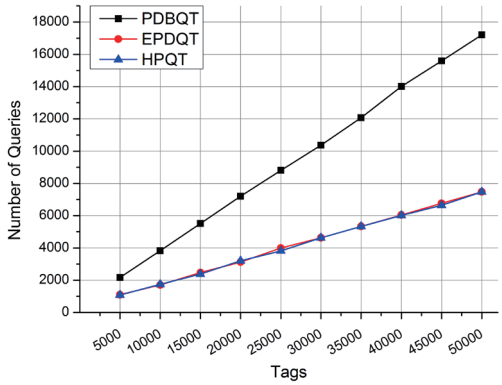

Fig. 7. (Color online) Total number of queries required for a group distribution with $g = 20\%$.

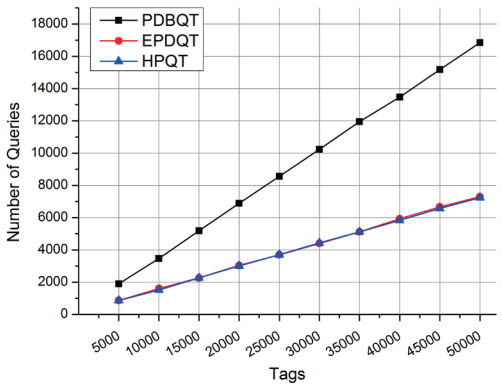

Fig. 8. (Color online) Total number of queries required for a group distribution with $g = 50\%$.

Figures 6–8 show the results of all protocols when the sequential distribution of tag IDs is used. These figures show that the number of queries in the PDBQT protocol remains almost unchanged with a change in the group size g. The numbers of queries required for both the HPQT and EPDQT protocols are similar when the group distribution of IDs is used and less than half the number required for the PDBQT protocol. This is because in the group distribution, many tags have long common prefixes and only differ in the last few least significant bits, which results in the tags becoming sibling leaves in the identification tree. Therefore, many tags respond to the same small timeslot in the predetection phase as compared with the number of tags responding to the same small timeslot when IDs are uniformly distributed. In such a scenario, the PDBQT protocol successfully detects more collisions than both the HPQT and EPDQT protocols, which results in a greater number of queries. Moreover, the number of queries required for the HPQT protocol is approximately 1–5% less than that for the EPDQT protocol, because the HPQT protocol detects more collision slots than the EPDQT protocol, and it identifies more tags in the tag response phase. Therefore, the number of queries required to identify tags in the HPQT protocol is less than that required in the EPDQT protocol.

4.2 Total transmission bits for different numbers of tags

Figures 9–12 show the communication complexity with different numbers of tags for the PDBQT, EPDQT, and HPQT protocols when uniform random and sequential distributions are used. We measured the communication complexity by calculating the total number of bits transmitted by these protocols in completing the tag identification. Figure 9 shows that the total number of transmission bits required to complete tag identification in each algorithm increases proportionally with the number of tags. The proposed HPQT protocol requires approximately 20–30% fewer transmission bits than the PDBQT and EPDQT protocols. The PDBQT protocol outperforms the EPDQT protocol. This is because the PDBQT protocol detects many collisions in the predetection phase, which results in only a few collision slots being wasted in the tag response phase. By contrast, the EPDQT protocol wastes numerous collision timeslots in the tag

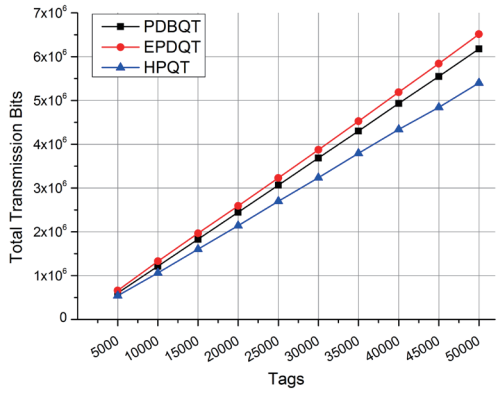

Fig. 9. (Color online) Total transmission bits required for a uniform distribution.

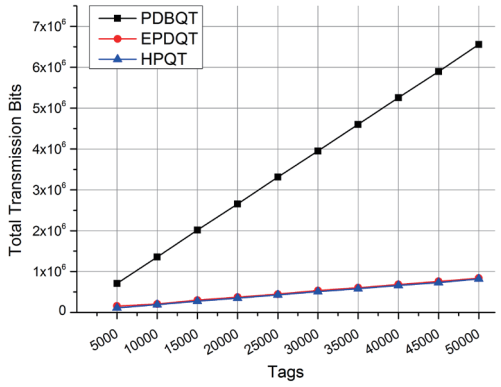

Fig. 10. (Color online) Total transmission bits required for a group distribution with g = 10%.

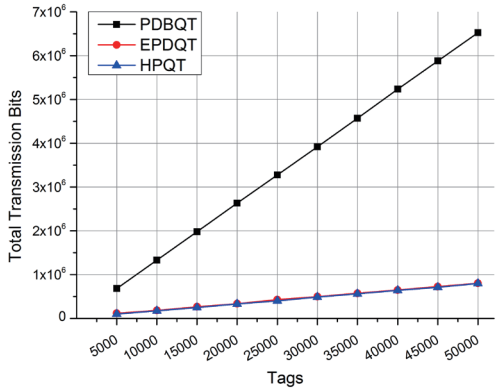

Fig. 11. (Color online) Total transmission bits required for a group distribution with g = 20%.

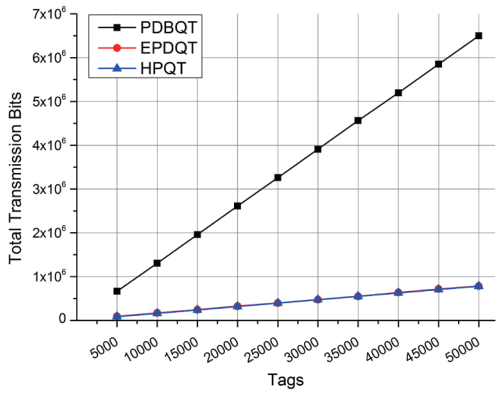

Fig. 12. (Color online) Total transmission bits required for a group distribution with g = 50%.

response phase. Moreover, our proposed HPQT protocol detects many collisions in the predetection phase and also identifies many tags in the tag response phase. Therefore, the HPQT protocol reduces the number of collision slots in the tag response phase and sends fewer queries than the PDBQT protocol.

Figures 10–12 show that the total number of transmission bits required to identify tags in the HPQT and EPDQT protocols when the sequential distribution is used is less than half of that required when the uniform random distribution is used. However, the total number of transmission bits required in the PDBQT protocol is similar for both sequentially distributed and uniformly distributed IDs. In the sequential distribution, because the tag IDs only differ in the last few least significant bits, many tags with the same prefix string respond to the same small slot in the predetection phase. Thus, the PDBQT protocol detects almost all the collisions, which results in many query cycles being performed to resolve the detected collisions. In the HPQT and EPDQT protocols, many tags respond with the same tag bit to the same small timeslot in the predetection phase, which also results in many query cycles being performed. However, in both

the HPQT and EPDQT protocols, three bits are added for the query prefix, compared with two bits in the PDBQT protocol; this means that the number of query cycles performed in the PDBQT protocol is greater than that in the HPQT and EPDQT protocols. Therefore, the total number of transmission bits required to identify tags in the HPQT and EPDQT protocols is much less than that in the PDBQT protocol. Furthermore, the total number of transmission bits required in the HPQT protocol is approximately 5% less than that in the EPDQT protocol. The reason for this is also clear. By employing a random bit, the HPQT protocol detects more collision slots in the predetection phase than the EPDQT protocol. Thus, the total number of transmission bits in the HPQT protocol is less than that in the EPDQT protocol.

4.3 Delay time for different numbers of tags

Figures 13–16 show the delay time observed with different numbers of tags for the PDBQT, EPDQT, and HPQT protocols when uniform and group distributions are used. We measure the delay time by calculating the total time required to complete the communication of all the query

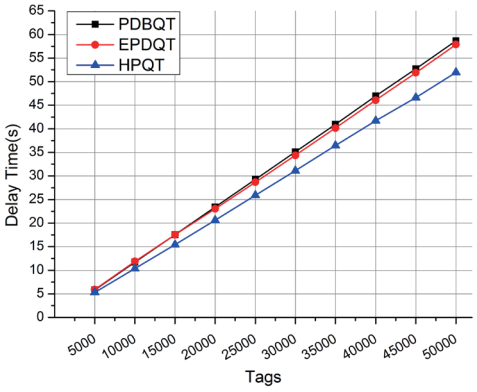

Fig. 13. (Color online) Time required to identify tags for a uniform distribution.

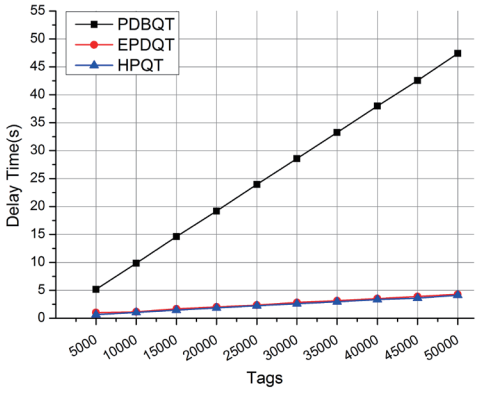

Fig. 14. (Color online) Time required to identify tags for a group distribution with g = 10%.

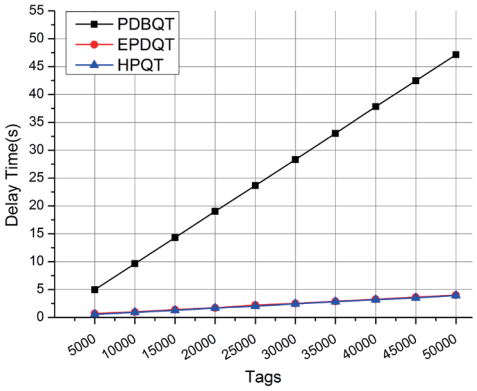

Fig. 15. (Color online) Time required to identify tags for a group distribution with g = 20%.

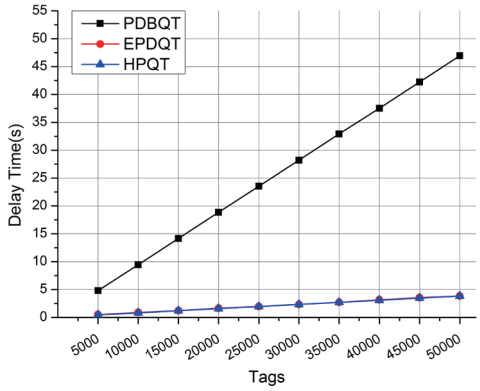

Fig. 16. (Color online) Time required to identify tags for group distribution with g = 50%.

commands and transmission bits and other communication overheads between the reader and tags. Figure 13 shows that when the number of tags increases, the HPQT protocol increasingly outperforms the PDBQT and EPDQT protocols. This is because the number of collision slots in the HPQT protocol is less than that in the EPDQT protocol. Thus, the total number of transmission bits in the HPQT protocol is less than that in the EPDQT protocol. Furthermore, the number of query cycles in the HPQT protocol is less than that in the PDBQT protocol, which results in the communication overhead in the HPQT protocol being lower than that in the PDBQT protocol.

Figures 14–16 show that the delay time observed in the HPQT and EPDQT protocols when the sequential distribution is used is much shorter than that when the uniform random distribution is used. This is because the tag IDs differ only in the last few least significant bits when the group distribution is used, which results in more tags with the same query prefix being recognized for each query cycle. Moreover, all protocols have similar performance in terms of the delay time regardless of the group size g. The HPQT and EPDQT protocols outperform the PDBQT protocol. This is because the number of queries in the HPQT and EPDQT protocols is much smaller than that in the PDBQT protocol when the sequential distribution is used.

4.4 Slot system efficiency for different numbers of tags

Figures 17–20 show the results of the system efficiency for the PDBQT, EPDQT, and HPQT protocols when uniform and group distributions are used. We measure the system efficiency by calculating the ratio of the number of tags to the total number of slots taken to complete the tag identification by these protocols. Figure 17 shows that the approaches have relatively constant slot efficiency as the number of tags increases. The slot system efficiency of the PDBQT protocol is significantly better than that of other protocols because the number of identification slots and the total number of slots increase as the number of tags increases, but the slot system efficiency does not vary much with the number of tags. In the HPQT and EPDQT protocols, each query cycle may allocate up to eight timeslots in which tags can respond. Some of the timeslots may identify a tag but the rest may encounter collisions. Therefore, slot utilization in the HPQT and

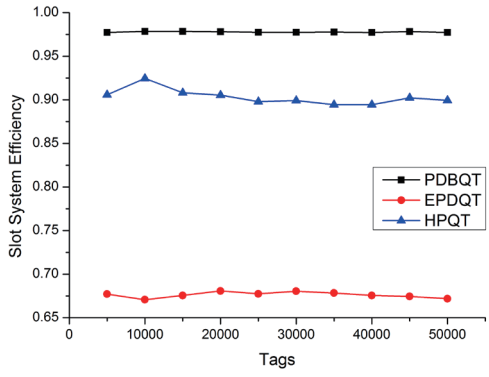

Fig. 17. (Color online) Slot system efficiency for a uniform distribution.

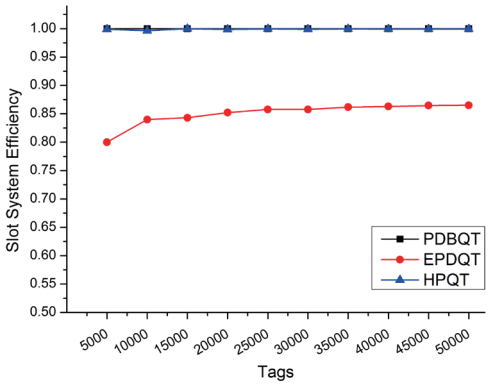

Fig. 18. (Color online) Slot system efficiency for a group distribution with g = 10%.

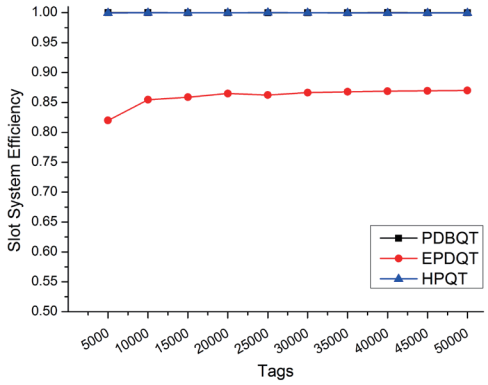

Fig. 19. (Color online) Slot system efficiency for a group distribution with g = 20%.

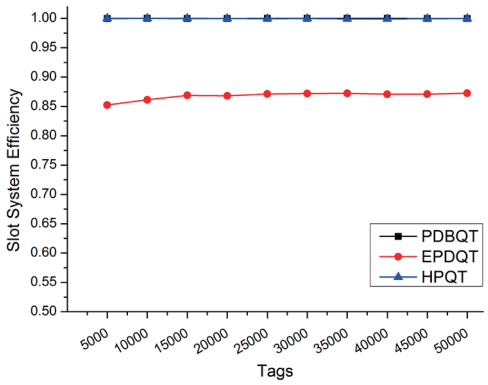

Fig. 20. (Color online) Slot system efficiency for a group distribution with g = 50%.

EPDQT protocols is less efficient than that in the PDBQT protocol, in which almost all identified slots are allocated for each query. Slot utilization in the EPDQT protocol is less efficient than that in the HPQT protocol because the HPQT protocol has a higher collision detection rate in the predetection phase.

Figures 18–20 show that the HPQT and PDBQT protocols outperform the EPDQT protocol in terms of the slot system efficiency when the group distributions are used regardless of the group size g. This is because the tags are siblings in the identification tree when the group distribution is used, and many collisions exist in the predetection phase during each query prefix. As a result, only a few collision slots occur in the tag response phase for the HPQT and PDBQT protocols. By contrast, many collision slots cannot be detected in the predetection phase in the EPDQT protocol, which results in numerous collision slots being produced in the tag response phase. Consequently, the HPQT and PDBQT protocols have higher slot system efficiency than the EPDQT protocol. Moreover, both the HPQT and PDBQT protocols experience similar slot system efficiencies (approximately 98–100%) irrespective of the value of N.

5. Conclusions

The widespread application and rapid technical progress of RFID sensing techniques have led to innovative solutions in various fields of application, and these solutions are promising for IoT sensing applications. Therefore, IoT comprising an enormous number of RFID sensor tags can become a reality. Reducing the identification processing time is an essential and challenging task for a large-scale RFID system. Many collisions may occur during the tag identification process owing to the characteristic of large-scale RFID systems. Identification protocols, such as the PDBQT and EPDQT protocols, can reduce the number of collision slots and eliminate idle slots by allocating some small timeslots in the predetection process, which results in a reduction of the number of query cycles required to identify tags. In this study, we extended the previous PDBQT and EPDQT protocols, in which a tag responds to small timeslots in the predetection phase with two bits, a tag bit and a random bit. Therefore, our proposed HPQT protocol has the advantages of both the PDBQT and EPDQT protocols. By comparing the HPQT protocol with

the PDBQT and EPDQT protocols in terms of the total number of required queries, the total number of bits transmitted, the delay time, and the slot system efficiency, the simulation results show that the HPQT protocol outperforms the PDBQT protocol in terms of the total number of required queries, the total number of required transmission bits, and the delay time, regardless of the tag ID distribution type. The HPQT protocol also exhibits similar performance to the PDBQT protocol in terms of the slot system efficiency when a group distribution is used. The proposed HPQT protocol also performs better than the EPDQT protocol in terms of the total number of required transmission bits, the delay time, and the slot system efficiency, regardless of the tag ID distribution type. Moreover, the HPQT and EPDQT protocols outperform the PDBQT protocol in terms of the delay time, regardless of the tag ID distribution type. For a group distribution, the HPQT protocol has similar slot system efficiency to the PDBQT protocol. Therefore, the HPQT protocol is more efficient than the PDBQT and EPDQT protocols.

References

1 Y. Li and X. Ding: Proc. 2nd ACM Symp. Information, Computer and Communications Security (ASIACCS, 2007) 234–241. https://doi.org/10.1145/1229285.1229318
2 M. Caldara, B. Nodari, V. Re, and B. Bonandrini: Procedia Eng. **87** (2014) 344. https://doi.org/10.1016/j.proeng.2014.11.746
3 L. Cui, Z. Zhang, N. Gao, Z. Meng, and Z. Li: Sensors **19** (2019) 4012. https://doi.org/10.3390/s19184012
4 C. Strangfeld, S. Johann, and M. Bartholmai: Sensors **19** (2019) 5514. https://doi.org/10.3390/s19245514
5 H. Xe, Y. Ding, P. Li, R. Wang, and Y. Li: Sensors **17** (2017) 1806. https://doi.org/10.3390/s17081806
6 W. Shi, J. Du, X. Cao, Y. Yu, Y. Cao, S. Yan, and C. Ni: Sensors **19** (2019) 968. https://doi.org/10.3390/s19040968
7 EVAL01-Spectre-RM: http://www.farsens.com/en/products/eval01-spectre-rm/ (accessed April 2021).
8 A. Rashee, E. Iranmanes, W. Li, X. Fen, A. S. Andrenk, and K. Wan: Proc. IEEE Int. Conf. RFID Technology & Application (RFID-TA, 2017) 243–247. https://doi.org/10.1109/RFID-TA.2017.8098911
9 S. Yang, M. Crisp, R. V. Penty, and I. H. White: IEEE J. Radio Freq. Identif. **2** (2018) 159. https://doi.org/10.1109/JRFID.2018.2822770
10 A. Bothe, J. Bauer, and N. Aschenbruck: Proc. IEEE Int. Conf. RFID Technology and Applications (RFID-TA, 2019) 505–510. https://doi.org/10.1109/RFID-TA.2019.8892140
11 B. Saggin, Y. Belaizi, A. Vena, B. Sorli, V. Guillard, and I. Dedieu: Proc. IEEE Int. Conf. RFID Technology and Applications (RFID-TA, 2019) 484–487. https://doi.org/10.1109/RFID-TA.2019.8892248
12 D. K. Klair, K. Chin, and R. Raad: IEEE Commun. Surv. Tutorials **12** (2010) 400. https://doi.org/10.1109/SURV.2010.031810.00037
13 Y. Xu and Y. Chen: Proc. 2015 IEEE Int. Conf. RFID (RFID, 2015) 1–8. https://doi.org/10.1109/RFID.2015.7113066
14 L. Zhu and T. Yum: IEEE Trans. Commun. **58** (2010) 2725. https://doi.org/10.1109/TCOMM.2010.080310.090421
15 C. Law, K. Lee, and K. Y. Siu: Proc. 4th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM, 2000) 75–84. https://doi.org/10.1145/345848.345865
16 G. T. Peeters and B. Van Houdt: IEEE Trans. Commun. **58** (2010) 3056. https://doi.org/10.1109/TCOMM.2010.093010.090222
17 C. K. Liang, Y. C. Chien, and C. H. Tsai: Proc. 7th Int. Conf. Sensor Technologies and Applications (SENSORCOMM, 2013) 51–56.
18 W. Zein and E. Shaaban: Proc. 2nd Int. Conf. Computer Engineering and Technology (ICCET, 2010) 349–353. https://doi.org/10.1109/ICCET.2010.5485458
19 J. Myung, W. Lee, and T. K. Shih: IEEE Trans. Multimedia **8** (2006) 1096. https://doi.org/10.1109/TMM.2006.879817
20 M. Chu, Z. Qian, and X. Wang: Int. J. Distrib. Sens. Netw. **14** (2018) 1. https://doi.org/10.1177/1550147718811823
21 Y. H. Lin and C. K. Liang: J. Sens. Actuator Netw. **7** (2018) 13. https://doi.org/10.3390/jsan7010013
22 EPCglobal: EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID, https://www.gs1.org/sites/default/files/docs/epc/Gen2_Protocol_Standard.pdf (accessed February 2021).

About the Authors

Kuo-Sen Hsu received his M.S. degree from the Department of Computer Science and Information Engineering at Chung Hua University, Hsinchu, Taiwan, Republic of China, in 2019. His research interests include wireless networks, sensor networks, and RFID systems.

Chiu-Kuo Liang received his Ph.D. degree in computer science from the National Tsing Hua University, Taiwan, in 1990. 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, RFID systems, Internet of Things, and parallel processing.