Sensors and Materials, Vol. 32, No. 10 (2020) 3443–3461
MYU Tokyo

S & M 2350

Enhanced Identification Algorithm Based on Dynamic Slots Collision Tracking in Radio Frequency Identification Systems

Yu-Hsiung Lin[1] and Chiu-Kuo Liang[2*]

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

(Received April 17, 2020; accepted July 31, 2020)

Keywords: RFID sensors, anticollision algorithm, dynamic slots, collision tracking tree

Radio frequency identification (RFID) technology is one of the key enabling technologies to realize the IoT. With the capabilities of sensing, wireless communication, and wireless-powered, non-line-of-sight transmission, lightweight RFID sensor tags are a critical enabling technology for future IoT applications, such as logistics, manufacturing, and healthcare. In contrast to alternatives, such as barcodes and QR codes, the radio-frequency-powered identification approach featuring the simultaneous reading of multiple tags allows connected "things" to be identifiable for further data communication and integration. Therefore, an effective anticollision protocol should be developed between the reader and the tags to achieve rapid identification, especially in a system with a large number of RFID sensor tags. The dynamic slots collision tracking (DSCT) algorithm, which is based on a collision tracking technique, can reduce the prefix and the iteration overhead through a time-divided responding scheme. The DSCT protocol performs well when consecutive collisions occur. However, for nonconsecutive collisions, DSCT generates many idle time slots. Thus, the identification time cannot be reduced. In this paper, we present an enhanced DSCT (EDSCT) algorithm to reduce the identification time for nonconsecutive collisions. The simulation results reveal that the proposed algorithm can effectively reduce the identification delay and improve the slot efficiency.

1. Introduction

Radio frequency identification (RFID) is a modern technology that has been widely employed in various industrial applications, such as object and human tracking, supply chain management, vehicle positioning, and inventory management.[1–3] Conventional identification systems, such as barcodes, are inefficient for conducting automatic identification and data collection because of their low read rate, tag visibility problems, and tag-reader contact limitations. However, RFID systems can provide rapid and reliable communication without requiring tag visibility or direct contact between readers and tags. Owing to these features, RFID technology transcends the function of object identification and has also been used for

*Corresponding author: e-mail: ckliang@chu.edu.tw
https://doi.org/10.18494/SAM.2020.2928

ISSN 0914-4935

localization and sensing applications.[4,5] Furthermore, since an RFID chip is powered by RF energy, components with sensing capability can also be integrated into RFID tags for simultaneous identification and sensing purposes. RFID tags with sensing components can ultimately provide identification and sensing capability in a wireless-powered, contactless, and non-line-of-sight manner. Different from traditional sensing, simultaneous wireless information and power transfer (SWIPT) has become a new paradigm of sensing and communication and could reshape the future IoT world.[6,7] Owing to the wide coverage and mobility of RFID interrogators and the possibility of completely passive RFID sensors, the measurement of RFID-sensor-tagged "things" is no longer limited to specific locations, and there is also no need to frequently change the batteries of the RFID sensors. Therefore, such a large-scale RFID system can be realized for many applications.

A research topic in large-scale RFID systems is how to decrease the processing time required to identify a certain number of tags within the range of an RFID interrogator. Anticollision protocols are required to achieve rapid tag identification. Collisions may occur when multiple tags simultaneously respond to a reader's inquiry. Generally, anticollision protocols aim to reduce the number of collisions during the tag identification process. Moreover, collisions can be categorized into two types: reader and tag collisions. Reader collisions occur when two or more neighboring readers simultaneously inquire about a tag. In this case, the tag cannot accurately provide its unique identification code (ID) to inquiring readers. Reader collisions can be easily eliminated by detecting these collisions and communicating with other readers. Tag collisions occur when more than one tag concurrently respond to a reader. In this scenario, the reader cannot identify either of the tags that responded concurrently. In RFID systems that include low-cost, passive tags, the tags can only respond to the reader's inquiries. Therefore, tag anticollision protocols are crucial for efficiently identifying tag IDs in RFID systems.

Numerous studies pertaining to anticollision protocols exist. These protocols can be classified into two categories: Aloha-based and tree-based anticollision schemes.[8–13] In the aloha-based schemes, an RFID reader creates a frame that contains several time slots and then adds the frame length to the inquiry message sent to the tags in the interrogation zone of the reader. Tags respond to the reader's inquiry by selecting a random time slot. The tags that simultaneously respond in the same time slot cannot be recognized by the reader owing to the occurrence of collisions. In this case, the reader has to repeatedly send inquiries until all tags are identified. Thus, aloha-based schemes require long processing times when they are used for large-scale RFID systems.

In tree-based schemes, RFID readers use a scheme similar to the binary search algorithm for recognizing tags. When a tag collision occurs, readers split collided tags into two subgroups and repeat the process until the tag IDs are recognized. Therefore, readers using a tree-based scheme can identify all tags within the interrogation range. Law *et al.* proposed a memoryless protocol known as a query tree (QT) through a binary splitting strategy for the tag identification problem.[10] If a reader transmits a k-length prefix, then the tags with tag IDs containing the same first k bits as the prefix transmit from the $(k + 1)$th bit to the end bit of their tag IDs. Moreover, if collision occurs while transmitting the tag IDs, then an extended prefix formed by attaching a "0" or "1" bit to the prefix is transmitted. Furthermore, if no collision occurs, then

the reader identifies one of the tags. Choi *et al.* proposed a fast anticollision algorithm known as the Improved Bit-by-bit Binary-Tree (IBBT) algorithm for use in ubiquitous identification systems and evaluated its performance through three existing schemes.[11] The reader used in the IBBT algorithm sends requests to all bits of tag IDs. After the reader receives responses from the tags, it saves the results of each receiving bit of the tag IDs. Therefore, the reader can identify the collided bits and sequentially sends requests to tags only for the collided bits in a bit-by-bit manner. Myung *et al.* proposed an adaptive memoryless tag anticollision protocol, which is an extended scheme based on the QT protocol.[12] The reader used in this approach employs a queue to maintain prefixes and an additional candidate queue for maintaining prefixes of identified and no-response nodes obtained during the previous identification attempt. Thus, the collision period can be shortened when the number of tags increases. Bhandari proposed the intelligent QT (IQT) algorithm, which is a modified version of the QT protocol for cases in which tags have some common prefixes.[13]

When aloha-based protocols are used, a particular tag may not be identified for a long time. However, tree-based protocols deliver 100% tag identification, but the identification delay is relatively longer owing to the large number of collisions and the large transmission overhead. Therefore, resolving collisions and reducing the transmission overhead are major issues that should be studied for tree-based anticollision protocols. Zhou *et al.* proposed the collision tracking tree algorithm (CTTA), which is based on a QT scheme but does not include the collision tracking process.[14] In the CTTA, a tag sends its ID from the $(k + 1)$th bit to the end bit if the prefix is the same as the tag's first k bits. However, if a collision is detected, then the reader transmits a signal to stop the tags from sending their IDs. Although the process of the CTTA appears similar to that of a QT, the CTTA forms the next prefix by using the bits received before collision and reduces time wastage due to collision when the bits are received. Gou *et al.* proposed the bit-collision-detection-based QT (BQT) protocol to reduce the number of collisions in the tag identification process.[15] In the BQT protocol, a reader sends a query string with the same length as that of tag IDs to all tags. Each bit position of the query string may include "0", "1", or a wild mask (*). These tags match the bit positions of values "0" and "1" and must respond with their values for the position corresponding to the wild mask "*". Jia *et al.* proposed a collision tree (CT) algorithm that is also based on a QT with the aim of eliminating collisions and idle time.[16] The proposed algorithm both generates prefixes and splits tags according to the first collided bit. Thus, the generated idle slots are eliminated effectively. Lai *et al.* proposed an optimal binary tracking tree protocol that employs a bit estimation algorithm to split tags into small sets and then uses a binary tracking tree method to quickly identify tags.[17] Su *et al.* proposed a hybrid anticollision algorithm, known as the anticollision protocol based on improved collision detection (ACP-ICD), to reduce the number of tag collisions.[18] Their approach incorporates the concepts of bit-tracking technology and dual-prefix matching into a collision arbitration mechanism in the RFID system. Landaluce *et al.* presented a bit window procedure to manage the length of tags' bit streams to reduce the energy consumption due to collisions.[19] They modified both QT and CT protocols by incorporating the bit window procedure and proposed the query window tree and collision window tree protocols. However, these algorithms have complicated implementation procedures and high identification delays.

Choi *et al.* proposed the bi-slotted collision tracking tree algorithm (BSCTTA) for reducing the transmission overhead through bi-slotted responses to increase the tag identification speed and the overall read rate and throughput in large-scale RFID systems.[20] The BSCTTA also employs the collision tracking technique. This algorithm appears similar to the CTTA; however, the BSCTTA provides two time slots in each query for tags to respond depending on their first-replied bit values. In the BSCTTA, a reader sends $(n - 1)$-length inquiring bits (that is, a prefix) to tags. Then, the tags that lie in the field of the reader send their tag IDs to the reader if the inquiring bits are the same as the first $(n - 1)$ bits of the tag IDs. When the tags send their IDs to the reader, they select one of the two time slots provided depending on whether the nth bit is "0" (first slot) or "1" (second slot). Thus, the nth bit contains the time slot details. After the tags decide the corresponding time slot to send their IDs, they send the remaining bits of their IDs from the $(n + 1)$th bit to the end until an ACK signal, which is sent from the reader when a collision occurs, is received. If a collision occurs, the reader saves a new prefix in the last input first output (LIFO) order.

In our previous study,[21] we proposed a novel technique for resolving collisions, which outperforms the BSCTTA scheme in terms of the tag identification speed and the overall throughput in large-scale RFID systems. We addressed the key design aspect of using a multiple-time-slotted technique known as the dynamic slots collision tracking (DSCT) algorithm for providing tag responses based on the prefix bits in the response bits in each query cycle to reduce the prefix and the iteration overhead.[21] The reader used in the DSCT protocol allocates multiple time slots based on the length of the consecutive collision bits that were obtained in the previous query cycle for tags to respond. Thus, the multiple time-slotted response obtained improves the tag identification efficiency of large-scale RFIDs.

Huang and Chen proposed an improved dynamic slots collision tracking tree anticollision algorithm (IDSCTTA) to eliminate the idle time slots during each query cycle in our previous DSCT algorithm.[22] To eliminate the idle time slots, the proposed IDSCTTA scheme utilizes a bit change method (BCM) technique. In each query cycle, if the consecutive collisions have been tracked by the reader, the IDSCTTA provides an additional time slot for those collided tags to transmit the changed bits to the reader so that the reader can realize which time slots are needed to resolve the collisions. Although the IDSCTTA can eliminate the idle time slots, the communication overhead between a reader and tags increases. As a result, the tag identification delay cannot be reduced compared with that of our previous DSCT scheme.

The present study is an extension of our previous work and investigates the optimal assignment of tag response time slots regardless of the consecutiveness of collision bits. The identification scheme proposed in this study (1) reduces the communication overhead between a reader and tags during the identification process, (2) reduces the number of query cycles, and (3) completes the identification process as quickly as possible.

Our contributions are summarized as follows.

- We introduce a tag identification problem observed in RFID systems, that is, a reader collects tag IDs and their sensing data to minimize the time required to complete ID identification and information collection.

- We proposed a novel design that uses a multiple-time-slotted technique for providing tag responses when collisions are detected. We exploited the collision tracking scheme when tags respond to a reader and allocated an adequate number of time slots in the next query cycle for resolving collisions. The number of allocated time slots depends on the number of detected collision bits regardless of whether they are consecutive or not. The maximum length of collision bits that can be detected is defined by users when using this algorithm.
- The simulation results of this study validate that the performance of the proposed approach is better than that of the previously proposed collision-tracking-based approaches.

The rest of this paper is organized as follows. Section 2 presents the concept of the dynamic slot allocation scheme as the preliminary work of this study. Section 3 introduces the proposed collision-tracking-based tag identification technique. Section 4 presents the performance evaluation of the proposed scheme—the enhanced DSCT (EDSCT) algorithm. Performance comparisons and analyses are also presented. Finally, the paper is concluded in Sect. 5.

2. DSCT Scheme

The primary idea of the proposed anticollision algorithm is the allocation of an appropriate number of time slots for tags to respond during each query cycle. When time slots are allocated, readers send information to tags so that the corresponding tags can respond in appropriate time slots. Thus, different tags respond to the reader in different time slots. This method substantially reduces the number of collisions.

In this study, we proposed a collision tracking scheme to track the collision bits and provide multiple time slots for tags to respond. The maximum length of the collision bits, which is denoted as C_{max}, that can be tracked using the scheme was varied and can be defined by users while using the proposed scheme. In each query cycle, a reader determines the appropriate number of time slots for tags to respond depending on the number of collision bits that are tracked in the previous query cycle. The reader stores two values for identifying tags: the length of collision bits C and the length of inquiring bits n (prefix). When the length of collision bits is tracked on the basis of the response provided by tags, the reader determines the number of time slots required for tags to respond in the next query cycle to be 2^C. This means that the reader provides 2^C time slots for tags to respond. The 2^C time slots are represented by C-bit values from 0 to $(2^C - 1)$. On the basis of this characteristic, the RFID systems can reduce the time required for identification by using the following procedure.

1) REQUEST: A reader sends prefix-inquiring bits of length n to tags.
2) RESPONSE: The tags present in the interrogation zone of the reader respond by transmitting their tag IDs to the reader if the inquiring bits are the same as the first n bits of the tag IDs.
   - When the tags transmit their IDs to the reader, they select one of the 2^C time slots based on the $(n + 1)$ to $(n + C)$ bits of the tag IDs. Thus, the time slots indicate the values of the $(n + 1)$th bit to the $(n + C)$th bit.

- Tags send their IDs from the $(n + C + 1)$th bit until the ACK signal is obtained from the reader.

3) DECISION: On the basis of whether collision has occurred and the length of consecutive collision bits C', the reader decides whether to continue the procedure using the following conditions:
   - If collisions occur and $C' \leq C_{max}$, then the reader sends ACK signals to the tags and saves a new prefix in the LIFO stack.
   - If a collision occurs at the last bit of the tag IDs, then the reader assumes that there are two tags because of the uniqueness of the tag IDs.
   - If no collision occurs, then the reader identifies a tag and sends an ACK signal after receiving the last bit.
   - If no tag responds, then the reader sends an ACK signal after the time allocated to the first received bit is over.
4) Perform the aforementioned steps until the LIFO stack is empty.

The previously proposed DSCT scheme focuses on the consecutive collision bits. Once a nonconsecutive situation occurs, the reader sends an ACK signal to the tags to notify them to stop transmitting their remaining ID bits. Then, in the subsequent query cycles, the reader allocates an appropriate number of time slots for tags to respond. The number of allocated time slots is based on the length of consecutive collision bits, which are tracked by the reader. If the length of the tracked consecutive collision bits is two, then four possible tag IDs are present for these two collision bits. Thus, the reader allocates 2^c time slots for the tags when the length of collision bits is c.

The following example is presented to provide a better understanding of our previously proposed scheme. Table 1 presents the operation of the scheme for eight tags with an ID length of 10 bits. The tag IDs are "0000001010", "0001011011", "0001001101", "0001011101", "0100010011", "0101000101", "0101100111", and "0101101001". Initially, the reader sends an empty query string to the tags in the interrogation zone and allocates one time slot for the tags to respond to simultaneously, which results in a collision. The reader realizes that a collision has occurred when the collision bit is obtained in the second bit, which is followed by a "0"-bit signal. This step introduces a nonconsecutive collision situation. Thus, the reader sends an ACK signal to all tags to make them cease the transmission of their remaining ID bits and updates the tag response string as "0*0". Subsequently, the reader pushes the collision prefix string "0" and a c-value of one into the stack. Then, the reader fetches the query string "0" from the prefix stack, broadcasts the string to all tags, and allocates two time slots for the corresponding tags to respond. When the query prefix string sent by the reader is received, the tags with the prefix bit matched with "0" send their remaining IDs back to the reader by selecting the corresponding time slot based on the second bit of their IDs. After the tags have sent their IDs in time slot "0", the reader recognizes the results from the tag response to be "0*0". Thus, the reader updates the response string as "000*0" and then pushes the collision prefix string "000" into the stack along with the c-value of two. Similarly, the reader pushes the collision prefix string "010" with a collision bit length of two into the stack after the tags send their IDs in time slot "1". Then, the reader begins a new query cycle in the same manner

Table 1
Detailed query cycle of the DSCT scheme for identifying eight tags.

Query cycle	Prefix string	Time slots	Tag response	Reader update string	Results	Prefix stack
1	(‘’,0)	all tags	‘0*0’	‘0*0’	Collision	(‘0’,1)
2	(‘0’,1)	‘0’	‘0*0’	‘000*0’	Collision	(‘000’,1)
		‘1’	‘0**’	‘010**’	Collision	(‘000’,1), (‘010’,2)
3	(‘010’,2)	‘00’	‘10011’	‘0100010011’	Identified	(‘000’,1)
		‘01’	ϕ	ϕ	Idle	(‘000’,1)
		‘10’	‘00101’	‘0101000101’	Identified	(‘000’,1)
		‘11’	‘0**’	‘010110**’	Collision	(‘000’,1), (‘010110’,2)
4	(‘010110’,2)	‘00’	ϕ	ϕ	Idle	(‘000’,1)
		‘01’	‘11’	‘0101100111’	Identified	(‘000’,1)
		‘10’	‘01’	‘0101101001’	Identified	(‘000’,1)
		‘11’	ϕ	ϕ	Idle	(‘000’,1)
5	(‘000’,1)	‘0’	‘001010’	‘0000001010’	Identified	ϕ
		‘1’	‘0*1’	‘00010*1’	Collision	(‘00010’,1)
6	(‘00010’,1)	‘0’	‘1101’	‘0001001101’	Identified	ϕ
		‘1’	‘1**’	‘0001011**’	Collision	(‘0001011’,2)
7	(‘0001011’,2)	‘00’	ϕ	ϕ	Idle	ϕ
		‘01’	‘1’	‘0001011011’	Identified	ϕ
		‘10’	‘1’	‘0001011101’	Identified	ϕ
		‘11	ϕ	ϕ	Idle	ϕ

*Note: The ϕ symbols shown in the Tag response, Reader update string, and Prefix stack columns indicate no response from tags, an empty string, and an empty stack, respectively.

by fetching the prefix string “010” from the stack, sending the string to all tags, allocating four time slots, and waiting for the tags to respond. Thus, two tag responses are obtained in time slots “00” and “10”, two collision bits are detected in time slot “11”, and no tag response is obtained in time slot “01”. The identification process proceeds until the prefix stack is empty.

Finally, all tags can be identified after four query cycles. The DSCT scheme uses 25 bits for prefixes and 44 bits for tag responses. Thus, the use of the DSCT scheme to recognize eight tags has a total communication overhead of 69 bits.

For comparison, Table 2 presents the results obtained after identifying the eight tags presented in Table 1 through the BSCTTA. Table 2 shows that the BSCTTA uses 29 bits for prefixes and 43 bits for responses. Thus, the use of this algorithm for identifying eight tags has a total overhead of 72 bits. The DSCT scheme can reduce the transmission overhead of the BSCTTA in the tag identification process by reducing the number of query cycles.

3. Proposed Anticollision Scheme

The DSCT scheme proposed in the previous study can reduce the number of query cycles by allocating a higher number of time slots during each query cycle for tags to respond. Thus, the number of transmission bits sent by the reader can be dramatically reduced. As a result, the total transmission overhead can be reduced. However, the DSCT scheme can only track consecutive collisions. This means that the reader may require a higher number of query cycles to finish the identification process once a collision is tracked regardless of the collision

Table 2
Detailed query cycle of the BSCTTA for identifying the eight tags presented in Table 1.

Query cycle	Prefix string	Time slots	Tag response	Reader update string	Results	Prefix stack
1	‘’	‘0’	‘*’	‘0*’	Collision	‘0’
		‘1’	ϕ	ϕ	Idle	‘0’
2	‘0’	‘0’	‘0*’	‘000*’	Collision	‘000’
		‘1’	‘0*’	‘010*’	Collision	‘000’, ‘010’
3	‘010’	‘0’	‘010011’	‘0100010011’	Identified	‘000’
		‘1’	‘*’	‘0101*’	Collision	‘000’, ‘0101’
4	‘0101’	‘0’	‘00101’	‘0101000101’	Identified	‘000’
		‘1’	‘0*’	‘010110*’	Collision	‘000’, ‘010110’
5	‘010110’	‘0’	‘111’	‘0101100111’	Identified	‘000’
		‘1’	‘011’	‘0101101011’	Identified	‘000’
6	‘000’	‘0’	‘001010’	‘0000001010’	Identified	ϕ
		‘1’	‘0*’	‘00010*’	Collision	‘00010’
7	‘00010’	‘0’	‘1101’	‘0001001101’	Identified	ϕ
		‘1’	‘1*’	‘0001011*’	Collision	‘0001011’
8	‘0001011’	‘0’	‘11’	‘0001011011’	Identified	ϕ
		‘1’	‘01’	‘0001011101’	Identified	ϕ

*Note: The ϕ symbols presented in the Tag response, Reader update string, and Prefix stack columns indicate no response from tags, an empty string, and an empty stack, respectively.

bit length. We proposed an enhanced version of the DSCT scheme in this study to reduce the number of query cycles—the EDSCT algorithm. The primary idea of the EDSCT algorithm is to allow the reader to track all collisions until the length of the tracked collision bits attains the value of the maximum collision bit C_{max}. After the value is attained, the reader adds a new query cycle for resolving these collision bits by allocating 2^{Cmax} time slots for tags to respond. The identification process in the EDSCT scheme is similar to that in the DSCT scheme, except that the collision bits may not always be consecutive. Therefore, the tags should know the positions of the collision bits to select the appropriate time slots for responding. Therefore, we used the protocol adapted in the BQT protocol with an appropriate modification for our method.[15] The proposed protocol is as follows. Initially, the reader broadcasts an empty string to all tags in its interrogation zone and allocates only one slot for all tags to respond. Subsequently, each tag responds simultaneously, thus resulting in collisions. The reader checks each individual bit of the tags' responses. When the reader detects collision bits through a hybrid Manchester coding system, it updates the query string according to the result of each bit position.[15] If all responses have values “0” or “1” for a particular bit position, no collisions occur at this bit. Moreover, 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, then the bit string at this position is indicated as “*”. In this case, the length of the collision bits C is incremented by 1. When the length of the collision bits reaches the maximum value C_{max}, the reader sends an ACK signal to tags to stop them from transmitting their remaining IDs. After the query string is updated, the reader considers the following three possible situations: 1) multiple “*” exist in the updated query string, 2) a single “*” is present in the updated query string, and 3) no “*” exists in the updated query string. In the first case, the reader saves a new prefix with the updated query string and the value of C into the LIFO stack. In the second case, the reader replaces “*” with “0” and

"1" and marks them as two successfully identified tags without extra queries. In the third case, the reader recognizes the tag ID and successfully identifies a tag. The reader then obtains the subsequent query string and the value of C from the stack for the next query cycle and repeats the identification process until the LIFO stack is empty. The reader broadcasts the query string to all tags in each query cycle. Each bit position of the query string may include "0," "1," or "*". If any tag matches the bit positions of values "0" and "1" in the query string, then it must respond with their values for the position corresponding to "*." For example, if a reader sends the query string "0*1**" to all tags, then the tags that contain "0" and "1" as the first and third bits, respectively, respond to the reader by transmitting their values of the second, fourth, and fifth bits. The reader also allocates 2^{Cmax} time slots for tags to respond to. The 2^{Cmax} time slots are represented by the values of the C_{max} bit from 0 to $(2^{Cmax} - 1)$. The values correspond to the positions of "*" in the query string. For example, if the reader sends the query string "0*1*" to tags, then the reader allocates four time slots that are indicated as "00," "01," "10," and "11" for the tags to respond to. Then, the tags whose first and third bits are "0" and "1," respectively, transmit the remaining bits of their IDs in the corresponding time slot set according to their second and fourth bits. The identification procedure of the proposed scheme is as follows:

1) REQUEST: A reader sends prefix-inquiring bits of length n to tags, including the wild mask "*".
2) RESPONSE: Tags in the interrogation zone of the reader respond to the reader with their tag IDs if the first n bits of the tag IDs match the bit positions of values "0" and "1" in the inquiring bits.
   - When the tags respond to the reader with their IDs, they select one of the 2^C time slots depending on their values at the positions corresponding to "*" in the prefix query string.
   - The tags send their IDs from the $(n + 1)$th bit until the time that the ACK signal sent from the reader is received.
3) DECISION: On the basis of whether a collision has occurred and the length of collision bits C', the reader decides whether to continue the procedure under the following conditions.
   - If all responses have values "0" or "1" for a particular bit position, then the reader updates the corresponding bit position of the query string with the value "0" or "1".
   - If a collision occurs at a particular bit position, then the corresponding bit position of the query string is indicated as "*" and the value of C' is incremented by 1.
   - If a collision occurs and $C' = C_{max}$, then the reader sends an ACK signal to all tags and saves the updated query string as a new prefix into the LIFO stack.
   - If no collision occurs, then the reader identifies a tag and sends an ACK signal after receiving the last bit.
   - If no response is obtained from any tag, then the reader sends an ACK signal after the receiving time of the first bit has passed.
4) Perform the above steps until the LIFO stack is empty.

An example is presented in Table 3 to understand the proposed algorithm better. Table 3 presents the operation of the proposed scheme for the eight tags presented in Table 1. The identification process is as follows. First, the reader sends the request command with an empty prefix to all tags in the first query cycle (Table 3). In this case, all tags respond to this request command, and the reader begins to track the received bits to verify whether the received bits

Table 3
Detailed query cycle of the EDSCT scheme for identifying eight tags.

Query cycle	Prefix string	Time slots	Tag response	Reader update string	Results	Prefix stack
1	(‘’,0)		‘0*0*’	‘0*0*’	Collision	(‘0*0*’,2)
2	(‘0*0*’,2)	‘00’	‘001010’	‘0000001010’	Identified	ϕ
		‘01’	‘0*1*’	‘00010*1*’	Collision	(‘00010*1*’,2)
		‘10’	‘010011’	‘0100010011’	Identified	(‘00010*1*’,2)
		‘11’	‘*0*’	‘0101*0*’	Collision	(‘00010*1*’,2), (‘0101*0*’,2)
3	(‘0101*0*’,2)	‘00’	‘101’	‘0101000101’	Identified	(‘00010*1*’,2)
		‘01’	ϕ	ϕ	Idle	(‘00010*1*’,2)
		‘10’	‘111’	‘0101100111’	Identified	(‘00010*1*’,2)
		‘11’	‘001’	‘0101101001’	Identified	(‘00010*1*’,2)
4	(‘00010*1*’,2)	‘00’	ϕ	ϕ	Idle	ϕ
		‘01’	‘01’	‘0001001101’	Identified	ϕ
		‘10’	‘11’	‘0001011011’	Identified	ϕ
		‘11’	‘01’	‘0001011101’	Identified	ϕ

*Note: The ϕ symbols presented in the Tag response, Reader update string, and Prefix stack columns indicate no response from tags, an empty string, and an empty stack, respectively.

are collided bits. Because the reader receives the second collided bit at the fourth bit, the reader sends an ACK signal to stop the ID transmissions from the tags. At this moment, the reader updates the query string as “0*0*” and stores the updated query string into the stack. Subsequently, the reader obtains the first query string in the stack, which is “0*0*”, and sends it to all tags along with four time slots, that is, “00”, “01”, “10”, and “11”. This means that the tags that match prefix strings “0000”, “0001”, “0100”, and “0101” respond in slots “00”, “01”, “10”, and “11”, respectively. In this case, tag A responds in slot “00”, tags B, C, and D respond in slot “01”, tag E responds in slot “10”, and tags F, G, and H respond in slot “11” (query cycle 2). In query cycle 2, tags A and E are identified, and two updated query strings, “00010*0*” and “0101*0*”, are saved into the stack. The identification process proceeds until the stack is empty. In this example, four query cycles are executed to identify eight tags.

Table 3 reveals that the EDSCT scheme uses 19 bits for prefixes and 38 bits for responses. Thus, a total overhead of 57 bits is used to identify eight tags. Compared with the DSCT scheme, the EDSCT scheme can effectively reduce the number of query cycles required in the tag identification process.

4. Performance Evaluation

To evaluate the performance of the proposed approach, we implemented both DSCT and EDSCT schemes with three different values of C_{max} (2, 3, and 4). We compared the performance of both the DSCT and EDSCT schemes with the BSCTTA and IDSCTTA schemes. The performance of the IDSCTTA scheme improves as the value of C_{max} increases. Therefore, in our experiments, we only evaluated the performance of the IDSCTTA when C_{max} = 4. The various resulting algorithms are indicated as BSCTTA, IDSCTTA-4, DSCT-2, DSCT-3, DSCT-4, EDSCT-2, EDSCT-3, and EDSCT-4. We conducted a set of simulation experiments to evaluate the proposed algorithm. All experiments were performed on a computer equipped

with a 3 GHz central processing unit and an 8 GB memory in C# on the .NET framework. Every experiment was repeated 50 times, and the recorded data were averaged over these runs to obtain the final results.

The simulation environment is in accordance with the EPCglobal C1 G2 standard as follows.[23] We employed an RFID system that has one reader and N tags within the reading range, where N = 5000, 10000, ..., and 50000. All tags have 96-bit-long IDs. We also considered two different tag ID distributions: uniform random and sequential distributions. The tag IDs in the sequential distribution are in groups and consecutive. The maximum group size g was set as 10, 20, or 50% of the number of tags to be identified. The rate of data communication in the transmission channels was set to 80 kbps. For convenience, we considered a noise-free channel between the reader and tags and ignored the propagation delay of the signal because all the aforementioned algorithms would be equally influenced by the propagation delay.

The conducted simulations focused on determining the performance of algorithms for different numbers of tags in terms of the average number of queries required, average number of transmission bits required, delay time, and system efficiency. The former two performance metrics were measured by conducting the identification of one tag. System efficiency was measured using $S = N/S_{tot}$, where S_{tot} is the total number of slots.

4.1 Average number of queries required versus number of tags

The results pertaining to the average number of queries of schemes required for one tag identification are presented in Figs. 1–4 for both uniform and group distributions. Note that the performance of the IDSCTTA scheme in terms of the average number of queries required is the same as that of our DSCT scheme. Therefore, the performance of the IDSCTTA scheme is omitted from Figs. 1–4. Figure 1 shows that both dynamic slot schemes significantly outperform the BSCTTA in terms of the average number of queries required to identify all tags when IDs are uniformly distributed. The performance of all schemes in terms of the

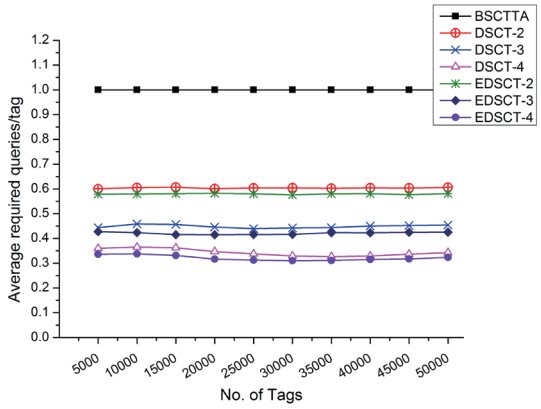

Fig. 1. (Color online) Average number of queries required with uniform distribution.

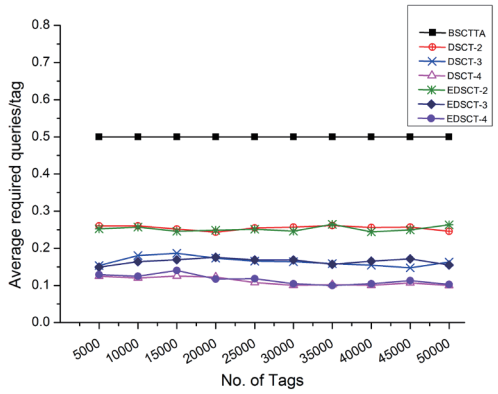

Fig. 2. (Color online) Average number of queries required with group distribution for a g value of 10%.

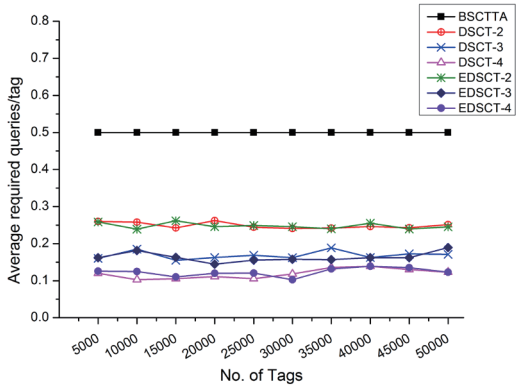

Fig. 3. (Color online) Average number of queries required with group distribution for a g value of 20%.

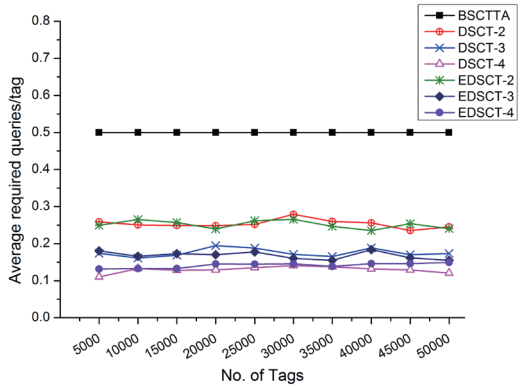

Fig. 4. (Color online) Average number of queries required with group distribution for a g value of 50%.

average number of queries required is almost flat irrespective of the value of N. The maximum number of allowed collision bits has a strong influence on the performance of the DSCT and EDSCT schemes. The performance of these schemes in terms of the average number of queries required improves as C_{max} increases. The average numbers of queries required in the DSCT schemes are approximately 0.61, 0.44, and 0.35 for C_{max} values of 2, 3, and 4, respectively. The average numbers of queries required in the EDSCT schemes are approximately 0.58, 0.42, and 0.33 for C_{max} values of 2, 3, and 4, respectively. The EDSCT schemes with C_{max} values of 2, 3, and 4 outperform the corresponding DSCT schemes by approximately 4.17, 5.32, and 6.42%, respectively. The reason for these results is clear. The number of query cycles increases proportionally with the number of tags, thus resulting in the average number of queries required almost remaining unchanged. Both the DSCT and EDSCT schemes allocate a higher number of slots than that used in the BSCTTA protocol. Thus, the average number of queries required for dynamic-slots-based protocols is lower. Moreover, the EDSCT schemes use more slots than the DSCT schemes during each query. Thus, the EDSCT schemes with different C_{max} values use fewer query cycles to complete the identification than that used by the corresponding DSCT scheme.

Figures 2–4 present the behavior evaluation of all protocols when the tag IDs are not uniformly distributed. From these figures, we deduce that the average number of queries of all protocols remains almost unchanged with the value of N. The average number of queries required for all protocols without uniformly distributed tag IDs is less than half of that for all protocols with tag IDs in a uniform distribution. The reason for this is also clear. Consider the identification process of a set of tags. If tag IDs have the longest common prefixes and only differ in a few least significant bits, then the tags that do not have uniformly distributed IDs become sibling leaves in the identification tree and result in fewer collisions than those that have uniformly distributed IDs. Thus, all protocols require fewer query cycles when tags do not have uniformly distributed IDs than when they have uniformly distributed IDs. Moreover, all protocols exhibited similar performance in terms of the average number of queries required irrespective of the group size g. The DSCT and EDSCT schemes outperform the BSCTTA

scheme. The maximum number of allowed collision bits C_{max} has a greater influence on the performance of the DSCT and EDSCT schemes. The performance of these schemes in terms of the average number of queries required improves as C_{max} increases. The performance of DSCT and EDSCT improves with increasing maximum number of allowed collision bits and decreasing number of collision slots. However, the behavior of the EDSCT schemes is similar to that of the DSCT schemes when the tag IDs have a group distribution. Collisions occur consecutively because the tag IDs differ in terms of the last few least significant bits, which results in the same collision bits in both the DSCT and EDSCT protocols.

4.2 Average transmission versus number of tags

Our next experiment evaluated the influence of the number of tags on the number of transmission bits required to complete tag identification for the BSCTTA, IDSCTTA, DSCT, and EDSCT schemes. We measured the performance of these protocols by calculating the average number of transmission bits required to complete the identification of one tag. These results are shown in Figs. 5–8 for both uniform and group distributions. Figure 5 reveals that the average number of transmission bits required to identify one tag in each algorithm decreases slightly as the number of tags increases, except in the BSCTTA protocol. In the BSCTTA protocol, the average number of transmission bits required to identify one tag is fixed regardless of the number of tags. This result is observed in BSCTTA because each query cycle comprises only two slots for tags to respond and each identified tag can save only one bit for transmitting its ID to the reader. In other protocols, multiple collision bits occur during the query process. Thus, each identified tag requires fewer bits to respond than the length of its ID. The DSCT scheme outperforms the IDSCTTA scheme when C_{max} = 4 because the communication overhead of the IDSCTTA scheme increases with the number of collisions. The EDSCT schemes outperform the IDSCTTA and DSCT schemes regardless of the C_{max} value because the EDSCT schemes always generate a higher number of time slots to resolve collisions than the

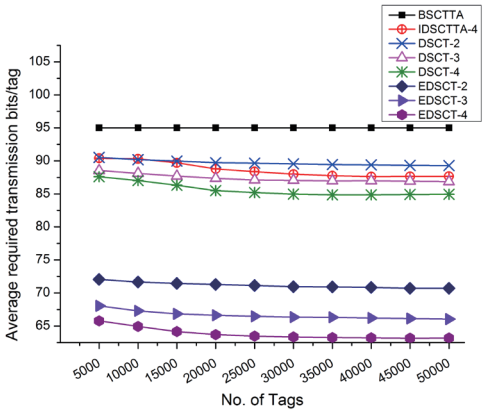

Fig. 5. (Color online) Average number of transmission bits required with uniform distribution.

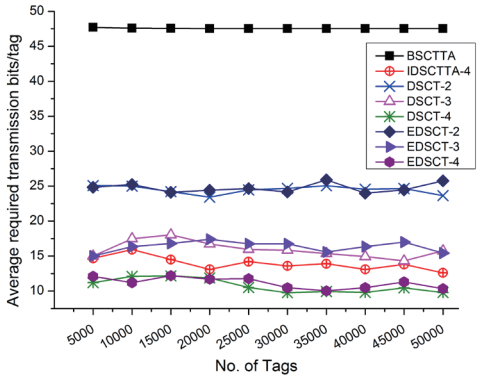

Fig. 6. (Color online) Average number of transmission bits required with group distribution and g = 10%.

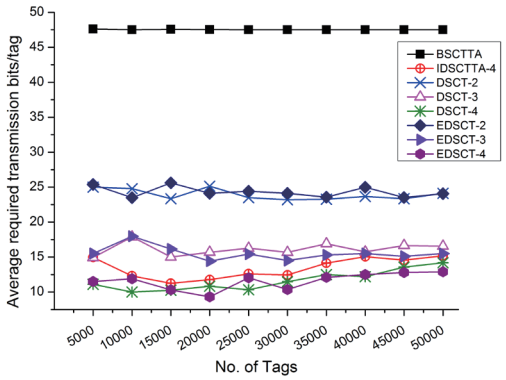

Fig. 7. (Color online) Average number of transmission bits required with group distribution and g = 20%.

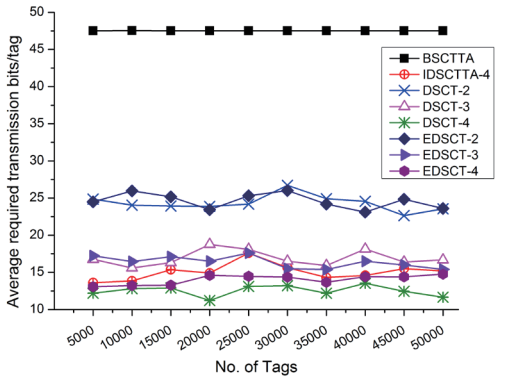

Fig. 8. (Color online) Average number of transmission bits required with group distribution and g = 50%.

DSCT schemes. As a result, the EDSCT schemes perform approximately 20.8 to 25.6% better than the DSCT schemes.

Figures 6–8 reveal that the average number of transmission bits required for each tag in all schemes when the group distribution is used is less than half of that required when uniformly distributed IDs are used. In the group distribution, because the tag IDs differ in terms of the last few least significant bits, each query cycle recognizes more tags with the same query prefix. Moreover, all protocols exhibit similar performance in terms of the average number of transmission bits required irrespective of the group size g. The DSCT and EDSCT schemes outperform the BSCTTA scheme. The DSCT and EDSCT schemes also outperform the IDSCTTA scheme when C_{max} = 4. The performance of these schemes in terms of the average number of transmission bits required improves as C_{max} increases because the number of collision slots decreases. The EDSCT schemes exhibit similar behaviors to the DSCT schemes because collision bits occur consecutively when the tag IDs have a group distribution.

4.3 Delay time versus number of tags

We next experimentally evaluated the influence of the number of tags on the total time required to complete tag identification through the BSCTTA, IDSCTTA, DSCT, and EDSCT schemes. We measured the performance by calculating the time required to complete the communication of all required transmission bits between the reader and tags. These results are shown in Figs. 9–12 for both uniform and group distributions. Figure 9 reveals that as the number of tags increases, the EDSCT schemes significantly outperform the BSCTTA, IDSCTTA, and DSCT schemes. The reason for this is clear. The number of collision slots is lower in the EDSCT schemes than in the BSCTTA, IDSCTTA, and DSCT schemes. As a result, the number of query cycles and total number of transmission bits in the EDSCT schemes are smaller than those in the BSCTTA, IDSCTTA, and DSCT schemes. Moreover, the delay time of the EDSCT schemes decreases as C_{max} increases because the number of identified tags increases in each query cycle.

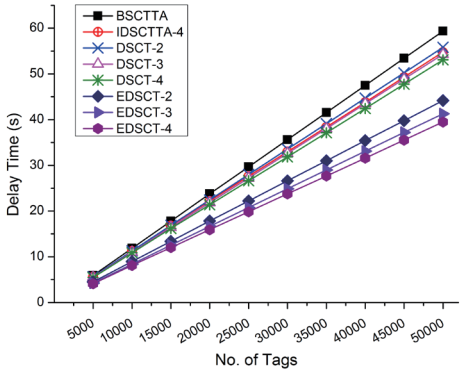

Fig. 9. (Color online) Time required for complete tag identification with uniform distribution.

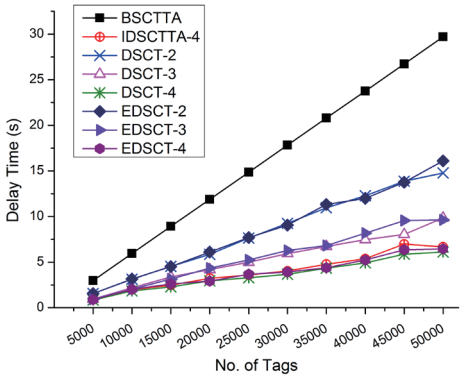

Fig. 10. (Color online) Time required for complete tag identification with group distribution and g = 10%.

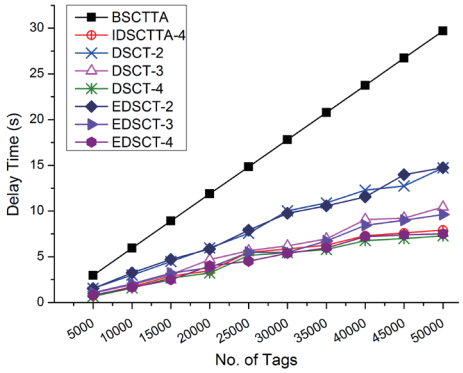

Fig. 11. (Color online) Time required for complete tag identification with group distribution and g = 20%.

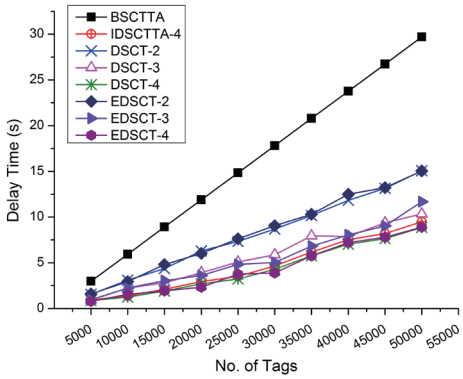

Fig. 12. (Color online) Time required for complete tag identification with group distribution and g = 50%.

Figures 10–12 reveal that the delay time observed in all schemes when the group distribution is used is less than half of that when uniformly distributed IDs are used. Similar reasoning to that of the previous experiment can be used to explain this. That is, because the tag IDs differ in the last few least significant bits in the group distribution, each query cycle can recognize more tags with the same query prefix. Moreover, both the DSCT and EDSCT schemes exhibit similar performance in terms of the average number of transmission bits required irrespective of the group size g. Furthermore, the DSCT and EDSCT schemes outperform the BSCTTA scheme and also the IDSCTTA scheme when C_{max} = 4. However, the delay time of dynamic-slots-based schemes increases as the group size g increases. In the group distribution, the number of different bits in the least significant bits of tag IDs increases as the group size g increases. Thus, the number of collision slots and the total number of transmission bits increase.

4.4 System efficiency versus number of tags

Results pertaining to the system efficiency of the protocols are shown in Figs. 13–16 for both uniform and group distributions. Figure 13 reveals that each of the compared approaches exhibits similar system efficiency as the number of tags increases. Moreover, the DSCT and EDSCT protocols perform significantly worse in terms of the slot system efficiency than the BSCTTA and IDSCTTA schemes. The rationale behind these results is clear. Because the numbers of identification slots and total slots increase as the number of tags increases, the slot system efficiency is almost flat. In the DSCT and EDSCT schemes, each query cycle allocates at least two time slots for tags to respond. Some of the slots may identify the tag when no collision occurs. However, the rest of the slots may receive multiple responses from tags or no response. In this case, the slot may collide or be idle. Therefore, the slot utilization in the DSCT and EDSCT schemes is poorer than that in the BSCTTA scheme, in which only two slots are allocated for each query, and the IDSCTTA scheme, in which no idle slot is allocated for

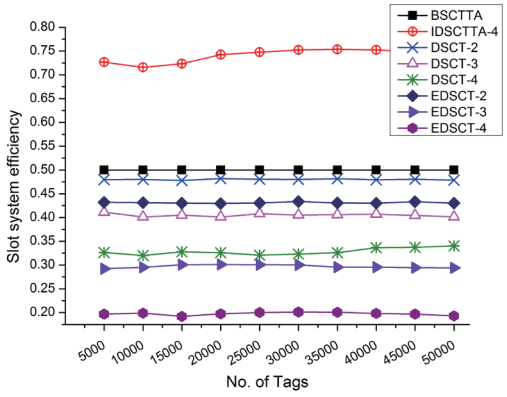

Fig. 13. (Color online) Comparison of slot system efficiency for uniform distribution.

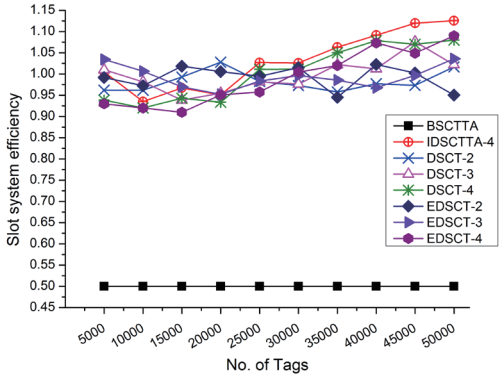

Fig. 14. (Color online) Comparison of slot system efficiency for group distribution and a g value of 10%.

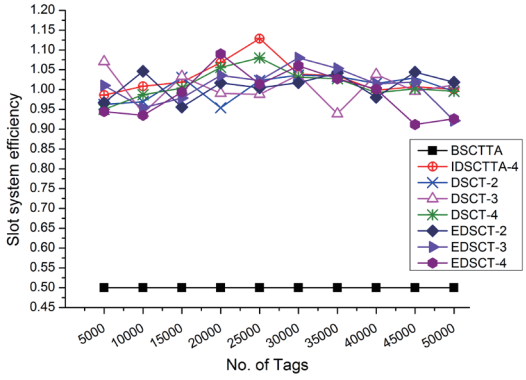

Fig. 15. (Color online) Comparison of slot system efficiency for group distribution and a g value of 20%.

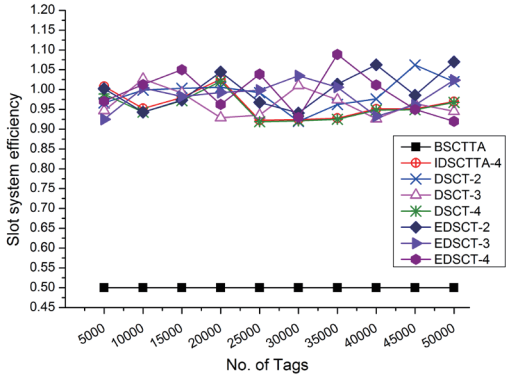

Fig. 16. (Color online) Comparison of slot system efficiency for group distribution and a g value of 50%.

each query. The slot utilization in the EDSCT schemes is poorer than that in the DSCT schemes at the same C_{max} value because the EDSCT schemes always allocate a higher number of slots during the identification process.

Figures 14–16 reveal that the DSCT and EDSCT schemes significantly outperform the BSCTTA scheme in terms of the slot system efficiency when the tag IDs have a group distribution irrespective of the group size *g*. The rationale for this is clear. In a group distribution, the tags are sibling leaves in the identification tree and cause many collision slots during each query prefix. Thus, the number of idle slots is reduced, which results in both the DSCT and EDSCT schemes having a superior slot system efficiency than that in the case of uniformly distributed tag IDs. Furthermore, both the DSCT and EDSCT schemes exhibit similar slot system efficiencies (approximately 90 to 110%) regardless of the value of *N*. Note that the IDSCTTA scheme also exhibits a similar slot system efficiency to the DSCT scheme because the number of idle slots decreases considerably when the tag IDs have a group distribution.

5. Conclusions

The rapid technical progress and widespread application of RFID sensing techniques have produced novel solutions in different fields of applications, which are very promising for future IoT-rich sensing applications. However, developing a highly efficient tag identification process for large-scale RFID systems is a crucial and very challenging task. Many collisions may occur during the tag identification process due to the nature of large-scale RFID systems. Identification protocols, such as the BSCTTA, can reduce the numbers of query cycles required and occurring collisions by allocating two slots for tags to respond on the basis of collision details obtained in the previous query cycle. In this study, we extended the previous DSCT scheme, in which the number of slots allocated in each query cycle is dynamic and depends on the number of collision bits detected in the previous query cycle, to manage both consecutive and nonconsecutive collision situations. To evaluate the performance of the proposed scheme, we conducted a series of experiments on both uniform and group distributions. Simulation results reveal that the proposed EDSCT scheme significantly outperforms the BSCTTA scheme in terms of the average number of queries required, the average number of transmission bits required, and the delay time regardless of the distribution of tag IDs. The proposed scheme also outperforms the BSCTTA scheme in terms of the slot system efficiency when tag IDs exhibit a group distribution. The proposed EDSCT scheme exhibits better performance than the DSCT scheme in terms of the average number of queries required, the average number of transmission bits required, the delay time, and the slot system efficiency when tag IDs exhibit a uniform distribution. Furthermore, both the DSCT and EDSCT schemes outperform the IDSCTTA scheme in terms of the average number of transmission bits required and the delay time regardless of the distribution of tag IDs. For a group distribution, the EDSCT scheme requires similar average numbers of queries and transmission bits and has a similar delay time and slot system efficiency to those in the DSCT scheme. Therefore, the EDSCT scheme has better performance than the BSCTTA, IDSCTTA, and DSCT schemes.

References

1 A. Shirehjini, A. Yassine, and S. Shirmohammadi: IEEE Trans. Inf. Technol. Biomed. **16** (2012) 1058. https://doi.org/10.1109/TITB.2012.2204896
2 J. Wang, D. Ni, and K. Li: Sensors **14** (2014) 4225. https://doi.org/10.3390/s140304225
3 X. Zhu, S. K. Mukhopadhyay, and H. Kuraya: J. Eng. Tech. Manage. **29** (2012) 152. https://doi.org/10.1016/j.jengtecman.2011.09.011
4 A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios: IEEE Commun. Surv. Tutorials **19** (2016) 1327. https://doi.org/10.1109/COMST.2016.2632427
5 C. Occhiuzzi, S. Caizzone, and G. Marrocco: IEEE Antennas Propag. Mag. **55** (2013) 14. https://doi.org/10.1109/MAP.2013.6781700
6 A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash: IEEE Commun. Surv. Tutorials **17** (2015) 2347. https://doi.org/10.1109/COMST.2015.2444095
7 L. Cui, Z. Zhang, N. Gao, Z. Meng, and Z. Li: Sensors **19** (2019) 4012. https://doi.org/10.3390/s19184012
8 J. Park, M. Chung, and T. J. Lee: IEEE Commun. Lett. **11** (2007) 452. https://doi.org/10.1109/LCOMM.2007.061581
9 H. Wu, Y. Zeng, J. Feng, and Y. Gu: IEEE Trans. Parallel Distrib. Syst. **24** (2013) 19. https://doi.org/10.1109/TPDS.2012.120
10 C. Law, K. Lee, and K. Y. Siu: Proc. 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM, 2000) 75−84. https://doi.org/10.1145/345848.345865
11 H. S. Choi, J. R. Cha, and J. H. Kim: Proc. 5th Pacific Rim Conf. Multimedia 2004, Eds. K. Aizawa, Y. Nakamura, and S. Satoh (Springer, Heidelberg, 2004) 696–703. https://doi.org/10.1007/978-3-540-30542-2_86
12 J. Myung, W. Lee, and J. Srivastava: IEEE Commun. Lett. **10** (2006) 144. https://doi.org/10.1109/LCOMM.2006.03031
13 N. Bhandari: Master Thesis (2006) Indian Institute of Technology Bombay, https://www.it.iitb.ac.in/~sri/students/naval-thesis.pdf (accessed January 2020).
14 F. Zhou, D. Jin, C. Huang, and M. Hao: Proc. 5th Int. Conf. ASIC (ICASIC, 2003) 1213−1217. https://doi.org/10.1109/ICASIC.2003.1277432
15 H. S. Gou, H. C. Jeong, and Y. H. Yoo: Proc. 6th Int. Conf. Wireless and Mobile Computing, Networking and Communications (WiMob, 2010) 421−428. https://doi.org/10.1109/WIMOB.2010.5645031
16 X. Jia, Q. Feng, and L. Yu: IEEE Trans. Commun. **60** (2012) 2285. https://doi.org/10.1109/TCOMM.2012.051512.110448
17 Y. C. Lai, L. Y. Hsiao, and B. S. Lin: IEEE/ACM Trans. Networking **23** (2015) 255. https://doi.org/10.1109/TNET.2013.2205839
18 J. Su, D. Hong, J. Tang, and H. Chen: IEICE Trans. Commun. **E99-B** (2016) 465. https://doi.org/10.1587/transcom.2015EBP3235
19 H. Landaluce, A. Perallos, E. Onieva, L. Arjona, and L. Bengtsson: IEEE Trans. Wireless Commun. **15** (2016) 4234. https://doi.org/10.1109/TWC.2016.2537800
20 J. H. Choi, D. Lee, H. Jeon, J. Cha, and H. Lee: Proc. IEEE Int. Conf. Communications (ICC, 2007) 3853−3858. https://doi.org/10.1109/ICC.2007.635
21 C. K. Liang and H. M. Lin: Proc. 9th Int. Conf. Ubiquitous Intelligence and Computing and 9th Int. Conf. Autonomic and Trusted Computing (UIC-ATC, 2012) 272−277. https://doi.org/10.1109/UIC-ATC.2012.32
22 Y. Huang and X. Chen: Proc. 10th Int. Conf. Wireless Communications, Networking and Mobile Computing (WiCOM, 2014) 662−669. https://doi.org/10.1049/ic.2014.0176
23 EPCglobal: EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID, https://www.gs1.org/sites/default/files/docs/epc/Gen2_Protocol_Standard.pdf (accessed January 2020).

About the Authors

Yu-Hsiung Lin received his M.S. degree from the Department of Electrical Engineering at Chung Hua University, Hsinchu, Taiwan, Republic of China, in 1996 and his B.S. degree from the Department of Computer Science at National Chiao Tung University, Hsinchu, Taiwan, Republic of China, in 1990. He is a lecturer at the Department of Electrical Engineering, Chung Hua University. His research interests include mobile application development, color science, and RFID systems.

Chiu-Kuo Liang received his Ph.D. degree in computer science from National Tsing Hua University, Taiwan, in 1990. He is currently an associate professor at the Department of Computer Science and Information Engineering, Chung Hua University. His research interests include wireless mobile computing, sensor networks, RFID systems, Internet of Things, and parallel processing.