Computer Communications 88 (2016) 73–83

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Efficient tag reading protocol for large-scale RFID systems with pre-reading

Shuen-Chih Tsai*, Yu-Min Hu, Chen-Hsun Chai, Jung-Shian Li

Department of Electrical Engineering, Institute of Computer and Communication Engineering, National Cheng Kung University, University Road, Tainan 701, Taiwan, ROC

ARTICLE INFO

Article history:
Received 3 September 2015
Revised 19 March 2016
Accepted 23 April 2016
Available online 25 April 2016

Keywords:
RFID tag collection
STT
Blocking protocol
Missing-tag problem

ABSTRACT

In large-scale RFID systems, collecting all of the tag IDs is a time-consuming process. A protocol designated as Smart Trend-Traversal (STT) has been proposed to reduce collisions during the tag collection process and to dynamically construct the query strings used to interrogate the tags. In general, if the tag ID information is known to the reader from a previous tag collection round, the efficiency of the current round can be significantly improved. Various protocols have been proposed for scanning a known tag set based on the use of a hash function. Accordingly, the present study proposes an Enhanced STT scheme based on a blocking protocol and a Distributed Record Tag-Check (DRTC) mechanism. Compared to the conventional STT scheme, the proposed protocol adaptively adjusts the length of the query string depending on the response received to the previous query. Moreover, in the DRTC mechanism, the tags determine their transmission slot frame directly without the assistance of the reader, and thus the overall overhead of the tag-collection process is reduced. The simulation results show that the Enhanced STT scheme reduces the total number of queries required to collect the entire tag set compared to the conventional STT method. Moreover, the proposed DRTC mechanism yields an effective reduction in the total number of frame slots compared to existing protocols such as TPP/CSTR and ECRB.

1. Introduction

Radio frequency identification (RFID) technology [1] has many important applications nowadays , including item identification, automatic inventory and asset management, payment, and so on [9,24,37]. RFID systems have many advantageous features, including a relatively low deployment cost, an ability to operate in harsh environments, the absence of batteries or any form of external energy source, and so forth. However, in implementing RFID systems, two problems arise, namely the RFID tag-collection problem [16] and the RFID missing-tag event problem [21,29]. Various protocols have been proposed for improving the efficiency of the tag ID collection process. Moreover, various methods have been developed for solving the missing-tag problem, i.e., detecting the missing tags among a set of known tags. However, in previous study, they usually either process the tag identification collection or detect the missing tags independently. If the process of tag identification finishes, the process of missing-tag detection will start separately. In general, the system spends some time on switching these two processes. Our proposed protocol will switch the processes smoothly. We want to solve these two problems together and let our proposed scheme more efficient.

The present study proposes an Enhanced Smart Trend-Traversal (Enhanced-STT) scheme based on the STT protocol [18] and together with a blocking mechanism [35] and a Distributed Record Tag-Check (DRTC) mechanism. The conventional STT protocol uses a depth-first-search method to construct the query traverse path. Moreover, when collisions occur, the number of bits appended to the query string used to interrogate the tags is dynamically adjusted in accordance with the tag ID distribution or the tag population. However, in the Enhanced STT scheme, the query length is adjusted dynamically in accordance with the response received from the previous query in order to accelerate the tree traversal process. In addition, in the DRTC mechanism, the tags determine their transmission frame-slots directly from an inspection of the query length. In other words, the reader is not required to transmit slot information to them, and thus the tag collection overhead is reduced. Compared to traditional binary tree-based protocols [34], DRTC has a lower system requirement (i.e., a fewer number of counters per tag), but retains an excellent performance.

The simulation results confirm that the Enhanced STT protocol reduces the total number of queries required to collect the entire tag set compared to STT. Moreover, the Enhanced STT scheme

* Corresponding author. Tel.: +886 6 2757575x62400.
E-mail address: pandaorz@gmail.com (S.-C. Tsai).

http://dx.doi.org/10.1016/j.comcom.2016.04.019
0140-3664/

reduces the number of collisions in the tag-collection process, and therefore reduces both the total tag identification time and the energy consumption of the active tags. The efficiency of the Enhanced STT scheme is further evaluated in terms of both the System Efficiency (*SE*) and the Time System Efficiency (*Time_SE*). It is shown that the *Time_SE* of Enhanced STT is superior to that of STT and other Query Tree-based (QT-based) protocols. In addition, it is shown that the DRTC mechanism provides an efficient approach for checking the IDs of staying tags and reduces the total number of frame slots required to collect all of the tag IDs compared to existing protocols such as Three-Phase Protocol with Collision Sensitive Tag Removal (TPP/CSTR) [21] and Enhanced Couple-Resolution Blocking (ECRB) [35].

The remainder of this paper is organized as follows. Section 2 reviews the related work in the RFID tag identification field. Section 3 introduces the Enhanced STT scheme and DRTC mechanisms proposed in this study. Section 4 presents and discusses the simulation results Finally, Section 5 provides some brief concluding remarks and indicates the intended direction of future research.

2. Related works

Research in the RFID field focuses predominantly on two main problems, namely the RFID tag anti-collision problem and the RFID missing-tag event problem. The former problem involves improving the efficiency of the tag collection problem by avoiding collisions [16]. The later problem involves scanning a large group of known tags and identifying any tags which are missing in the current scanning round [4]. The present study proposes an approach for reducing the tag identification time by extending the conventional STT algorithm. STT is a QT-based anti-collision protocol. Thus, the following sub-section introduces previous work relating to QT-based protocols. Section 2.2 then reviews existing proposals for dealing with the missing tag event in the second round (and on) of the tag-collection process.

2.1. RFID tag anti-collision protocols

Existing RFID tag anti-collision protocols generally adopt either a tree-based approach [11,15,23,25,26,34–36] or an Aloha-based approach [3,7,8,14,19,20,22,30–32]. Because of scalability in real life [5], our study adopts tree-based approaches. Among protocols adopting the former method, the Query Tree (QT) algorithm [2,6], and Binary Tree (BT) algorithm [34] are among the most common. In the QT algorithm, the RFID reader uses the tag ID prefix to perform the tag-collection process. By contrast, the BT algorithm identifies the tags by using random binary numbers to partition the RFID tags into small groups. Meanwhile, in Aloha-based protocols, the reader first estimates the number of tags within its communication range and then assigns these tags an appropriate number of frame slots such that they can return their IDs without contention [28]. We adopt the mechanism without estimating the number of tags, because it is more easy to be deployed in real life.

Irrespective of the anti-collision protocol employed, it is necessary to deal with two different types of tags, namely staying tags and arriving tags. Staying tags are tags identified by the reader in a previous tag-collection round which are also present in the current round. By contrast, arriving tags are tags which appear in the reader's communication range for the first time in the present round. In other words, they are unknown to the reader prior to the current round. As described above, in attempting to improve the efficiency of the tag-collection process, the present study extends the conventional QT-based STT protocol. Thus, the following discussions briefly review the original QT algorithm and its variants.

2.1.1. QT protocol and variants

In the QT protocol, the tag set is split using the tag ID prefix in order to construct a query tree. Having constructed this tree, the reader then broadcasts tag ID queries in a breadth-first manner. The reader maintains a queue (Q) of query strings and initializes this queue with two 1-bit strings, i.e., 0 and 1, at the beginning of each tag-collection round. The reader then de-queues a query string from Q in order to interrogate the tags within its communication range. Any tags having a prefix which matches the query transmit their IDs to the reader. If the tag IDs arrives at the reader in different time slots, they can be successfully recognized. However, if multiple tags respond simultaneously, the reader detects a collision event and splits the original query $b_1b_2\ldots b_{lc}$ into two sub-tree queries where $b_i \in \{0, 1\}$, i.e., $b_1b_2\ldots b_{lc}0$ and $b_1b_2\ldots b_{lc}1$, which it then enqueues to the end of its queue. In the event that the reader detects just one tag response or no tag response, referred to as a single-tag-response query and an idle query, respectively, it concludes that query $b_1b_2\ldots b_{lc}$ is the end of the current sub-tree, and labels the query string "end" accordingly. The reader then dequeues the next query string from Q and repeats the interrogation process. The process is repeated iteratively in this way until all of the query strings in Q are labelled as "end".

In the QT-based Adaptive Query Splitting (AQS) protocol [13], the single-tag-response and idle queries obtained in the previous tag-collection round are stored by the reader in a candidate queue (CQ). At beginning of each tag-collection round, the queries in CQ are copied to Q and are then sequentially dequeued in order to identify the staying tags and arriving tags, respectively. The Couple-Resolution Blocking (CRB) protocol [35] extends AQS by introducing a blocking mechanism. In the proposed approach, the reader "mutes" arriving tags as it is identifying the staying tags in order to prevent collisions between them. CRB distinguishes between the two types of tags by allowing the reader and all the tags to store both the last reader ID and the frame number. CRB divides the tag-collision process into two phases, namely a staying tag identification phase (Phase 1) and an arriving tag identification phase (Phase 2). In Phase 1, the reader dequeues two queries from Q and transmits a concatenated query including both queries. Since the arriving tags are muted in Phase 1, the reader interprets the occurrence of a collision event as meaning that the two tags are still present (i.e., they are staying tags). Importantly, this "2-collision" concept yields a significant reduction in the time required to identify the staying tags compared to the original AQS protocol. Enhanced CRB combines two similar prefixes into one query. Thus, sending this query can further reduce the bits sent by reader.

2.1.2. Smart trend-traversal protocol

Various protocols based on QT have been proposed in which the reader broadcasts queries in a depth-first rather than breadth-first manner [10,18,33]. In such protocols, when a query $b_1b_2\ldots b_{lc}$ results in a collision, the reader infers that the current query is at a higher bit-level in the query tree, and thus the next query is set to $b_1b_2\ldots b_{lc}0$ such that the reader traverses down to the child node in the query tree. In the event of a single-tag-response, the next query is set as $(b_1b_2\ldots b_{lc})+1$ and traverses horizontally to the node located immediately to the right of the child node in the tree. Finally, if an idle query occurs, the next query is set to $b_1b_2\ldots b_{lc-1}1$ if $b_{lc}=0$ (i.e., b_{lc} is a left child node), or $b_1b_2\ldots b_{lc-1}$ if $b_{lc}=1$ (i.e., b_{lc} is a right child node), such that the query traverses in the right upper direction and converge to the next prefix node. Fig. 1 presents an illustrative example of the STT protocol. There are 5 tags in the query tree. Each node represents the status of a query, including collision, single-tag-response and idle. Note that the numbers on the graph show the sequence step in the STT process.

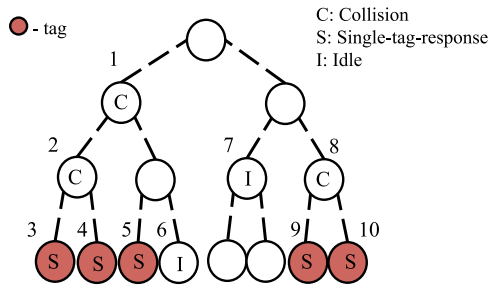

Fig. 1. Illustrative example of STT protocol.

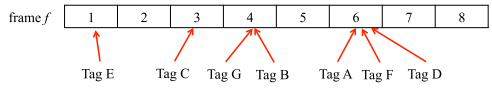

Fig. 2. Mapping of tags to frame slots using hash function.

2.2. RFID missing-tag event

In many RFID applications, it is necessary to monitor a very large number of tags. Furthermore, the tag IDs recorded at the reader or in a linked database require constant maintenance as new tags enter the network or existing tags leave. For large-scale systems, scanning the RFID tag set repeatedly using the original anti-collision protocol is highly inefficient; particularly if most of the tags are staying tags and their IDs are therefore already known to the reader. In such a situation, a more intuitive approach is simply to identify the tags which were previously known to the reader but no longer appear in the current round (i.e., the so-called RFID missing-tag problem [21,29]).

To monitor the missing tags, a reader can use a hash function to map the tags to a frame slot. As shown in Fig. 2, each tag is pseudo-randomly mapped to a frame slot. In performing the roll call, the reader broadcasts a request $< r, f >$, where r is a random number and f is the frame size. Therefore, all tags know the frame size. While receiving the request, each tag calculates the slot number $s = H(id, r) \bmod f$, where id is the tag identification and H is a hash function. Each tag will decrement their slot number s. When a tag decreases to 0 and send a response which is a 16-bit random number. If only one tag whose slot number is 0 responses, the reader will receive a 16-bit random number and reply an ACK, and then the tag responses its identification. If two more tags responses at the same time, the reader will detect a collision and dose not reply any ACK. The tags keep silent until next round. The reader repeats this process and knows which tags should appear in each time slot, and therefore easily detects any tags which are missing.

In [29], the proposed scheme, Trusted Reader Protocol (TRP), let the tag reply random bits instead of its identification. The length of the random bits is less than tag ID', so that the execution time can be decreased. The authors in [21] proposed five protocols for solving the missing-tag problem. First four protocols use only one hash function, the fifth one uses two hash functions. In our study, we will focus on one hash function and improves it without hash function. That means the requirement of the RFID system is lower.

3. Enhanced STT protocol with distributed record tag-check mechanism

This section describes the RFID tag identification protocol proposed in the present study. As discussed in the following, the proposed protocol differs from existing methods in two key regards.

According to query construction rules, the QT protocol adds only a single bit to the query string following a collision event. However, such an approach not only increases the risk of further collisions (due to the larger number of query strings), but also results in a slower (i.e., 1 bit-level) downward traversal rate of the query tree. The STT protocol uses two deterministic parameters (r and w) to move down (or up) the query tree more rapidly.

In developing the tag identification protocol, it is argued that the time required to obtain all of the single-tag responses can be reduced by progressively increasing the number of bits appended to the query string following consecutive collision events. Importantly, such an approach increases the traversal rate and potentially avoids some collision nodes. However, by appending more than 1 bit to the query, some of the nodes in the query tree may be missed during the downward traversal process. Thus, in the Enhanced STT protocol, the interrogation process moves up 1 bit-level in the query tree each time an idle query response occurs. Furthermore, a blocking technique is applied to improve the performance of the identification protocol in checking the staying tags within the reader's communication range. Whenever an idle query (or collision) occurs, the length of the following query is decreased (or increased). In other words, in the proposed protocol, the next query length remains unchanged only when a single-tag response is received. Importantly, this variable query length is exploited to develop a Distributed Record Tag-Check (DRTC) mechanism, which enables each tag to record its transmission "Slot number" directly without the assistance of the reader. In other words, the reader is not required to transmit frame-slot information to the tags, and thus the overall overhead of the tag-collection process is reduced.

3.1. System assumptions

In developing the proposed tag-collection protocol, the following assumptions are imposed:

- Assumption 1: The signals between the reader and the tags are transmitted over an error-free wireless channel. As a result, data errors occur only when multiple tags respond simultaneously, resulting in a collision event at the reader.
- Assumption 2: Once the current tag identification round begins, the tags in the reader's communication range remain unchanged until the round terminates since if a tag arrives in the reader's communication range just after the reader has broadcast its query, it may miss the opportunity to be identified.
- Assumption 3: A tag entering or leaving the reader's communication range are independent of one another.

Note that all three assumptions are consistent with those used in previous studies on RFID anti-collision protocols [2,6,10,12,13,18,33–35]. For the fairness of comparisons between our algorithm and previous studies, this paper still holds these assumptions. Our study focus on the algorithm of protocols. Although the issues, e.g. transmission on the error-prone channel, are important, it is out of the scope in our research and should be studied separately.

3.2. Enhanced STT (E-STT)

3.2.1. Dynamic query down

In the tag-collection phase, the conventional STT scheme traverses only 1-bit level down whenever it encounters a collision node. Thus, if many collisions occur consecutively, STT traverses query tree very slowly. Accordingly, as described above, the Enhanced STT scheme utilizes a Dynamic Query Down (DQD) mechanism, in which a gradually increasing number of bits is appended to the query string each time consecutive collisions occur. Steps 1 to 3 in Fig. 5 (Example 1) present an illustrative example of the proposed approach. Table 1 and 2 compare the respective operations of the DQD and STT protocols. It is seen that the DQD scheme

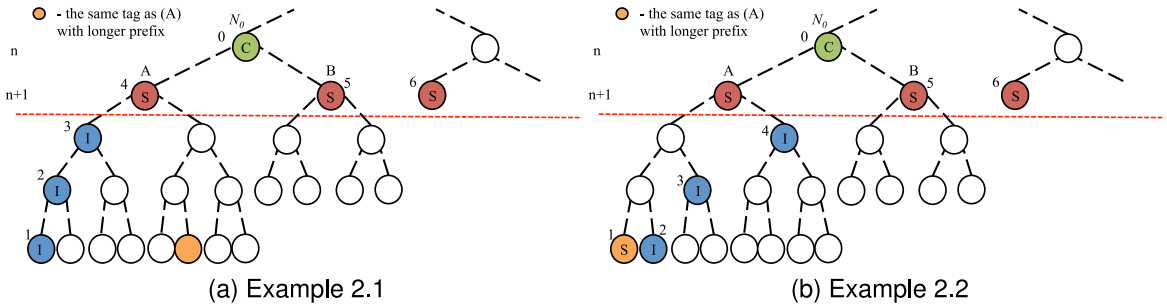

Fig. 3. Undesirable DQD outcome.

Table 1
Example 1 - DQD.

Step	Query string	Status	Appending
0	N_0	(C)	null
1	$N_0$0	(C)	+1
2	$N_0$000	(C)	+2
3	$N_0$000000	(S)	+3

Table 2
Example 1 - STT.

Step	Query string	Status	Appending
0	N_0	(C)	null
1	$N_0$0	(C)	+1
2	$N_0$00	(C)	+1
3	$N_0$000	(C)	+1
4	$N_0$0000	(C)	+1
5	$N_0$00000	(C)	+1
6	$N_0$000000	(S)	+1

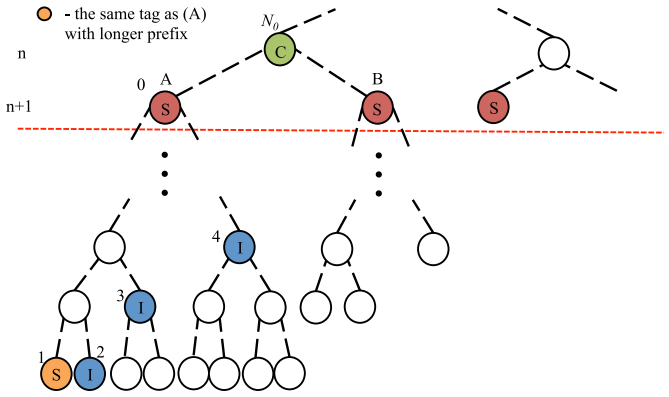

Fig. 4. Worst-case DQD outcome.

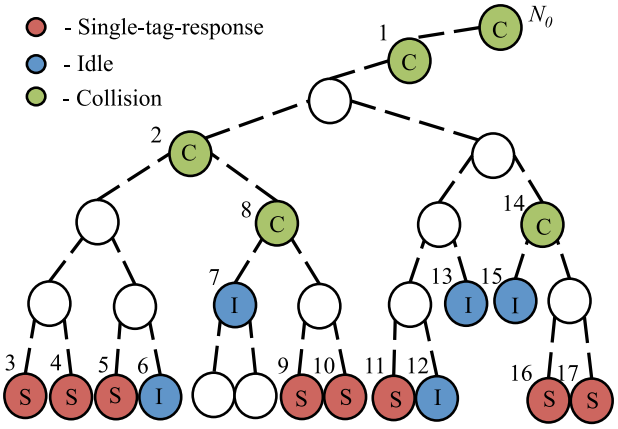

Fig. 5. Illustrative example of query construction rules.

saves a total of three queries in identifying the first single-tag response node compared to the conventional STT protocol. In the present study, the term “appending pattern” is used to define the strategy adopted by the reader in appending bits to the query following each consecutive collision.

In the original STT protocol, if an idle query occurs at the left-child node, the next query traverses to the child node to the right of this node. By contrast, in the Enhanced STT protocol proposed in this study, an idle response from a left-child node is followed by a query to the parent node since this node may not yet have been visited in the traversal process. However, if the parent node is a collision node, a closed-loop is formed and the traversal procedure cannot continue. Thus, in the present study, a shortcutting mechanism is applied to avoid this closed-loop scenario and to improve the overall efficiency of the query tree traversal process. Note that in contrast to the shortcutting mechanism proposed in [33], in the present study, when a collision occurs at the $(K-1)$th bit-level, the traversal process still visits the two child nodes and results in two single-tag-response nodes. Note that this strategy is deliberately adopted since the resulting variation in the query length is taken as the basis for the DRTC mechanism proposed in Section 3.3.

Importantly, while appending additional bits to the query string enables the query tree to be traversed more rapidly, it may cause two undesirable outcomes, as shown in Figs. 3a and 3b, respectively. Fig. 3a (Example 2.1) shows the case where the next query exceeds the next shortest prefix node. If node N_0 with current query q_c results in a collision and consecutive collisions then occur, the reader constructs the next query by appending 4 bits to q_c, i.e., $q_n = q_c0000$. However query q_c0000 in level $n+4$ may result in an idle (I) response. Thus, the Enhanced STT scheme traverses 1 bit-level to query q_c000, and so on. Fig. 3b (Example 2.2) shows the case where the next query exceeds the next shortest prefix node, but the query is happened to the prefix of the next single tag, and thus a single-tag response (S) occurs. Since a tag is successfully identified, the query process traverses to the right horizontally, sets $q_n = q_c0001$ and results in an idle response (I).

In both examples described above, the Enhanced STT protocol results in three more idle queries than the original STT scheme. Furthermore, for the case considered in Example 2.2, the Enhanced STT scheme uses more bits than in Example 2.1. In practice, Example 2.2 represents the worst-case scenario for the performance of the Enhanced STT scheme relative to the STT protocol (see Fig. 4).

In the event of consecutive collisions, the next query is formed by adding x_i 0's to q_c. The next shortest prefix node is node A and is located 1 bit-level down in the query tree. The reader first identifies node 1 with a query of length $n+x_i$. It then encounters

Table 3
Appending pattern table.

Cons. collision i	Appending pattern x_i	Query length $l_i = l_{i-1} + x_{i-1}$	Skipped queries $S_i = \{s_j \mid l_{i-1}+1, l_{i-1}+2, ..., l_i - 1\}$ # of S_i is $N_i = x_{i-1} - 1$, if $x_i > 1$
0	$x_0 = 1$	$l_0 = 1$	–
1	$x_1 = 2$	$l_1 = l_0 + x_0 = 2$	$S_1 = \emptyset$, $N_1 = 0$
2	$x_2 = 3$	$l_2 = l_1 + x_1 = 4$	$S_2 = \{l_1 + 1 = l_2 - 1\} = \{3\}$, $N_2 = 1$
3	$x_3 = 4$	$l_3 = l_2 + x_2 = 7$	$S_3 = \{l_2 + 1, l_2 + 2 = l_3 - 1\} = \{5, 6\}$, $N_3 = 2$
4	$x_4 = 5$	$l_4 = l_3 + x_3 = 11$	$S_4 = \{l_3 + 1, ..., l_3 + x_3 - 1 = l_4 - 1\} = \{8, 9, 10\}$, $N_4 = 3$
5	$x_5 = 6$	$l_5 = l_4 + x_4 = 16$	$S_5 = \{l_4 + 1, ..., l_4 + x_4 - 1 = l_5 - 1\}$, $N_5 = x_4 - 1$
...	...	...	...

$x_i - 1$'s consecutive idle queries $2, 3 \ldots$. After $x_i - 1$'s idle queries, the reader identifies node B. In other words, the proposed protocol saves $x_i - 1$ queries in identifying node 1, but wastes $x_i - 1$ queries in identifying node B.

As described above, x_i is progressively increased following consecutive collisions. In other words, a large value of x_i appended to the query indicates that multiple consecutive collisions have occurred. In practice, the total number of saved queries in previous collisions will be greater than the total number of idle queries wasted. Thus, it is expected that the total number of collision queries in the Enhanced STT scheme is lower than that in the traditional STT scheme.

In general, let $(l_i + 1)$ be the next single-tag-response node and let the appending pattern table have the form shown in Table 3. The total number of wasted idle queries is equal to $x_i - 1$. Meanwhile, the total number of saved queries is equal to

$$Q_{saved} = \sum S = \sum_{j=1}^{i} (x_{j-1} - 1) \tag{1}$$

Q_{saved} is greater than zero when $i \geq 2$ since it is assumed that x_0 is 1 and $S_1 = \emptyset$. The probability of $i \geq 2$ is given by

$$P(i = 2) + P(i = 3) + ... = \sum_{j=1}^{i} P(j), l_i \leq K \tag{2}$$

Assuming that the probability of the next bit being 0 is equal to 0.5, the probability of $i \geq 2$ is around 0.125 for the node located at the $K - 3$ bit-level above since $i \geq 2$ means that there are at least three consecutive 0's after q_c. The total number of queries required to complete the tag identification process in STT is directly proportional to the tag population. Thus, for a large tag population, the Enhanced STT scheme proposed in the present study based on the DQD mechanism yields a significant reduction in the total number of queries required.

3.2.2. Query construction rules

The construction of the next query q_n depends on the response received to query q_c. As described earlier, the reader may receive three different responses, namely collision, idle or single-tag. The query construction rules for each type of response are described in the following.

- Collision:
  When the reader detects a collision, it infers that the current query string $q_c = b_1 b_2 \ldots b_{lc}$ is at a higher bit-level and should traverse down to a lower bit-level. The reader first checks if the shortcutting condition exists. If the query string $q_1 = q_c 0$ has been visited and resulted in an idle response, the next query string is set as $q_n = b_1 b_2 \ldots b_{lc-1} 10$. As a result, the query process skips an idle and double query string q_1. However, if the shortcutting condition does not exist, the next query string is constructed using the DQD mechanism, as described above. The choice of the number of bits to append to the current query string $q_c = b_1 b_2 \ldots b_{lc}$ is determined from an appending pattern table. In general, the reader appends x_i bits of 0's to q_c. Thus, $q_n = q_c 0 \ldots 0$. Note that subscript i denotes the number of consecutive collisions, and its value increases by one each time a collision occurs. Importantly, if the current query length l_c plus x_i exceeds the total tag ID length K (i.e., 96 bits), q_n is simply truncated to 96 bits. It is seen from the preceding discussions that the length of query l_n following a collision event is always longer than the length of the current query l_c.
- Idle:
  When the reader detects an idle response, it indicates that the current query $q_c = b_1 b_2 \ldots b_{lc}$ is at a lower bit-level and should traverse up the query tree to a higher bit-level. The consecutive collision subscript i is thus set to zero. As for the collision case described above, the reader first checks for the presence of the shortcutting condition. If the idle response occurs at a left-child node, i.e., b_{lc} is 0, and the query string $q_2 = b_1 b_2 \ldots b_{lc-1}$ has been visited and results in collision, then the next query is set as $q_n = b_1 b_2 \ldots b_{lc-1} 10$. Else; if b_{lc} is 0 but $q_2 = b_1 b_2 \ldots b_{lc-1}$ has not been visited, the next query string q_n is set as q_2, $q_n = b_1 b_2 \ldots b_{lc-1}$ in order to query the parent node. Note that if this shortcutting process is not performed, a closed loop will be formed. For example, if $q = 010$ results in a collision and $q' = 0100$ results in an idle response, the next query string q'' will be set to 010 if shortcutting is not performed. If the idle response occurs at a right-child node, i.e., b_{lc} is 1, the next query should be sent to/directed to the upper-right node relative to q_c, i.e., $q_n = (b_1 b_2 \ldots b_{lc-1}) + 1$. It is noted from the preceding discussions that the length of query l_n following an idle response is always shorter than that of the current query l_c if shortcutting is not performed. However, l_n is greater than l_c by one if shortcutting is performed. In other words, the length of query l_n following an idle response is always different from that of l_c irrespective of whether or not shortcutting is performed.
- Single-tag response:
  When the reader detects a single-tag response, the query should traverse horizontally to the right in order to identify the node next to the current node. The next query string is therefore set as $q_n = q_c + 1$. The length of query l_n following a single-tag response is equal to that of the current query l_c. In other words, l_n is unchanged.

The following discussions demonstrate the query construction rules described above using Example 1 in Fig. 5 for illustration purposes. The query construction steps are shown in Fig. 5 and Table 4. In Steps 1 to 3, the number of appended bits is progressively increased following multiple consecutive collisions. Moreover, Steps 8 and 9 relate to the shortcutting process following collision. Specifically, knowing from Step 7 that the left-child node was idle, the reader appends '10' to the query in Step 8. Steps 15 and 16 relate to the shortcutting in idle condition since in Step 14, the parent node was visited and found to collide.

As described above, the query length is changed following an idle response or a collision, but remains unchanged following a

Table 4
Query construction steps for Example 1.

Step	Query string	Status
0	N_0	(C)
1	$N_0$0	(C)
2	$N_0$000	(C)
3	$N_0$000000	(S)
4	$N_0$000001	(S)
5	$N_0$000010	(S)
6	$N_0$000011	(I)
7	$N_0$00010	(I)
8	$N_0$0001	(C)
9	$N_0$000110	(S)
10	$N_0$000111	(S)
11	$N_0$001000	(S)
12	$N_0$001001	(I)
13	$N_0$00101	(I)
14	$N_0$0011	(C)
15	$N_0$00110	(I)
16	$N_0$001110	(S)
17	$N_0$001111	(S)

single-tag response. In the present study, the variation (or not) between l_c and l_n, is exploited to develop a mechanism designated as Distributed Record Tag-Check (DRTC) for improving the efficiency of the tag identification process in checking for staying tags.

3.3. Distributed record tag-check on E-STT with blocking protocol

In our proposed scheme, the reader identifies all tags first time by the Enhanced-STT. After that the reader will do the Distributed Record Tag-Check on E-STT with Blocking Protocol (DRTC/BP) every times. And the time of tag collection process will be saved much more, especially in an inventory system. The blocking feature has an advantage in the case, but it does not in a dynamic application case. DRTC/BP comprises two phases, namely staying tag phase: Distributed Record Tag-Check (DRTC) and arriving tag phase: Enhanced STT with blocking protocol (E-STT/BP). Note that the Enhanced STT has been described in Section 3.2. E-STT/BP extends STT so that having blocking feature which describes in Section 3.3.2.

3.3.1. Distributed record tag-check mechanism

There are two parts in DRTC. The first part identifies the missing tags by the slotted frame. The second part shortens the length of the slotted frame to decrease the execution time.

- Distributed record by tags:
  The "distributed record" concept used in the present study is adopted from the Three-Phase Protocol with Collision Sensitive Tag Removal (TPP/CSTR) method proposed in [21]. In order to realize the distributed record concept, this study further uses the blocking protocol presented in [35] to distinguish the staying tags from the arriving tags. As described earlier, the staying tags are those tags which were identified in a previous tag-collection process and whose IDs are therefore stored in the reader's memory or in a linked database. The situation is similar to [21] when we considering the multiple rounds of tag-collection process.
  In contrast to the protocols presented in [21], in which the identified tags are mapped to a given frame slot using a hash function, the DRTC scheme proposed in the present study maps the identified tags using the variable query length information. In the TPP/CSTR protocol, more than 2 tags in one frame-slot is reduced to 2-collision slot. However, the optimal frame size is equal to approximately 1.14 times the total number of tags (N_{tag}) since the hash function results in the creation of some idle slots and single-tag slots during the mapping process. However, if every frame-slot is a 2-collision slot, as shown in Fig. 6, the frame size can be reduced to around half the total number of tags. In other words, the performance of the reader in checking the staying tags can be significantly improved. Thus, in developing the DRTC mechanism proposed in this study, the overriding goal is to generate 2-collision slots in every frame slot during the mapping process.

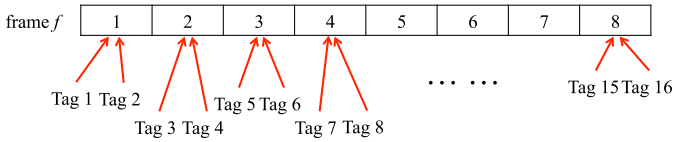

Fig. 6. Basic concept of DRTC mechanism.

As described in Section 3.2.2, the length l_n of the query following a single-tag response remains unchanged from that of the current query l_c. By contrast, for the case of a collision, l_n is always greater than l_c, while for the case of an idle response, l_n is generally less than l_c unless the shortcutting condition occurs. In other words, the query length remains unchanged only for the case of a single-tag response. By exploiting this variable query length information, each tag can determine the query state directly without the assistance of the reader. Thus, as described below, in DRTC, each tag stores to memory a parameter l_l and a counter to record its slot number.
In the Enhanced STT protocol, all of the tags maintain a counter, i.e., *Slot_num*, to record their slot parameter. The counter value is set to 1 initially. In addition, the tags store to memory the last query length as parameter l_l. On receiving a query, the tags then need only to compare the current query length l_c with the stored value l_l. If l_c differs from l_l, the tags infer that no tag was identified by the reader in the preceding query process. Hence, the value of *Slot_num* remains unchanged. However, if l_c is equal to l_l, the tags infer that the reader identified a tag, and therefore increment the value of *Slot_num* by 1.
When the prefix of a tag matches the current query string q_c, the tag should remain and the *Slot_num* counter is locked. Note that the *Slot_num* value also means that the next identified tag's sequential number. That is, if a tag prefix matches q_c, the tag is the *Slot_num*th tag identified by the reader.
In the frame phase of DRTC process, the reader broadcasts a *request* command *r*(*f*) and each tag is mapped to the $\lceil \frac{Slot_num}{2} \rceil$th frame-slot. Since every staying tag knows the order in which it is identified by the reader, i.e., *Slot_num*, mapping to the $\lceil \frac{Slot_num}{2} \rceil$th frame-slot causes all of the frame-slots to be two-collision slots. On receiving the *request* command *r*(*f*) from the reader, each tag counts down to its frame-slot and then transmits a long response. Note that the long response comprises multi-bits and indicates whether the corresponding slot is idle, single-tag-response, or collision. For example, in the protocol proposed in [21], the long response is based on the Philips I-Code system [27] and uses 10 bits to distinguish single-tag-response slots from collision slots. In the event that some of the slots transpire to be single-tag-response slots, the reader records the slots and initiates the polling phase.
In the polling phase, the reader sends a query to those tags in single-tag response slots in order to verify their presence. The total execution time of the DRTC mechanism is given by

$$T = (t_{prefix} + t_s) \times (M) + f \times t_l \quad (3)$$

where t_{prefix} is the time required to transmit a query prefix, t_s is the time required to transmit a short response, M is the number of tags in a single-tag-response slot, f is the frame size (equal to approximately half the total number of tags, i.e., N_{tag})

and t_l is the time required to transit a long response. As in TPP/CSTR, the number of missing tags is unknown, and thus the reader cannot evaluate the total execution time of the polling phase. However, assuming that missing-tag events are relatively rare, i.e., tens of missing tags to thousands even tens of thousands of total tags, the execution time of the polling phase can be effectively ignored. Hence, the total execution time of DRTC can be given as

$$T' = f \times t_l. \tag{4}$$

As discussed earlier, the optimal frame size in TPP/CSTR is equal to around 1.14 times the total number of tags, N_{tag}. However, in DRTC, the optimal frame size is equal to only 0.5 times N_{tag}. In other words, DRTC reduces the frame size by around 56% compared to TPP/CSTR, and therefore significantly improves the performance (i.e., execution time) of the reader in checking the staying tags. Moreover, the reader is not required to send the *Slot_num* parameter to the identified tags since all of the tags in the system record this parameter in a distributed way. Consequently, the DRTC mechanism does not increase the overall overhead of the proposed tag identification protocol.

As in the Enhanced Couple-Resolution Blocking (ECRB) protocol proposed in [35], the reader in the current protocol also exploits the 2-collision concept. However, in ECRB, the reader needs to broadcast the prefix queries which result in 2-collision events and tags with matching prefixes then return their entire ID to the reader. Consequently, the execution time is given by

$$T_{ECRB} = (t_{prefix} + t_{tagID}) \times \lceil \frac{N_{tag}}{2} \rceil \tag{5}$$

The term $(t_{prefix} + t_{tagID})$ is greater than t_l intuitively. Thus, of the two protocols, the DRTC protocol results in a lower execution time when checking the staying tags.

It is noted that a problem occurs if some of the tags receive a query incorrectly during the tag-collection process. In such a situation, an error will occur when updating the *Slot_num* counter value. In theory, Assumption 1 given in Section 3.1 should prevent such a scenario from occurring. However, in real-world systems, this problem must be taken into account.

- Re-allocation mechanism:

If having executed the DRTC mechanism some of the time-slots transpire to be idle slots, the reader initiates a *Slot_num* re-allocation process to remove the idle slots and rearrange the *Slot_num* values of the remaining tags accordingly. Specifically, the reader broadcasts a *Slot_num* reallocation command containing a parameter *s*, where *s* is the former *Slot_num* value of an idle slot. On receiving this command, each tag with a *Slot_num* value equal to or greater than s counts up to a parameter *de_n*. Once the reader has broadcast reallocation commands containing the *Slot_num* values of all the idle slots, it broadcasts a reallocation termination command. On receiving this command, each tag reduces the value of its *Slot_num* counter by $(2 \times de\_n)$. In addition, the reader decreases the total tag number, N_{tag}, by decreasing an amount equal to (2× number of re-allocation command sends).

Consider the illustrative case shown in Fig. 7. Assume that in the polling phase, the reader finds that tags with *Slot_num* values equal to 3, 5, 6, 7, 8, 11 and 12 are missing. The reader thus broadcasts *Slot_num* reallocation commands with $s = 9$, $s = 9$, and $s = 13$ and then issues a reallocation termination command. On receiving the reallocation commands, slots with *Slot_num* values equal to 9 and 10 reduce their *Slot_num* counter by $2 \times 2 = 4$. Meanwhile, slots with *Slot_num* values equal to 13, 14, 15 and 16 reduce their *Slot_num* counters by $2 \times 3 = 6$.

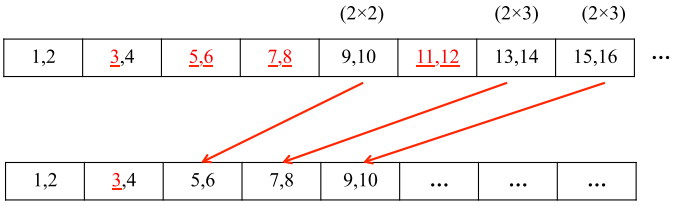

Fig. 7. Schematic representation of *Slot_num* re-allocation mechanism.

The protocol proposed in this study deliberately uses a simple mechanism to remove idle slots since it is assumed that the tags in the RFID network have a low mobility and therefore generally remain within the reader's communication range (i.e., most of the tags are staying tags). In dynamic environments, the simplistic reallocation mechanism may result in a significant increase in the overall overhead of the tag collection process. Thus, a threshold value should be applied such that a new tag-collection round is initiated if the number of missing tags exceeds the specified threshold value.

3.3.2. E-STT with blocking protocol

In the arriving tag phase, the reader has been identified staying tags and want to find out arriving tags. Thus, the reader uses E-STT with blocking protocol(E-STT/BP) which is as same as E-STT essentially. The E-STT/BP differs only from the initial value *Slot_num*. The initial value *Slot_num* of E-STT is set to zero. However, the one of E-STT/BP is set to N_{tag} for mapping the arriving tags to their frame slots. The arriving tags become the staying tags next DRTC/BP round.

In this tag-collection process, the reader broadcasts the command with parameter N_{tag}. In addition, the arriving tags set their *Slot_num* values to N_{tag} in preparation for the next round of the DRTC mechanism.

3.4. Reader and tag pseudocode

This section summarizes the basic steps in the reader and tag operations in the proposed protocol. The basic reader operation is as follows:

1. Set identified tag number to N_{tag}.
   Send Staying Frame Phase command (for staying tags).
   Then send request command $r(f)$, $f = \lceil \frac{N_{tag}}{2} \rceil$.
2. 1st-*phase* Distributed Record Tag-Check.
3. Send *Slot_num* re-allocation command.
4. Send 2nd-*phase* command, with parameters (send N_{tag} to tag, set $Slot\_num = N_{tag}$).
5. 2nd-*phase* Enhanced STT with blocking protocol. (For unidentified tags).

Meanwhile, the basic tag operation is as follows:

1. Initialize *Slot_num* counter to 1.
2. On receiving Staying Frame Phase command and request $r(f)$,
   (a) Send Long_Response in $\lceil \frac{Slot_num}{2} \rceil$th slot.
   (b) On receiving *Slot_num* re-allocation command.
3. On receiving Phase 2 command with N_{tag},
   (a) If $Slot\_num == 1$, set $Slot\_num = N_{tag}$;
   (b) On receiving a query q_c, compare its length l_c to previous query length l_l.
   If l_c equals l_l, set $Slot\_num = Slot\_num + 1$.
   (c) Set $l_l = l_c$;

Table 5
Notations used in proposed protocol.

Notation	Description
$q_c = b_1 b_2 \dots b_{lc}$	Current query prefix
q_n	Next query prefix
l_c	The length of current query
l_l	The length of last query
i	The number of consecutive collisions
x_i	The number of appending bits
K	The total length of tag ID
N_{tag}	The total number of tags record in reader
Slot_num	The counter value used in 1st-*phase*
CurF	The current frame number
NextF	The next frame number
rRID	The reader's ID record in reader
tRID	The reader's ID record in tag
f	The frame length in request command

(d) If $prefix(ID) == q_c$, transmit ID;

The notations used in the proposed algorithm are summarized in Table 5. Meanwhile, the pseudocode for the reader and tag operations is presented in Table 6 and 7.

3.5. Design complexity comparison

Table 8 compares the design complexity of the proposed Enhanced STT with DRTC protocol with that of other protocols presented in the literature. In realizing the blocking protocol, the tags need to store parameters such as *tRID* and *TF*. Thus, of all the protocols shown in Table 8, only the original STT protocol preserves the memory-less feature of the original QT protocol. In addition, the hash function is not necessary in our algorithm. But it is needed when finding missing tag in other algorithms. The complexity of our algorithm is O(1) so that the algorithm can save the computing time and power. Moreover, our algorithm would reduce costs of equipment.

4. Performance evaluation

4.1. Simulation setup

The performance of the Enhanced STT protocol was evaluated by means of MATLAB simulations and compared with that of the conventional STT protocol. Note that the DRTC mechanism has no direct counterpart in the literature. Moreover, its performance has already been evaluated in Section 3.2. Thus, its performance is not discussed here. As for the QT evaluation process performed in [17], the present study evaluates the performance of the Enhanced STT protocol in terms of two metrics, namely the System Efficiency (*SE*) and the Time System Efficiency (*Time_SE*).

The *SE* metric evaluates the efficiency of the protocol in using the frame slots, and is defined as follows:

$$SE = \frac{Q_S}{Q_{Total}} = \frac{Q_S}{Q_I + Q_S + Q_C} \tag{6}$$

where Q_S is the number of single-tag response queries and Q_{Total} is the total number of queries used in identifying the complete tag set. In other words, Q_{Total} is the sum of the single-tag response queries (Q_S), idle queries (Q_I) and collision queries (Q_C). The time durations of these queries are different. Specifically, the time duration of the idle queries is much shorter than that of the single-tag-response queries or collision queries since in an idle query, the tags do not need to transmit their IDs to the reader. Since the queries have different durations, a time system efficiency is introduced. In computing the *Time_SE*, the idle queries are normalized to the length of the other two types of query through a multiplicative factor β such that all three types of query have the same duration. The *Time_SE* metric is computed as follows:

$$Time\_SE = \frac{Q_S}{\beta Q_I + Q_S + Q_C} = \frac{Q_S}{Q_{Total} + (\beta - 1)Q_I} \tag{7}$$

where β is taken as 0.13 in the present evaluations. As described in [17]. β represents the ratio of the number of idle queries to the number of collision queries, and has a value of 0.13 according to the EPC global standard specification.

As discussed in [17], maximizing the *SE* requires the reader to collect all of the tags in the system using the minimum total number of queries. Regarding the *Time_SE*, a trade-off exists between the number of collision queries and the total number of queries. Specifically, the *Time_SE* improves when a small increase in the number of idle queries results in a significant reduction in the number of collision queries. In addition to the *SE* and *Time_SE*

Table 6
Pseudocode of reader operation.

```
Reader operation:
1 CurF = NextF;
2 NextF = NextF + 1;
3 Phase = 1;   N_tag = 0;   q_c = 0
4 Transmit the 1st-phase command with rRID, CurF, and
NextF
5 if Phase == 1 then
6   f = ⌈N_tag / 2⌉;
7   if f > 0 then
8     Transmit request command r(f);
9     Receiving tags response until f slots;
10    while there are some empty slot do
11      Transmit Slot_num reallocation command with s;
12    end while
13    Transmit reallocation ending command;
14    Transmit the 2nd-phase command with N_tag;
15    Phase = 2;
16  end if
17 end if
18 while Phase == 2 do
19   Transmit the query q_c;
20   if The reader detects Collision then
21     if q = q_c0 was visited and idle then
22       q_n ← (b_1 b_2 ... b_{lc-1} 10) ;
23     end if
24   else
25     i = i + 1 ;
26     q_n ← q_c ;
27     if l_c + x_i > K then
28       for j = 1 to K − l_c do
29         q_n ← q_n0
30       end for
31     else
32       for j = 1 to x_i do
33         q_n ← q_n0;
34       end for
35     end if
36   if The reader detects Idle then
37     i = 0;
38     if q = (b_1 b_2 ... b_{lc-1}) was visited and collided then
39       q_n ← (b_1 b_2 ... b_{lc-1} 10) ;
40     else if sum(q_c) == length(q_c)
41       Phase = 1;
42     else if b_lc == 0 then
43       q_n ← b_1 b_2 ... b_{lc-1}
44     else
45       q_n ← (b_1 b_2 ... b_{lc-1}) + 1
46     end if
47   else if detects Single-tag-response then
48     i = 0;
49     N_tag = N_tag + 1;
50     q_n ← (b_1 b_2 ... b_lc) + 1;
51   end if
52 q_c ← q_n
53 end while
```

Table 7
Pseudocode of tag operation.

```
Tag operation:
1 Slot_num = 1;
2 Receive message m from the reader;
3 while m! = the frame ending command do
4   if m is the 1st-phase command
      and tRID == rRID and TF == CurF then
5     Respond = 1;
6   else
7     Respond = 0;
8   end if
9   if m is the request command r(f) and Respond == 1 then
10    Transmit Long_Response in ⌈Slot_num/2⌉th slot;
11  if m is the Slot_num reallocation command with s then
12    if Slot_num ≥ s then
13      de_n = de_n + 1;
14    end if
15  if m is the reallocation ending command then
16    Slot_num = Slot_num − 2 × de_n;
17  end if
18  if m is the 2nd-phase command with N_tag then
19    if Slot_num == 1 then
20      Slot_num = N_tag;
21    end if
22    Respond = not Respond;
23    tRID = rRID;
24    TF = NextF;
25  end if
26  if m is a query and Respond == 1 then
27    l_c = length(m);
28    if l_l == l_c then
29      Slot_num = Slot_num + 1;
30    end if
31    if prefix(ID) == m then
32      Transmit ID;
33    end if
34  end if
35  Receive message m from the reader;
36 end while
```

metrics, this study also evaluates the performance of the proposed protocol by investigating the reduction achieved in the numbers of total queries (Q_{Total}); idle queries (Q_I) and collision queries (Q_C), respectively, compared to the original STT scheme.

To determine the optimal appending pattern for the Enhanced STT protocol, a preliminary set of simulations was performed using the six appending patterns shown in Table 9 given the assumption of a RFID system containing 5000 tags. The corresponding *SE* and *Time_SE* results are shown in Table 10. It is observed that appending pattern X_1 results in the highest *Time_SE* of the various patterns. Thus, X_1 was taken as the default pattern for the Enhanced STT scheme in all of the remaining simulations. However, appending pattern X_4 was taken as the appending pattern for the STT-s (STT with shortcutting) scheme. Note that STT-s is a particular case of the Enhanced STT protocol in which the appending pattern contains only 1's. In other words, the query process traverses down the query tree 1-bit level each time it encounters a collision.

Table 9
Trial appending patterns and frequency of x_i.

	X_1	Freq.	X_2	Freq.	X_3	Freq.	X_4	Freq.
x_1	1	1671	1	1685	1	1689	1	1690
x_2	2	853	1	850	1	846	1	845
x_3	3	74	2	285	1	285	1	285
x_4	4	3	3	21	2	72	1	72
x_5	5	1	4	1	3	5	1	21
x_6	6	0	5	0	4	1	1	3
x_7	7	0	6	0	5	0	1	1
x_8	8	0	7	0	6	0	1	0

Table 10
Simulation results for *SE* and *TIME_SE* given different appending patterns.

$K = 96$bits, $N_{tag} = 5000$				
	X_1	X_2	X_3	X_4(STT-s)
Q_{Total}	14091	13826	13685	13644
Q_C	3195	3374	3422	3429
Q_I	5896	5452	5263	5205
SE	0.355	0.361	0.365	0.366
Time_SE	0.558	0.550	0.549	0.548

The simulations commenced by investigating the effect of the total number of tags on the performance of the Enhanced STT, STT and STT-s protocols. In performing the simulations, N_{tag} was assigned values of 1000, 3000, 5000, 10000 and 20000, respectively. The tag ID length was specified as 96 bits, because the tags'electronic product code usually has 96 bits. This setting will make it closer real world. And the tags were randomly generated. In other words, the tag IDs were uniformly distributed. The simulations then considered the case of different tag ID distributions given a constant total number of tags (i.e., $N_{tag} = 5000$). The simulations considered four tag ID distributions, namely: (1) a normal distribution with mean 2^{K-1}, i.e., the tags are congregated at central of 2^K; (2) a normal distribution with mean 2^{K-2}, i.e., the tags are congregated at the quarter of 2^K; (3) a normal distribution with mean $2^{K-1} + 2^{K-2}$, i.e., the tags are congregated at the third quarter of 2^K; and (4) an equal division of the tag set into two groups

For each simulation environment, 30 simulations were performed using a different tag set on every occasion. The results were then verified by means of the statistical means and 95% confidence intervals of the corresponding evaluation metrics. The statistical mean was computed as

$$\bar{Y} = \sum \frac{y_i}{N_{sim}} \tag{8}$$

where y_i is the simulation result and N_{sim} is the number of simulations.

The statistical variance is given by

$$S^2 = \sum \frac{(y_i - \bar{Y})^2}{(N_{sim} - 1)} \tag{9}$$

Table 8
Design complexity comparison.

	DRTC on enhanced STT	Enhanced STT	STT	ECRB	PRB	TPP/CSTR
Counters in tag	1	N	N	N	3	1
Prefix matcher	Y	Y	Y	Y	N	N
Hash function	N	N	N	N	N	Y
Blocking protocol	Y	Y	N	Y	Y	N
Tag-collection process	Improve from STT	Improve from STT	Improve from QT	Query tree	Binary tree	Tag IDs store in database
Missing-tag detection	Frame-slots / 2-collision	Known prefix / 2-collision	Repeat STT again	Known prefix / 2-collision	BT / 2-collision	Frame-slots / 2-collision

Table 11
Simulation results for query given different tag populations.

Comparison	Q_{Total}	95% confidence interval	Q_C	95% confidence interval	Q_I	95% confidence interval
STT-s	−8.86%	(−8.98%, −8.74%)	4.01%	(3.70%, 4.32%)	−22.14%	(−22.31%, −21.97%)
E-STT	−5.75%	(−5.91%, −5.60%)	−3.40%	(−3.78%, −3.25%)	−11.33%	(−11.57%, −11.10%)

Table 12
Simulation results for *SE* and *Time_SE* given different tag populations.

	SE	95% confidence interval	*Time_SE*	95% confidence interval
STT	0.339	(0.338, 0.340)	0.549	(0.548, 0.550)
STT-s	0.372	(0.371, 0.373)	0.553	(0.551, 0.553)
E-STT	0.359	(0.358, 0.360)	0.562	(0.561, 0.563)

Table 13
Simulation results of query comparison given different tag distributions.

Comparison	Q_{Total}	95% confidence interval	Q_C	95% confidence interval	Q_I	95% confidence interval
STT-s	−8.86%	(−8.96%, −8.76%)	4.02%	(3.82%, 4.19%)	−22.12%	(−22.30%, −21.96%)
E-STT	−5.71%	(−5.85%, −5.65%)	−3.40%	(−3.66%, −8.13%)	−11.34%	(−11.55%, −11.12%)

Table 14
Simulation results of *SE* and *Time_SE* in varying tag distribution.

	SE	95% confidence interval	*Time_SE*	95% confidence interval
STT	0.339	(0.338, 0.340)	0.549	(0.547, 0.550)
STT-s	0.372	(0.370, 0.373)	0.553	(0.551, 0.553)
E-STT	0.359	(0.358, 0.360)	0.562	(0.561, 0.563)

Thus, the 95% confidence interval can be derived as

$$\left(\bar{Y} - Z_{\frac{0.05}{2}} \times \frac{S}{\sqrt{N_{sim}}}, \bar{Y} + Z_{\frac{0.05}{2}} \times \frac{S}{\sqrt{N_{sim}}}\right) \quad (10)$$

where $Z_{\frac{0.05}{2}}$ is 1.96.

4.2. Simulation results

Table 11 shows the performance of the STT-s and Enhanced STT schemes in reducing the various types of query within the RFID system relative to the original STT scheme. The simulation results are obtained for uniformly distributed RFID systems with N_{tag} = 1000, 3000, 5000, 10000, and 20000 tags. Note that the plotted points correspond to the statistical mean of the corresponding metric in every case, and the margin of error is the 95% confidence interval. In general, the results show that none of the metrics are affected by the size of the tag set in any of the considered protocols. This result is to be expected since the tag populations considered in the present simulations are relatively sparse compared to the whole tag ID space, which is of the order of 2^{96}. It is observed that even for N_{tag} = 20000, the query tree does not become saturated as in [18]. Table 12 summarizes the simulation results obtained for the SE and Time SE performances of the three schemes. It is seen that the SE of the Enhanced STT scheme is slightly lower than that of the STT-s scheme, but higher than that of the original STT scheme. Furthermore, it is observed that the Enhanced STT scheme has the highest Time SE of the three schemes.

Table 13 and 14 show the four different tag distributions considered in the present study. The four distributions were found to yield very similar simulation results. Thus, the results are presented in the forms shown in Table 13 and 14. It is observed that the results are very similar to those presented in Table 11 and 12. In other words, STT and its variants are unaffected by the tag ID distribution. This result is reasonable since STT has the ability to adaptively adjust its Query Traversal Path (QTP) depending on the distribution of the tags [17]. For example, if the tag IDs are concentrated primarily in the left part of the query tree, STT (and its variants) traverse down the tree to identify the locally dense tags and then traverse back up the tree to the more sparse areas. As a result, the performance is similar. Table 13 and 14 summarize the performance of the various STT schemes for the different tag distributions.

Table 15
Comparison of *SE* and *Time_SE* for different tree-based protocols.

	SE	*Time_SE*
BS	0.34	0.40
QT	0.34	0.40
QTI	0.376	0.41
STT	0.339	0.549
STT-s	0.372	0.552
E-STT	0.359	0.562

Finally, Table 15 compares the *SE* and *Time_SE* performances of the three STT protocols with those of the Binary Splitting (BS), Query Tree (QT) and Query Tree Improved (QTI) protocols presented in [17]. It is seen that while the STT protocols obtain a lower *SE* than the QTI protocol, they achieve a higher *Time_SE* than the BS, QT or QTI protocols. In addition, it is seen that the proposed Enhanced STT protocol achieves the highest *Time_SE* of all the considered schemes. From inspection, its *Time_SE* is 1.8% higher than that of STT with shortcutting, 2.4% higher than STT, and 40.5% higher than QT.

5. Conclusion and future work

5.1. Conclusion

This paper has presented a tag identification protocol for RFID systems designated as Enhanced STT with Distributed Record Tag-Check (DRTC). In previous study, they usually either process the tag identification collection or detect the missing tags independently. We solve these two problems together and let our proposed scheme more efficient.