EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Adaptive collision resolution for efficient RFID tag identification

Yung-Chun Chen[1], Kuo-Hui Yeh[2*], NaiWei Lo[1], Yingjiu Li[3] and Enrico Winata[1]

Abstract

In large-scale RFID systems, all of the communications between readers and tags are via a shared wireless channel. When a reader intends to collect all IDs from numerous existing tags, a tag identification process is invoked by the reader to collect the tags' IDs. This phenomenon results in tag-to-reader signal collisions which may suppress the system performance greatly. To solve this problem, we design an efficient tag identification protocol in which a significant gain is obtained in terms of both identification delay and communication overhead. A *k*-ary tree-based abstract is adopted in our proposed tag identification protocol as underlying architecture for collision resolution. Instead of just recognizing whether tag collision happens at each interrogation time period, the reader can further obtain the reason of why the collision occurs in the current tag inquiry operation. With this valuable information, we can reduce tag signal collisions significantly and at the same time avoid all of the tag idle scenarios during a tag identification session. The rigorous performance analysis and evaluation show that our proposed tag identification protocol outperforms existing tree-based schemes.

Keywords: anti-collision, RFID, tag identification

1. Introduction

As rapid advances in semiconductor technology have enabled the production of low-cost tags (usually in a range of five to ten cents), the Radio Frequency IDentification (RFID) technique is promptly adopted to replace traditional bar-code-based identification mechanism in many daily life applications such as inventory tracking, library book managing, and airport baggage conveying. RFID technology utilizes Radio Frequency (RF) to store and retrieve data via an RF compatible integrated circuit. An RFID application system, in general, consists of a number of readers and tags (or tagged objects). The tags typically derive their energy for operation and data transmission from a reader's electric, magnetic, or electromagnetic field. The reader recognizes tagged objects through a wireless channel in which each tag transmits its unique ID and other information.

Tag reading throughput is critical while scanning tagged items in a large-scale RFID application. Two main performance criteria, i.e., tag reading delay (which should be within acceptable time period) and the energy consumption of RF reader (which should be minimized) [1,2], are used for measuring RFID system throughput. In a normal tag identification process, the existence of numerous tags within the interrogation area of a reader may lead to a great number of signal collisions. This is because the reader and the tags communicate over a shared wireless channel. If more than two tags respond to the reader simultaneously, the signals transmitted by these tags collide with each other. Due to the signal collisions, either the reader cannot recognize tags (or tagged objects) or a retransmission request for tags' IDs is required, and thereby both of communication overhead and identification delay increase during the tag identification process. It is thus important to design an efficient tag collision arbitration mechanism in RFID systems.

In recent years, reader-talk-first (RTF) RFID tag identification protocols have seriously been investigated as new improvements in silicon technology and digital signal processing technology have mitigated or overcome the major shortcomings of RTF protocols: complex circuitry and reader interfe-rence problem, and the capability of RTF protocols on detecting large populations

* Correspondence: d9409101@mail.ntust.edu.tw
[2]Department of Information Management, Chinese Culture University, Taipei 111, Taiwan
Full list of author information is available at the end of the article

of tags in a short time period has been observed. RTF tag identification protocols are broadly classified into aloha-based schemes [3-12] and tree-based schemes [2,13-29]. In aloha-based schemes [30,31], a reader predicts the number of current tags within its interrogation area and assigns estimated number of timeslots to all of the tags. The tags randomly pick up their own timeslots for ID transmission. Aloha-based protocols reduce the probability of tag collisions by separating tag responses into distinct timeslots. However, aloha-based mechanisms suffer from the so-called "tag starvation problem," in which certain tags may not be identified for a long time and the time period required for all tags' recognition may not be guaranteed.

On the other hand, tree-based schemes [32,33] utilize the tags-set splitting mechanism based on a prefix value issued by the interrogator, i.e., the reader continuously split a set of currently collided tags into two subsets until each set contains only one tag. This kind of procedure guarantees that all tags' IDs are identified within a certain time period. Nevertheless, the tree-based protocols exploit the information obtained from tags' responses to determine which tag subset should be constructed. It results in higher energy consumption and identification delay due to the vast splitting and invoking operations. In general, for aloha-based protocols synchronization command such as *Null* is used by reader to signal all tags the end of a timeslot and consequently synchronize all tag responses in line with the time duration of given timeslots. Therefore, a signal collision caused by tag response can only be occurred at a given timeslot when multiple tags send their replies at the same timeslot. For tree-based protocols, bit-by-bit synchronization on tag response is desired such that the reader can detect the colliding bit positions of re-ceived tag responses. This kind of collision detection mechanism can be implemented using synchronization command and specific signal encoding scheme such as Manchester code.

Recently, four more anti-collision studies were presented. Zhu et al. [11] proposed an optimal-framed aloha-based anti-collision protocol, in which the reading process is modeled as a Markov Chain and the optimal reading strategy is accordingly derived by first-passage-time analysis. Later, Li et al. [12] presented an aloha-based anti-collision scheme. The capture-aware backlog estimation method and optimum frame length equation are exploited to analyze the maximum achievable throughput of their scheme. Jia et al. [29] developed an efficient tree-based tag identification protocol, where a collision tree is used to capture the complete communications between the reader and the tags. The novelty of their me-thod is that the prefixes generation and tag group splitting are based on the collided bit directly. Porta et al. [28] proposed a new metric, i.e., time system efficiency, to evaluate anti-collision protocols. This metric provides a direct measure of the time taken to read a group of tags. In this study, we propose a tree-based tag identification protocol, called *k*-ary Tree-based Anti-collision Scheme (*k*-TAS) to pursue better identification efficiency. In *k*-TAS, the reader first recognizes whether collision happens and, if it happens, the reason of why the collision occurs at each tag inquiry time period. Within a tree-based structure, the reader knows which descendant nodes collide with the currently visited node; in the next interrogation time, the reader can only focus on those nodes. As a result, this design allows the reader to avoid visiting all idle nodes. The reduction of tag signal collision and the elimination of all idle scenarios reduce the identification delay and communication overhead compared to existing tree-based protocols. Our performance analysis and evaluation show that *k*-TAS is efficient in reducing tag collisions while preserving low communication overhead.

This article is organized as follows. Section 2 introduces three application scenarios which are relevant to this study. A new RFID tag identification protocol (i.e., *k*-TAS) is introduced in Section 3. The performance analysis on the identification delay and communication overhead of *k*-TAS is addressed in Section 4. Next, Section 5 presents the simulation results of our proposed *k*-TAS. Finally, we give a conclusion in Section 6.

2. Relevant applications

As RFID technology provides an efficient and accurate way to identify physical resources and at the same time preserves very attractive deployment characteristics to industries such as simple system installation and deployment process, wireless accessibility and low-cost manufacture, various innovative applications have promptly been developed. Each of these RFID applications might require distinct system criteria in terms of the knowledge of the number of current tags, the duration of tag identification and the tag reading speed [9]. We elaborate on three application scenarios which are most relevant to our study as follows.

2.1. Airport baggage checking

Number of tags: known

Duration of tag identification: known

Reading speed: fast

In this case, we describe a scenario where there is a known number of tags in the interrogation field of a reader with a fast reading speed and known duration for tag identification. This scenario might occur in the following two examples: (1) a luggage management center in which lots of tagged luggage are being transported into target airplane via the conveyor, and (2) a luggage

storage place of ready-to-take-off airplane where all passengers' luggage is required to be assured again. In such scenario, the reader must be able to recognize tags as soon as possible in order to accelerate boarding activities.

2.2. Inventory managing

Number of tags: known

Duration of tag identification: known

Reading speed: slow

This section illustrates a scenario that might occur in a warehouse where companies could immediately know the location of any item as well as the management of inventory, such as goods boxes leaving, entering, and monitoring. In such scenario, there are a known number of tags within the reader's interrogation area according to a goods list in the backend database. The reading speed could be slow as the reader might be either fixed on specific location, such as goods stand, or a mobile handheld device. Accordingly, the duration of tag identification might be known.

2.3. Transported merchandise tracking

Number of tags: known

Duration of tag identification: unknown

Reading speed: fast

This case envisions future merchandise transporting in which goods are attached with tags on themselves. This case might occur at the retail distribution center where all tagged goods are being transported on a forklift through a portal embedded with a reader. Companies or retails could promptly recognize whether all transported merchandise exists at current time period by scanning attached tags. Hence, the reader requires identifying all tags as soon (and correctly) as possible when the tagged merchandise passes through the monitoring portal.

3. New tag identification protocol

In this study, we focus on the tag identification in above RFID applications in which the reader must be able to identify all tags as fast as possible. We consider a target system where a single RFID reader intends to efficiently communicate with a pile of passive tags (denoted as q tags throughout this article) within in its interrogation range for object tracking and monitoring. In addition, k-TAS requires that all transmitted data between the reader and all involved tags are synchronized during each interrogation time period. Current RFID technologies [4,6,16,28,34-37] have demonstrated this possibility by exploiting a Manchester encoding technique during each query session. In the following, we formally define some terminologies.

• Session

A session is the period from the moment the reader initializes the tag identification procedure to the time of all tags are actually recognized by the reader. Let S_l denote the lth session.

• Cycle

An interrogation cycle is the duration when the reader transmits a triggering signal command to all tags and the tags respond with their corresponding output. According to the number of tag responses, a cycle is idle, readable, or collided when no tag responds, one tag responds, or multiple tags respond, respectively. Note that in the abstract tree structure of tag identification protocol, such as Query Tree protocol, Binary Search (BS) Scheme, or k-TAS, a cycle can be represented as a node. Note that a session usually consists of numerous interrogation cycles.

3.1. k-Ary tree-based anti-collision scheme (k-TAS)

In k-TAS, we exploit an abstract k-ary tree structure to resolve tag signal collisions more effectively. Based on triggering command issued by the reader, involved tags insert useful piece information, i.e., a bit-sequence derived from some part of their IDs, in the corresponding response, i.e., the tags will respond a new data sequence, which possesses extra (and useful) information for collision resolution, to the reader instead of just their IDs. This design allows the reader to greatly reduce tags' response collisions and accordingly save more operation time and power energy during tag identifi-cation procedure. Figures 1 and 2 show the pseudocode at the tag side and the reader side, respectively, in k-TAS. We illustrate the detailed processes of k-TAS in the following.

Step 1: At the beginning of lth session S_l, the normal identification procedure is initially invoked when the reader broadcasts a *Start* command along with a predefined parameter i to all existing tags (lines 7-8 of Figure 2), where $i = \log_2 k$. Once receiving this *Start* command, each tag t_j retrieves a bit block B_j (lines 6-7 of Figure 1), which is the first i bits of its identity ID_j, and calculates the decimal value M_j of B_j (line 15 of Figure 1). A distinct *Collision Resolution String* (CRS_j) is then generated by each tag t_j and sent to the reader as one part of the response. A bit sequence generator, which can be implemented with simple circuit logics and a bit counter, is utilized by a tag t_j to construct its CRS_j. The generation process of CRS_j is as follows. Every time the bit sequence generator produces and sends out a bit, the bit counter is then increased by 1. The generator only generates bit 1 when the value of its bit counter is equal to M_j. Otherwise, bit 0 is generated by the bit generator.

```
The operation at tag side in k-TAS
/* Respond to the reader's query */

1   Receive a message p which should be a Start
2       command, a Terminate command or a Query
3       command with prefix value n.
4
5   while p is valid && p != Terminate command do
6       if (p is a Start command) then
7           retrieve the first i bits from identity as B_j
8       else if (p is a Query command with prefix n &&
9       the first successive bits of identity D_j = n) then
10          retrieve i continuous bits behind the prefix D_j
11              in its identity as B_j
12      else
13          go to line 18
14      end if
15      Calculate the decimal value M_j of B_j
16      Generate CRS_j and RS_j  = CRS_j || PID_j
17      Transmit RS_j
18      Wait for a message p from the reader
19  end while
```

Figure 1 Pseudocode of *k*-TAS: tag's operation.

Each tag t_j performs these procedures iteratively until the bit-length of CRS_j is $2^i = k$.

Once completing CRS_j, each tag t_j resets the value of its bit counter to zero and appends its partial identity PID_j behind the CRS_j to construct a *Response String* (RS_j), i.e., $RS_j = CRS_j||PID_j$. Note that PID_j is constructed by removing the first i bits from t_j's identity ID_j and || denotes a concatenate operation. Finally, each tag t_j sends the bit string RS_j to the reader as a response. The above processes can be referred to lines 16-17 of Figure 1.

Step 2: As a Manchester encoding technique is adopted for all data transmission, the communication channel among the reader and the tags at each interrogation cycle is synchronized. As a result, the reader can easily detect the positions of collided bits among received RS_j bit strings from all responding tags. Then, the reader recognizes all M_j values based on the collided bit positions among CRS_j bit strings contained in received RS_j bit strings (lines 14-16 of Figure 2). Note that if no bit collision occurs among received RS_j bit strings at the first time of tag enquiry, it means that the reader identifies a tag which is the only one tag within the reader's interrogation area. Next, the reader recovers all recognized M_j values to the original bit block B_j values, and stores these values (lines 20-23 of Figure 2). For each B_j, a command *Query* along with a prefix value n is broadcasted by the reader to all tags as tag ID interrogation in which $n = B_j$ (lines 10-13 of Figure 2). Based on the *Query* command and prefix value n sent from the reader, the tags act as follows.

Step 3: Once receiving *Query* command, only tags which possess the same prefix value D_j, i.e., $D_j = n$, in its identity ID_j will respond. Each responding tag t_j retrieves i conti-nuous bits, which is behind the prefix value D_j in its ID_j, as its bit block B_j (lines 10-11 of Figure 1). Next, the involved tag t_j computes the decimal value M_j of B_j. Based on the derived value M_j, each tag t_j generates a corresponding output data sequence CRS_j and $RS_j = CRS_j||PID_j$ in which PID_j is t_j's identity ID_j except prefix value D_j and the conti-nuous i bits behind D_j. Note that the generation procedure of CRS_j is the same with that in Step 1. Finally, each involved tag simultaneously sends its computed bit string RS_j to the reader as the responses. These processes can be referred to lines 15-17 of Figure 1.

```
The operation at reader side in k-TAS
/* Transmit queries and receives tag responses */

1   Initialize stack Q, temporary stack TQ, a predefined
2      parameter i and system values m, n
3   Q = NULL
4   TQ = NULL
5   n = φ
6
7   Transmit a Start command to start current session
8       and go to line 14
9   while Q!=NULL do
10        m = Pop(Q)
11        n = Pop(TQ)
12        n = n || m
13        Transmit a Query command with prefix value n
14        Wait for tag responses RS_j=CRS_j||PID_j and
15            detect the positions of collided bits among
16            received CRS_j bit strings
17        if (there are bit collisions occurred among
18              received CRS_j bit strings) then
19              for each collided bit position
20                  retrieve CRS_j
21                  restore M_j and B_j
22                  Push (Q, B_j)
23                  Push (TQ, n)
24        else if (there is no collision occurred among
25              received CRS_j bit strings) then
26                  if (there are bit collisions occurred
27                        among received PID_j bit strings) then
28                      retrieve CRS_j and the first successive
29                          bit strings o until the 1st collision
30                          bit position among PID_j bit strings
31                      restore M_j and B_j
32                      Push (Q, B_j || o)
33                      Push (TQ, n)
34                  else if (there is no collision occurred
35                        among received PID_j bit strings) then
36                      Store the tag ID
37                  end if
38          end if
39  end while
40  Empty TQ /* release consumed memory*/
41  Transmit a Terminate command to cease current
42        session
```

Figure 2 Pseudocode of *k*-TAS: reader's operation.

Step 4: With the synchronized channel, the reader first detects all positions of collided bits among received CRS_j bit strings and recognizes all M_j values. Based on the recognized M_j values, the reader can recover the original bit block B_j values and store these recovered values (lines 20-23 of Figure 2). The reader then broadcast a *Query* command along with a new prefix value n to all tags again (lines 10-13 of Figure 2). If no bit

collision occurs among received RS_j bit strings, it denotes that the reader actually identifies one tag within its interrogation area (line 36 of Figure 2). After that, the reader goes back to other unvisited (and collided) bit positions among CRS_j identified at the last cycle and performs the collision resolution mechanism. The above identification process will be recursively operated until all existing tags have been exactly identified.

In lines 26-33 of Figure 2, we illustrate a scenario of that there is no collision occurred among received CRS_j bit strings but some bit collisions happen among received PID_j bit strings. In such case, the reader requires not only resolving currently involved block B_j values, but also retrieving the first successive bit strings o from the first bit to the first collided bit position among received PID_j bit strings. The block values B_j, which consists of current prefix value n and a bit string o retrieved from PID_j, will be maintained at reader side for next tag inquiry cycle. This design allows the reader to save more signal collision resolution steps in k-TAS by ignoring all non-collided bit position (or non-collided intermediate node in the abstract tree structure). Otherwise, for the aspect of tree structure, the reader must waste time on visiting some non-collided intermediate node without any useful feedback.

3.2. An example of k-TAS

Table 1 demonstrates a normal tag identification process of k-TAS with parameter i = 3. The identities of example tags are as follows.

Tag X, ID_X = 000001011; *Tag W*, ID_W = 000010100;

Tag Y, ID_Y = 000001000; *Tag Z*, ID_Z = 001111101.

Table 1 An example of k-TAS with i = 3

Time	Reader side	RS_X (tag X)	RS_W (tag W)	RS_Y (tag Y)	RS_Z (tag Z)
1	*Start*	00000001 001011	00000001 010100	00000001 001000	00000010 111101
Reader receives 000000xxxxxxxx					
2	*Query* and 000	00000010 011	00000100 100	00000010 000	
Reader receives 00000xx0xxx					
3	*Query* and 000001	00001000		00000001	
Reader receives 0000x00x					
4	*Query* and 000001000			*Response*	
Reader identifies tag *Y*					
5	*Query* and 000001011	*Response*			
Reader identifies tag *X*					
6	*Query* and 000010		*Response*		
Reader identifies tag *W*					
7	*Query* and 001				*Response*
Reader identifies tag *Z*					

*x means the positions of collided bits among received RS_j.

At the beginning of session S_l, the reader broadcasts a *Start* command to all tags. Once receiving this command, tag X, W, Y, and Z retrieve the first three bits from its own identity to construct B_X = 000, B_W = 000, B_Y = 000, and B_Z = 001, respectively. Next, based on the value B_j, each tag t_j calculates the corresponding output values M_j, CRS_j, and RS_j. Finally, each tag t_j send its own RS_j back to the reader simultaneously. Note that the communication channel among the reader and all existing tags is synchronized.

M_X = 0, CRS_X = 00000001,
$RS_X = CRS_X \,||\, PID_X$ = 00000001001011
M_W = 0, CRS_W = 00000001,
$RS_W = CRS_W \,||\, PID_W$ = 00000001010100
M_Y = 0, CRS_Y = 00000001,
$RS_Y = CRS_Y \,||\, PID_Y$ = 00000001001000
M_Z = 1, CRS_Z = 00000010,
$RS_Z = CRS_Z \,||\, PID_Z$ = 00000010111101

Upon obtaining the bit strings RS_X, RS_W, RS_Y, and RS_Z, the reader detects two collided bit positions, i.e., 0 and 1, among the CRS_j bit strings contained in the incoming data sequences. Note that the status of stack Q and TQ is *NULL* so far. Next, the reader retrieves two CRS_j bit strings from the result of resolving received RS_j bits sequences, and restores the corresponding values M_j and B_j. Meanwhile, the stack Q and TQ is invoked to maintain derived B_j strings for memorizing all unvisited collided bit positions.

Retrieved $CRS_j \rightarrow$ 00000001; 00000010
Restored $M_j \rightarrow$ 0; 1
Restored $B_j \rightarrow$ 000; 001
Current status of stack $Q \rightarrow$ 000; 001
Current status of stack $TQ \rightarrow \varphi$; φ

Since stack Q is not *NULL*, the reader first takes an item, i.e., m = 000 and $n = \varphi$, out of Q and TQ and produces a new system value $n = n||m$ = 000 as a prefix value for next tag ID inquiry. A *Query* command with this derived value n is then issued to all tags.

Current status of stack $Q \rightarrow$ 001
Current status of stack $TQ \rightarrow \varphi$

After getting the *Query* command with prefix value n = 000, only tags, i.e., X, W, and Y, which possess the same prefix 000 will respond. In such case, each responding tag t_j individually retrieves three continuous bits, which is behind the prefix value D_j = 000 in its ID_j, as the bit block B_j and calculates the decimal value M_j of B_j. Next, the values CRS_j and RS_j are derived. Finally, tags X, W, and Y send back the response bit sequences to the reader simultaneously.

B_X = 001, M_X = 1, CRS_X = 00000010,
$RS_X = CRS_X||PID_X$ = 00000010011
B_W = 010, M_W = 2, CRS_W = 00000100,
$RS_W = CRS_W||PID_W$ = 00000100100

B_Y = 001, M_Y = 1, CRS_Y = 00000010,
$RS_Y = CRS_Y||PID_Y$ = 00000010000

From the incoming RS_X, RS_W, and RS_Y bit strings, the reader recognizes two collided bit positions, i.e., 1 and 2, among the received CRS_j bit strings. Similarly, the reader retrieves these two identified CRS_j bit strings, i.e., 00000010 and 00000100, and maintains the corresponding values B_j, i.e., 001 and 010, into stack Q. At the same time, TQ is inserted with currently involved prefix values 000 twice.

Current status of stack $Q \rightarrow$ 001; 010; 001
Current status of stack $TQ \rightarrow$ 000; 000; φ

Due to the non-*NULL* status of stack Q, the reader retrieves two objects m = 001 and n = 000 from Q and TQ. Next, the reader calculates a new prefix value $n = n \,||\, m$ = 000001 and broadcasts it with a *Query* command to all tags.

Current status of stack $Q \rightarrow$ 010; 001
Current status of stack $TQ \rightarrow$ 000; φ

With the incoming value 000001, tags X and Y which own the same prefix value will retrieve a three bits block B_j behind the prefix value D_j = 000001 in their identities, and compute the decimal value M_j of B_j. Next, tags X and Y derive the corresponding response se-quences CRS_X, CRS_Y, RS_X, and RS_Y, and send them back to the reader at the same time.

B_X = 011, M_X = 3, CRS_X = 00001000,
$RS_X = CRS_X \,||\, PID_X$ = 00001000 || φ = 00001000
B_Y = 000, M_Y = 0, CRS_Y = 00000001,
$RS_Y = CRS_Y \,||\, PID_Y$ = 00000001 || φ = 00000001

Similarly, the reader will recognize two collided bit positions, i.e., 0 and 3, among the received CRS_j bit strings which are 00001000 and 00000001. Then, the reader computes the corresponding values B_j, i.e., 000 and 011, and maintains them in stack Q. Note that PID_X and PIX_Y are empty at the responding period. Meanwhile, current involved prefix values 000001 is maintained in TQ.

Current status of stack $Q \rightarrow$ 000; 011; 010; 001
Current status of stack $TQ \rightarrow$ 000001; 000001; 000; φ

Because stack Q is non-*NULL*, the values m = 000 and n = 000001 are extracted from Q and TQ, respectively. Next, the reader issues a prefix value $n = n \,||\, m$ = 000001000 with a *Query* command to all tags and only tag Y will respond and be identified by the reader.

Current status of stack $Q \rightarrow$ 011; 010; 001
Current status of stack $TQ \rightarrow$ 000001; 000; φ

In next step, the reader issues another new prefix value 000001011, which is constructed by the values m = 011 and n = 000001 in the top of Q and TQ, with a *Query* command to all tags. In that case, tag X is able to be recognized at current cycle.

Current status of stack $Q \rightarrow$ 010; 001
Current status of stack $TQ \rightarrow$ 000; φ

Similar to above procedures, the reader detects that stack Q is not *NULL* and then retrieves m = 010 and n = 000 from stacks Q and TQ. With the derived prefix value $n = n \,||\, m$ = 000010, the reader will identify the tag W actually.

Current status of stack $Q \rightarrow$ 001
Current status of stack $TQ \rightarrow \varphi$

Finally, the reader takes the last item, i.e., 001 and φ, from stacks Q and TQ and creates a prefix value 001 which is soon issued to all current tags. Tag Z then sends a response bit sequence 10000000101 back to the reader. With this response, the reader can actually identify tag Z.

M_Z = 7, CRS_Z = 10000000,
$RS_Z = CRS_Z \,||\, PID_Z$ = 10000000101

As the current status of stacks Q and TQ is *NULL*, the reader understands that all tags have been identified successfully. After that, the reader broadcasts a *Terminate* command to cease current tag identification session.

Current status of stack $Q \rightarrow$ *NULL*
Current status of stack $TQ \rightarrow$ *NULL*

4. Performance analyses

In this section, we analyze the identification delay and the communication overhead for recognizing all tags in k-TAS in terms of the amount of interrogation cycles and total transmitted bits [19,20,23,25]. Let $A_{r,l}$ denote the set of α tags within the reader r's interrogation range during the lth tag identification session S_l. The identification delay, i.e., $d_{\text{total}}(A_{r,l})$, caused by recognizing $A_{r,l}$ is as follows. Note that d_{reader} is the delay of transmission time of the reader's *Query* command including any appended information, d_{tag} is the delay of delivering the tag ID, d_{cycle} is the average delay of an interrogation cycle and $T(A_{r,l})$ is the total interrogation cycles in a session when the reader r recognizes $A_{r,l}$.

$$d_{\text{total}}(A_{r,l}) = \sum (d_{\text{reader}} + d_{\text{tag}}) \approx T(A_{r,l}) \cdot d_{\text{cycle}} \qquad (1)$$

Lemma 1. The number of collided cycles in the β th layer of target abstract k-ary tree structure when k-TAS recognizes α tags in $A_{r,l}$, $C_{k\text{-ary}}(\alpha,\beta)$, is

$$C_{k\text{-ary}}(\alpha,\beta) = k^{\beta}\left[\left(1-\frac{1}{k^{\beta}}\right)^{\alpha} - \frac{\alpha}{k^{\beta}}\cdot\left(1-\frac{1}{k^{\beta}}\right)^{\alpha-1}\right]$$

Proof: Let $I_{k\text{-ary}}(\alpha,\beta)$ and $R_{k\text{-ary}}(\alpha,\beta)$ be the number of idle cycles and readable cycles, respectively, in the β th layer of target abstract k-ary tree structure when k-TAS recognizes $A_{r,l}$. As the total nodes in the β th layer of target k-ary tree is k^{β}, $I_{k\text{-ary}}(\alpha,\beta)$ and $R_{k\text{-ary}}(\alpha,\beta)$ can be derived as follows.

$$I_{k\text{-ary}}(\alpha,\beta) = k^{\beta}\cdot\left(1-\frac{1}{k^{\beta}}\right)^{\alpha} \tag{2}$$

$$R_{k-\text{ary}}(\alpha,\beta) = \alpha\cdot\left(1-\frac{1}{k^{\beta}}\right)^{\alpha-1} \tag{3}$$

Hence, from (2) to (3), we obtain

$$\begin{aligned} C_{k-\text{ary}}(\alpha,\beta) &= k^{\beta} - I_{k-\text{ary}}(\alpha,\beta) - R_{k-\text{ary}}(\alpha,\beta) \\ &= k^{\beta} - k^{\beta}\cdot\left(1-\frac{1}{k^{\beta}}\right)^{\alpha} - \alpha\cdot\left(1-\frac{1}{k^{\beta}}\right)^{\alpha-1} \\ &= k^{\beta}\left[1-\left(1-\frac{1}{k^{\beta}}\right)^{\alpha} - \alpha\cdot\frac{1}{k^{\beta}}\cdot\left(1-\frac{1}{k^{\beta}}\right)^{\alpha-1}\right] \end{aligned}$$

Lemma 2. Let $C_{k\text{-ary}}(A_{r,l})$ be the number of collided nodes of k-TAS for recognizing $A_{r,l}$. Then, the number of required interrogation cycles when r recognizes $A_{r,l}$ under a k-ary tree structure, $T_{k\text{-ary}}(A_{r,l})$, is

$$T_{k-\text{ary}}(A_{r,l}) = \alpha + C_{k-\text{ary}}(A_{r,l})$$

Proof: Since k-TAS always splits the set of currently collided tags into k subsets, the whole tag identification procedure of k-TAS can be represented as a k-ary tree when the reader r recognizes $A_{r,l}$. It is obvious that in k-TAS all idle cycles are ignored. Therefore, in the target k-ary tree corresponded to k-TAS, all the intermediate nodes are collided nodes and all the leaf nodes are readable nodes.

Theorem 1. For any α, $T_{k-\text{ary}}(A_{r,l}) = \alpha + \sum_{\beta=0}^{\chi_{/i}-1} k^{\beta}\left[1-\left(1-\frac{1}{k^{\beta}}\right)^{\alpha} - \frac{\alpha}{k^{\beta}} - \left(1-\frac{1}{k^{\beta}}\right)^{\alpha-1}\right]$, where $i = \log_2 k$, and χ is the bit-length of each tag ID.

Proof: From Lemmas 1 and 2, the number of collided nodes while k-TAS recognizes $A_{r,l}$, $C_{k\text{-ary}}(A_{r,l})$, can be derived as

$$\begin{aligned} C_{k\text{-ary}}(A_{r,l}) &= \sum_{\beta=0}^{\chi_{/i}-1} C_{k\text{-ary}}(\alpha,\beta) \\ &= \sum_{\beta=0}^{\chi_{/i}-1} k^{\beta}\left[1-\left(1-\frac{1}{k^{\beta}}\right)^{\alpha} - \frac{\alpha}{k^{\beta}}\cdot\left(1-\frac{1}{k^{\beta}}\right)^{\alpha-1}\right] \end{aligned}$$

Therefore,

$$T_{k\text{-ary}}(A_{r,l}) = \alpha + \sum_{\beta=0}^{\chi_{/i}-1} k^{\beta}\left[1-\left(1-\frac{1}{k^{\beta}}\right)^{\alpha} - \frac{\alpha}{k^{\beta}}\cdot\left(1-\frac{1}{k^{\beta}}\right)^{\alpha-1}\right]$$

Theorem 2. Let $S_{k\text{-ary}}(A_{r,l})$ be the amount of transmitted bits transmitted by the reader and all tags for recognizing $A_{r,l}$ in k-TAS. Then, the total transmitted bits when r recognizes $A_{r,l}$ under a k-ary tree structure, $S_{k\text{-ary}}(A_{r,l})$, is

$$S_{k-ary}(A_{r,l}) = \sum_{\beta=0}^{\chi_{/i}-1} (2^{i}+\chi-i)\cdot k^{\beta}\cdot\left[1-\left(1-\frac{1}{k^{\beta}}\right)^{\alpha}\right],$$

where $i = \log_2 k$, and χ is the bit-length of each tag ID.

Proof: Let $S_{k\text{-ary}}(\alpha,\beta)$ be the amount of transmitted bits collected from the depth β th layer of target abstract k-ary tree structure when k-TAS recognizes $A_{r,l}$. As mentioned before, the nature of k-TAS always makes the corresponding abstract k-ary tree structure possesses only collided nodes and readable nodes. In k-TAS, for each reader inquiry and each tag response, the transmitted bits is $i{\cdot}\beta$ and $2^{i}+(\chi\text{-}i{\cdot}\beta\text{-}i)$, respectively. With the results of Lemma 1 and Equation 3, we obtain

$$\begin{aligned} S_{k-ary}(\alpha,\beta) &= \{i\cdot\beta + [2^{i}+(\chi - i\cdot\beta - i)]\}\cdot[R_{k-\text{ary}}(\alpha,\beta) + C_{k-\text{ary}}(\alpha,\beta)] \\ &= (2^{i}+\chi-i)\cdot k^{\beta}\cdot\left[1-\left(1-\frac{1}{k^{\beta}}\right)^{\alpha}\right] \end{aligned}$$

Therefore,

$$\begin{aligned} S_{k-ary}(A_{r,l}) &= \sum_{\beta=0}^{\chi_{/i}-1} S_{k-ary}(\alpha,\beta) \\ &= \sum_{\beta=0}^{\chi_{/i}-1} (2^{i}+\chi-i)\cdot k^{\beta}\cdot\left[1-\left(1-\frac{1}{k^{\beta}}\right)^{\alpha}\right] \end{aligned}$$

In brief summary, we learn from Theorems 1 and 2 that the total number of required interrogation cycles $T_{k\text{-ary}}(A_{r,l})$ and the total amount of transmitted bits $S_{k\text{-ary}}(A_{r,l})$ for k-TAS are mainly dependent on the bit length of tag ID χ, the value of k and the number of tags α. When the number of tags is large enough, for example α = 200, and the bit length of tag ID is long enough, for example χ = 96, then the main performance difference between 2-ary protocols (i = 1) and k-ary protocol (k-TAS) where $k = 2^{i}$ and $i \geq 2$ are dependent on the computation of $\sum_{\beta=0}^{\chi_{/i}-1} k^{\beta}$. It is obvious that the final value of $\sum_{\beta=0}^{\chi_{/i}-1} k^{\beta}$ is much smaller when $i \geq 2$ in comparison with 2-ary protocols (i = 1).

5. Performance evaluation

In this section, we evaluate the performance of k-TAS in comparison with existing tree-based tag identification methods, i.e., Query Tree (QT) [13,14,21,27], BS [16,34], RN16QT [17], and ETIP [38], which are most relevant to k-TAS. The main criteria for evaluating the efficiency during a normal tag identification session are the identification delay and communication overhead [19,20,23-25]. Here, we utilize the total interrogation cycles required in a tag recognition session to represent the identification delay. In addition, the metric of the number of bits transmitted by the reader and the tags in

Table 2 Comparison among k-TAS and other relevant studies

	Assumption & collision detection technique	Collision resolution mechanism	Performance comparison	
			Identification delay*	Communication overhead
k-TAS	Transmission, synchronization & bit-by-bit collision detection	*k*-splitting	Low ($\log_k(n)$)	Low
ETIP [38]		*k*-splitting	Low ($\log_k(n)$)	High
BS [16,34]		2-splitting	Medium ($\log_2(n)$)**	High
RN16QT [17]	Additional tag Memory for randomly generated prefixes & bit-by-bit collision detection	2-splitting	High ($\log_2(n)$)	High
Query tree (QT) [13,17,21,27,32]	Bit-by-bit collision detection	2-splitting	High ($\log_2(n)$)	High

*n is the number of tags.

**BS gains a better performance than QT-based methods by eliminating all idle cycles.

a session is critical for influencing the system performance of a tree-based tag identification protocol. This important metric usually can be denoted as the communication overhead. Our simulation was written with C# under Visual Studio .NET environment. For simplicity, event dispatcher is implemented in our simulation tool to dispatch reader commands and tag responses as events. The reader and each tag have their own event queue to store events dispatched to it. All events are generated with timestamp. A global timer is implemented. The reader gathers all tag response events occurred at the same timestamp for further data synchronization and signal collision detection. First, we investigate the effect of the parameter i on the performance of k-TAS in which the best candidate value for i is derived. Second, we demonstrate that the performance of k-TAS is superior to existing tree-based tag identification schemes under various numbers of tags and bit-lengths of tag ID.

5.1. Impact of the system parameter *i*

Figure 3 demonstrates that the identification delay of k-TAS under different number of tags and various parameters i in terms of total interrogation cycles. In general, as the number of tags increases, the collided cycles will raise and this scenario causes longer identification delay. Since k-TAS does not possess any idle cycles, the factor of the number of collided cycles will play a significant role to influence the performance of k-TAS. Note that the readable cycles in k-TAS is always equal to the number of the existing tags. More concretely, compared to traditional binary tree-based protocols, k-TAS can resolve tag signal collision more effectively, i.e., splitting currently collided tags set into k subsets instead of only two subsets. In addition, k-TAS can understand why the collision happens at each collided cycle, i.e., which descendant nodes collide with currently visited node in the abstract k-ary tree structure. This design allows the reader to avoid all idle scenarios. Hence, as shown in Figure 3, k-TAS can save at least 38.8 and 12% interrogation cycles when comparing to QT and BS, respectively. As the value of parameter i increases, the improvement is more significant. In Figure 4, we find that QT, BS, and k-TAS (i = 5) possess the same level of performance owing to similar amount of

Table 3 Improved ratio of the number of interrogation cycles in *k*-TAS under different bit-length of tag ID and different number of tags

Compared target	Bit-length of tag ID	Performance improvement of *k*-TAS
QT	64 bits	Reduce around 39.7-50.5% interrogation cycles
	96 bits	Reduce around 39.8-50.2% interrogation cycles
	256 bits	Reduce around 39.6-49.5% interrogation cycles
BS	64 bits	Reduce around 12.8-27.9% interrogation cycles
	96 bits	Reduce around 11.8-28.6% interrogation cycles
	256 bits	Reduce around 13.4-26.6% interrogation cycles
RN16QT	64 bits	Reduce around 37.9-49.8% interrogation cycles
	96 bits	Reduce around 36.1-47% interrogation cycles
	256 bits	Reduce around 38.4-49% interrogation cycles
ETIP	64 bits	Almost the same interrogation cycles
	96 bits	Almost the same interrogation cycles
	256 bits	Almost the same interrogation cycles

Table 4 Improved ratio of the amount of transmitted bits in *k*-TAS under different bit-length of tag ID and different number of tags

Compared target	Bit-length of tag ID (bits)	Performance improvement of *k*-TAS
QT	64	Reduce around 27.8-39.6% transmitted bits
	96	Reduce around 19.5-29.3% transmitted bits
	256	Reduce around 8.8-13% inter transmitted bits
BS	64	Reduce around 27.3-39% transmitted bits
	96	Reduce around 17.6-29.9% transmitted bits
	256	Reduce around 8.9-13.1% transmitted bits
RN16QT	64	Reduce around 36.9-46.5% transmitted bits
	96	Reduce around 26.5-35.7% transmitted bits
	256	Reduce around 12.7-17.1% transmitted bits
ETIP	64	Reduce around 27.7-36.4% transmitted bits
	96	Reduce around 18.8-27.6% transmitted bits
	256	Reduce around 8.5-12.5% transmitted bits

transmitted bits. Only *k*-TAS with i = 2, 3, or 4 can outperform QT and BS, where at least 9.8-29.3% of transmitted bits can be eliminated. These results lead to a conclusion of that the best candidate value for i should be 2 or 3 due to the system efficiency tradeoff between the total interrogation cycles and the amount of transmitted bits. Note that i = 4 is an acceptable value, however, it cannot provide the best performance due to the huge transmitted bits. In the next section, we present the comparisons among *k*-TAS (i = 2 or 3), QT, BS, ETIP, and RN16QT with different number of tags in our simulation program.

5.2. The performance comparisons among *k*-TAS and other relevant studies

Figures 5 and 6 show the performance comparison among *k*-TAS and other relevant tree-based tag identification protocols [13,14,16,17,21,27,34] in terms of the total interrogation cycles and the amount of transmitted bits. From Figure 5, we know that *k*-TAS requires fewer interrogation cycles for identifying all existing tags than other related methods. The improvement is significant as *k*-TAS eliminates around 39.8-50%, 11.8-28.6%, and 36.1-47% of interrogation cycles in comparison with QT, BS, and RN16QT, respectively. This result is because *k*-TAS reduces a significant number of collided cycles by arbitrating signal collision with *k*-splitting technique instead of two-splitting mechanism adopted in QT, BS, and RN16QT. This nature allows *k*-TAS to outperform other relevant studies during a collision resolution procedure on the aspect of the interrogation cycles. On the other hand, since *k*-TAS and ETIP both utilize *k*-splitting-based arbitration to resolve each signal collision, their protocol efficiency on identi-fication delay are quite similar, i.e., the total interrogation cycles required in *k*-TAS is almost the same as ETIP. Next, as *k*-TAS uses a series of synchronized challenge-response bit sequences, such as CRS_j and RS_j, to communicate with the tags, all idle cycles in a tag identification process can be eliminated as ETIP and BS do. This advantage makes *k*-TAS, ETIP, and BS more efficient than QT and its variant RN16QT. In other words, since *k*-TAS, ETIP, and BS can ignore all idle cycles instead of wasting time to visit them as QT-based scheme does, the improvement on identification efficiency is promised.

Figure 6 presents the comparison among *k*-TAS and related studies in terms of the amount of transmitted

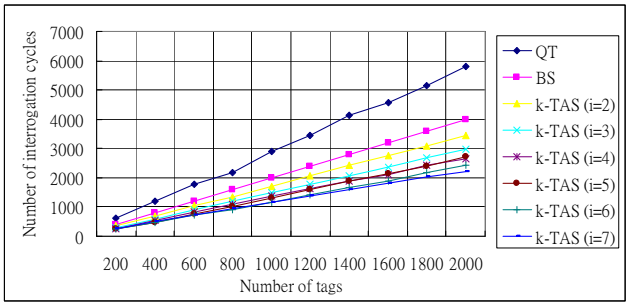

Figure 3 The identification delay of *k*-TAS with different *i*. Note that the length of ID is 96 bits.

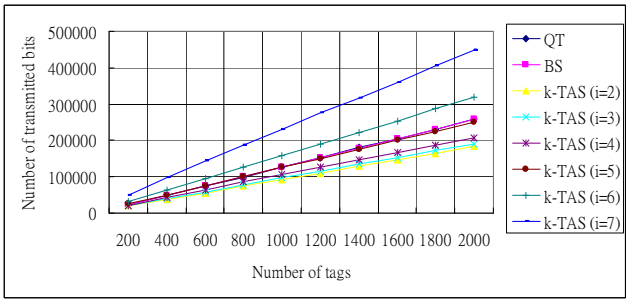

Figure 4 The communication overhead of *k*-TAS with different *i*. Note that the length of ID is 96 bits.

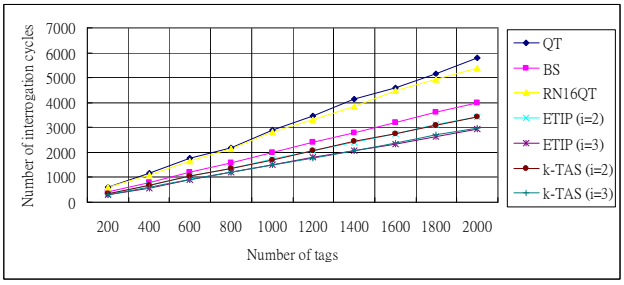

Figure 5 The identification delay of *k*-TAS compared to other relevant studies in which the length of ID is 96 bits.

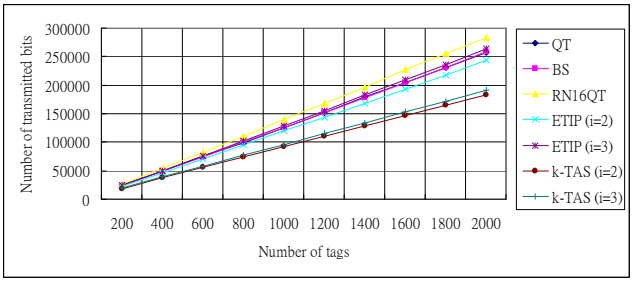

Figure 6 The communication overhead of *k*-TAS compared to other relevant studies in which the length of ID is 96 bits.

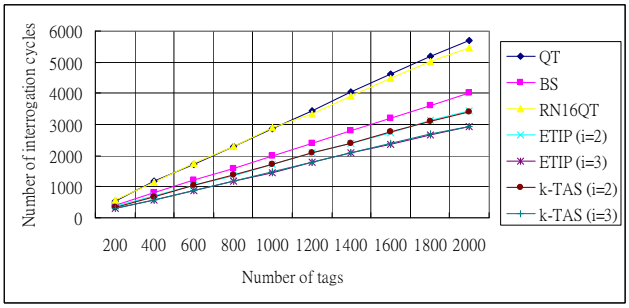

Figure 7 The identification delay of *k*-TAS compared to other relevant studies in which the length of ID is 64 bits.

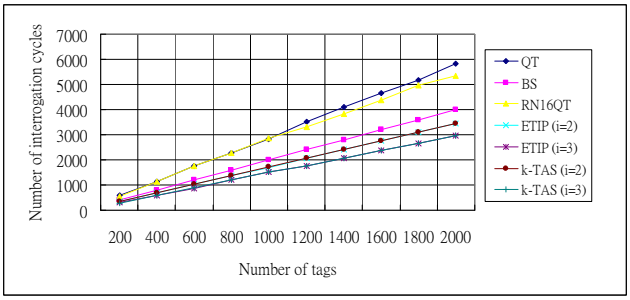

Figure 8 The identification delay of *k*-TAS compared to other relevant studies in which the length of ID is 256 bits.

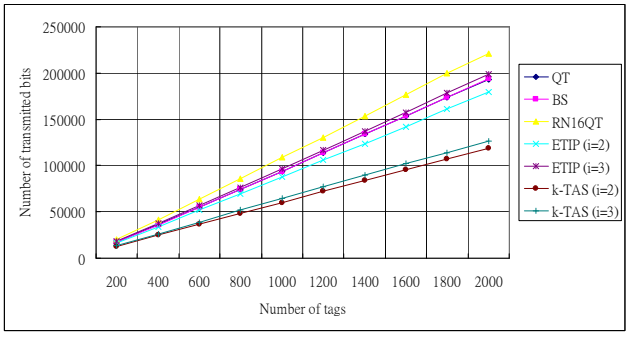

Figure 9 The communication overhead of *k*-TAS compared to other relevant studies in which the length of ID is 64 bits.

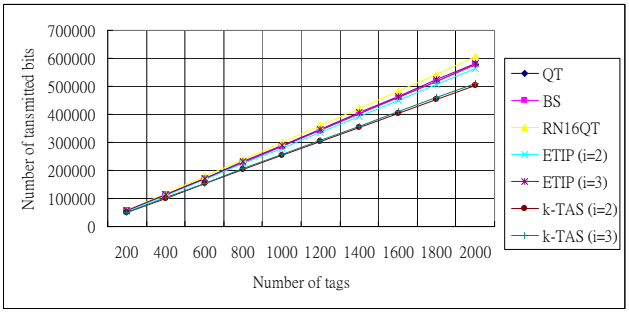

Figure 10 The communication overhead of *k*-TAS compared to other relevant studies in which the length of ID is 256 bits.

bits. Obviously, *k*-TAS significantly outperforms QT, BS, RN16QT, and ETIP by reducing around 19.5-29.3%, 17.6-29.9%, 26.5-35.7%, and 18.8-27.6% transmitted bits, respectively, in total. In comparison with QT, BS, and RN16QT, as less collided cycles are required in *k*-TAS, the corresponding transmitted bits collected from all collided cycles are less. This enhancement is from the efficient *k*-splitting scheme. Next, we find that the system efficiency of ETIP is not well-performed during tag signal responses collection and resolution. Therefore, the communication overhead of ETIP is large. This limitation inspires us to re-design a novel and efficient tag response collection mechanism to gain better protocol performance. In addition, since no idle cycle exists in *k*-TAS, ETIP, and BS, the number of corresponding transmitted bits from idle cycle is zero in these three protocols, and the efficiency on communication overhead is guaranteed.

Table 2 introduces the main characteristics and performance comparison among *k*-TAS and other related protocols. The second column of Table 2 presents the assumption of each method and corresponding collision detection technique. Obviously, the feasibility of *k*-TAS is reasonable as BS protocol does. Regarding the collision resolution scheme in the third column, *k*-TAS and ETIP

both exploit efficient k-splitting mechanism while the other methods adopt inefficient 2-splitting scheme. On the aspect of identification delay and communication overhead, k-TAS obviously outperforms other methods and is thus more practical for real world application.

5.3. Impact of bit-length of tag ID

This section describes the performance improvement of k-TAS under different bit-length of tag ID and different number of tags. From Figures 5, 7, and 8 and Table 3 it is obvious that the performance improvement of k-TAS is independent with the factor of tag ID's bit-length on the aspect of the identification delay, i.e., the number of interrogation cycles. We infer that tree-based tag identification protocol can identify all tags without visiting the leaf node of abstract tree structure in a tag inquiry session when the number of tags is not many. Nevertheless, as shown in Figures 6, 9, and 10 and Table 4, the total transmitted bits are significantly increased and the performance improvement of k-TAS is decreased as the bit-length of tags ID becomes larger. This is because the amount of transmitted bits collected from all readable cycles becomes a significant part compared to the total transmitted bits in k-TAS when the bit-length of tag ID is longer. In that case, the effect of the transmitted bits collected from all collided cycles is comparatively smaller than that collected from all readable cycles while evaluating the communication overhead in k-TAS. Note that no idle cycles exist in a tag identification process of k-TAS.

6. Conclusions

Collision resolution is critical for efficiency improvement in implementing an RFID tag identification protocol. In this article, we present a novel anti-collision protocol, i.e., k-TAS, for identifying passive tags in which an adaptive collision resolution scheme is adopted to make whole tag identification process more efficient. The nature of k-TAS exploits the information on whether currently visited node is collided and, if yes, which descendant nodes collide with it. Such information is extracted from all tags' response data at each inquiry time period. This design successfully avoids wasting time on many collided cycles and all idle cycles in k-TAS and accordingly increases the system throughput during tag identification procedure. Next, we analyze the system performance of k-TAS in both theory and simulations, which show that our proposed schemes significantly reduce identification delay and communication overhead at each tag identifying process. In the future, we would like to design a dynamic collision resolution mechanism for tag identification in which the system parameter i could be adjusted dynamically to pursue better protocol performance. Moreover, to identify whether an optimal value of parameter i exists and is suitable for any RFID application environment is another interesting research topic for us.

Acknowledgements

The authors gratefully acknowledge the support from the Taiwan Information Security Center (TWISC) and the National Science Council, Taiwan, under the Grant nos. NSC 100-2219-E-011-002, NSC 100-2218-E-011-005, and NSC 100-2218-E-034-001-MY2.

Author details

[1]Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan [2]Department of Information Management, Chinese Culture University, Taipei 111, Taiwan [3]School of Information Systems, Singapore Management University 178902, Singapore

Competing interests

The authors declare that they have no competing interests.

Received: 28 February 2011 Accepted: 26 October 2011
Published: 26 October 2011

References

1. I Chamtac, C Petrioli, J Redi, Energy-conserving access protocols for identification networks. IEEE/ACM Trans Netw. **7**(1), 51–59 (1999). doi:10.1109/90.759318
2. V Namboodiri, L Gao, Energy-aware tag anti-collision protocols for RFID systems. IEEE Trans Mobile Comput. **9**(1), 44–59 (2010)
3. T Cheng, L Jin, Analysis and simulation of RFID anti-collision algorithms, in *Proceedings of 9th International Conference on Advanced Communication Technology*, 697–701 (2007)
4. *EPC™ Radio-Frequency Identification Protocols Class 1 Generation-2 UHF RFID Protocol for Communication at 860-960 MHz Version 1.0.9* (EPCGlobal Inc., December 2005)
5. C Floerkemeier, M Wille, Comparison of transmission schemes for framed ALOHA based RFID protocols, in *Proceedings of International Symposium on Applications and the Internet Workshops*, 92–97 (2006)
6. *Information Technology-Radio Frequency Identification for Item Management-Part 6: Parameters for Air Interface Communications at 860 MHz to 960 MHz, Amendment 1: Extension with Type C and Update of Types A and B, ISO/IEC 18000-6:2004/Amd. 1:(E)* (June 2006)
7. DK Klair, KW Chin, R Raad, On the suitability of framed slotted aloha based RFID anti-collision protocols for use in RFID-enhanced WSNs, in *Proceedings of 17th International Conference on Computer Communications and Networks*, 583–590 (2007)
8. DK Klair, KW Chin, A novel anti-collision protocol for energy efficient identification and monitoring in RFID-enhanced WSNs, in *Proceedings of 17th International Conference on Computer Communications and Networks*, 1–8 (2008)
9. Y Maguire, R Pappu, An optimal Q-algorithm for the ISO 18000-6C RFID protocol. IEEE Trans Autom Sci Eng. **6**(1), 16–24 (2009)
10. CP Wong, Q Feng, Grouping based bit-slot ALOHA protocol for tag anti-collision in RFID systems. IEEE Commun Lett. **11**(12), 946–948 (2007)
11. L Zhu, TSP Yum, Optimal framed aloha based anti-collision algorithms for RFID systems. IEEE Trans Commun. **58**(12), 3583–3592 (2010)
12. B Li, J Wang, Efficient anti-collision algorithm utilizing the capture effect for ISO 18000-6C RFID protocol. IEEE Commun Lett. **15**(3), 352–354 (2011)
13. *860 MHz - 930 MHz Class 1 Radio Frequency Identification Tag Radio Frequency and Logical Communication Interface Specification Candidate Recommdation Version 1.0.1*, Auto-ID Center, (2002)
14. KW Chiang, C Hua, TS Peter Yum, Prefix-randomized query-tree protocol for RFID systems, in *Proceedings of IEEE International Conference on Communications*, 1653–1657 (2006)
15. JS Cho, JD Shin, SK Kim, RFID tag anti-collision protocol: query tree with reversed IDs, in *Proceedings of 10th International Conference on Advanced Communication Technology*, 225–230 (2008)
16. HS Choi, JR Cha, JH Kim, Fast wireless anti-collision algorithm in ubiquitous ID system, in *Proceedings of IEEE 60th Vehicular Technology Conference*, 4589–4592 (2004)

17. JH Choi, D Lee, H Lee, Query tree-based reservation for efficient RFID tag anti-collision. IEEE Commun. Lett. **11**(1), 85–87 (2007)
18. *Draft Protocol Specification for a 900 MHz Class 0 Radio Frequency Identification Tag*, Auto-ID Center, (2003)
19. YC Lai, CC Lin, A pair-resolution blocking algorithm on adaptive binary splitting for RFID tag identification. IEEE Commun Lett. **12**(6), 432–434 (2008)
20. YC Lai, CC Lin, Two blocking algorithms on adaptive binary splitting: single and pair resolutions for RFID tag identification. IEEE/ACM Trans Netw. **17**(3), 962–975 (2009)
21. C Law, K Lee, KY Siu, Efficient memoryless protocol for tag identification, in *Proceedings of the 4th International Workshop on Discrete Algorithm and Methods for Mobile Computing and Communication*, 75–84 (2000)
22. L Liu, ZH Xie, JT Xi, SL Lai, An improved anti-collision algorithm in RFID system, in *Proceedings of 2nd International Conference on Mobile Technology, Applications and Systems*, 1–5 (2005)
23. J Myung, W Lee, J Srivastava, Adaptive binary splitting for efficient RFID tag anti-collision. IEEE Commun Lett. **10**(3), 144–146 (2006). doi:10.1109/LCOMM.2006.1603365
24. J Myung, W Lee, TK Shih, An adaptive memoryless protocol for RFID tag collision arbitration. IEEE Trans Multimedia **8**, 1096–1101 (2006)
25. J Myung, W Lee, J Srivastava, TK Shih, Tag-splitting: adaptive collision arbitration protocols for RFID tag identification. IEEE Trans Parallel Distrib Syst. **18**(6), 763–775 (2007)
26. HS Ning, Y Cong, ZQ Xu, T Hong, JC Chao, Y Zhang, Performance evaluation of RFID anti-collision algorithm with FPGA implementation, in *Proceedings of 21st International Conference on Advanced Information Networking and Applications Workshops*, 153–158 (2007)
27. TP Wang, Enhanced binary search with cut-through operation for anti-collision in RFID systems. IEEE Commun Lett. **10**(4), 236–238 (2006). doi:10.1109/LCOMM.2006.1613732
28. T La Porta, G Maselli, C Petrioli, Anti-collision protocols for single-reader RFID systems: temporal analysis and optimization. IEEE Trans Mobile Comput. **10**(2), 267–279 (2011)
29. XL Jia, QY Feng, CZ Ma, An efficient anti-collision protocol for RFID tag identification. IEEE Commun Lett. **14**(11), 1014–1016 (2010)
30. F Schoute, Dynamic frame length aloha. IEEE Trans Commun. **31**(4), 565–568 (1983). doi:10.1109/TCOM.1983.1095854
31. JE Wieselthier, A Ephremides, LA Michaels, An exact analysis and performance evaluation of framed aloha with capture. IEEE Trans Commun. **37**(2), 125–137 (1989). doi:10.1109/26.20080
32. JI Capettanakis, Tree algorithms for packet broadcast channels. IEEE Trans Inf Theory **25**, 505–515 (1979). doi:10.1109/TIT.1979.1056093
33. J Mosely, P Humblet, A class of efficient contention resolution algorithms for multiaccess channels. IEEE Trans Commun. **33**(2), 145–151 (1985)
34. K Finkenzeller, *RFID Handbook: Radio-Frequency Identification, Fundamentals and Applications*, (Wiley, 1999)
35. *ISO/IEC 14443. Inentification cards–Contactless integrated circuit cards–Proximity cards*
36. *ISO/IEC 15693. Inentification cards–Contactless integrated circuit cards–Vicinity cards*
37. *ISO/IEC 18000. ISO/IEC 18000 Information Technology AIDC Technologies–RFID for Item Management–Air Interface*
38. KH Yeh, NW Lo, E Winata, An efficient tree-based tag identification protocol for RFID systems, in *Proceedings of 22nd International Conference on Advanced Information Networking and Applications Workshops*, 966–970 (2008)

doi:10.1186/1687-1499-2011-139
Cite this article as: Chen *et al.*: Adaptive collision resolution for efficient RFID tag identification. *EURASIP Journal on Wireless Communications and Networking* 2011 **2011**:139.