International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 03 | Mar 2019 www.irjet.net p-ISSN: 2395-0072

An Energy-Oriented Collision Arbitration Protocol for Passive RFID Tag Identification

Kyongjin Sok[1], Prof. Gon Hong[1], Kwangnam Choe[2], Kinam Sin[3], Changho Kim[4], Chongil Rim[5]

[1]Institute of Information Technology, University of Sciences, Pyongyang, Democratic People's Republic of Korea
[2]Department of Light Industry Machinery Engineering, Pyongyang University of Mechanical Engineering, Pyongyang, Democratic People' Republic of Korea
[3]Faculty of Mathematics, Kim Il Sung University, Pyongyang, Democratic People's Republic of Korea
[4]Department of Information Technology, Kim Hyong Jik University, Pyongyang, Democratic People's Republic of Korea
[5]Department of Automation Engineering, Pyongyang University of Mechanical Engineering, Pyongyang, Democratic People's Republic of Korea

---

Abstract - *Portable readers such as smartphones and handheld devices are becoming increasingly popular in a wide range of RFID applications due to the relatively easy and inexpensive way to collect data. In order to expand the battery life of the reader, efficient tag identification protocols are indispensable in the large-scale passive RFID systems. In the conventional tree-based protocols, an excessive collision and a huge waste of transmitting bits lead to considerable performance degradation. In this study, an improved M-ary query collision tree protocol (IMQCT) was proposed. In order to restrict the length of bits transmitted by tags, a fixed-window procedure was applied to M-ary query tree protocol, without calculating and transmitting the window size in each query cycle. As a result of the simulation, the proposed protocol was proved to be a considerably improved protocol in energy and time savings, compared to the existing tree-based protocols.*

Key Words: RFID, anti-collision, tag identification, tree-based, window methodology

1. INTRODUCTION

Radio frequency identification (RFID), a non-contact automatic identification technology through radio frequency signal, is vital to the implementation of Internet of Things (IoT). With the rapid development of IoT technology, RFID technology has been more and more useful in our life and industry [1-3]. Because passive RFID tags have very limited computation and communication capabilities, how to quickly identify all tags is a challenging problem [1]. Besides, energy saving for portable readers such as smartphone and handheld device is also an important issue [2]. This is because, in passive RFID systems, the reader needs to not only supply energy for its own operations, but also power up all tags in its interrogation area. In RFID systems, a tag collision, in which a reader cannot identify any tag, may occur when multiple tags try to respond to the reader at the same time. The occurrence of such collisions causes the tags to retransmit their messages in the subsequent query; therefore, it can not only elongate the tag identification time but also increase the energy consumption at the reader [1].

So far, many anti-collision protocols have been proposed to solve such a collision problem, and they can be classified into three broad categories: ALOHA-based, tree-based and hybrid protocols. In general, ALOHA-based protocols show a good performance in case that the number of tags is relatively small due to the time slot used for tag identification [1-6]. However, ALOHA-based protocols have a tag starvation problem and suffer a considerable degradation of performance in large-scale systems. Tree-based protocols have the advantage of successfully identifying all the tags even when the number of tags in the interrogation area is enormous [1-3, 12-20]. Finally, hybrid protocols combine the advantages of ALOHA and tree protocols. Hybrid protocols usually perform better at the expense of higher hardware and software complexity [2, 7].

The bit-tracking technology based on Manchester code, which allows the reader to identify the locations of the collided bit, have been widely used in tree-based protocols, such as collision tree (CT) [14], k -Ary tree-based anti-collision scheme (k -TAS) [15], M -ary query tree (MQT) [16] and collision window tree (CwT) [3, 18, 19] protocols. In general, the tree-based protocols such as query tree (QT) [12] and CT transmit two queries which differ only in the last bit. In k -TAS and MQT, multi-bit arbitration is performed once per cycle using a mapping function; therefore, it can reduce the number of query cycles as compared to the binary tree protocols [15, 16]. As a result of the comparison between several ALOHA-based and tree-based protocols, k -TAS showed the best performance regarding the interrogation cycle and the time required for the identification of all tags [17].

Anti-collision protocols for active RFID systems considering energy efficiency have been proposed in several papers [8-11]. However, the energy consumed in the passive RFID systems has not yet been extensively studied. Recently, the increasing number of RFID systems that use handheld or portable devices requires the energy

saving of the readers. Moreover, the used anti-collision protocol affects the reader's power consumption, particularly in a large-scale system. In general, the energy cost of passive systems is related to two factors: the number of query cycles and the time spent on transmitting information between the reader and tags [1, 3]. Therefore, to save the reader's energy, both the number of query cycles and the number of transmitted bits must be reduced as low as possible. In previous research, most existing tree-based protocols focus on eliminating or reducing idle and collision slots.

Window-based methodologies, i.e., Query window tree (QwT), collision window tree (CwT) and flexible query window tree (FQwT), are attractive choices, which can limit the length of tag response in passive RFID system [18-20]. Window-based protocols effectively reduce the reader's energy with the help of heuristic window procedures. However, this benefit is accompanied by the addition of the number of slots due to the introduction of a new type of slot, go-on slot.

In this study, a new protocol called improved M-ary query collision tree (IMQCT) protocol was proposed by applying a fixed-window procedure to M-ary query tree protocol with memoryless and m-bits arbitration. The fixed-window methodology can limit the length of the tag response, without calculating and transmitting the window size in each query cycle. As a result of the comparison with the several existing tree-based protocols, the proposed protocol showed better performance mainly in terms of energy and time savings. The content of this paper is arranged as follows. Section 2 describes the bit window methodology and MQT protocol. The IMQCT protocol is presented in Section 3. Section 4 shows the simulation results compared with the several existing tree-based protocols. Finally, the conclusions are drawn in Section 5.

2. RELATED WORK

2.1 Bit tracking and MQT protocol

At first, the bit-tracking technology is briefly explained for a better understanding of the proposed protocol. Manchester coding has got very useful in the numerous protocols for RFID tag identification because it can accelerate the identification process due to its capability of detecting the locations of collided bits [14-16]. In addition, this coding method is specified in the ISO/IEC 14443 standard. However, it requires all tags within the readers reading range to transmit their data synchronously. In detail, 0 and 1 are logically encoded by the positive and negative transitions of the voltage level, respectively. If the bits with different values (0 and 1) are transmitted simultaneously by more than two tags, the transitions (positive and negative) of the received bits do not conform to the coding rules as shown in Figure 1; as a result, the location of collision bits can be detected.

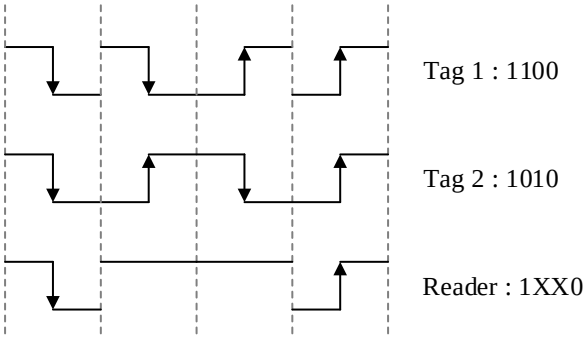


Figure 1. Example of Manchester coding. "X" denotes a collision bit.

In the traditional tree-based protocols including BS and QT, the combined response of tags is processed bit by bit. In case that the current bit is readable, the reader moves to the next bit after directly identifying the bit. In case of the bit on collision state, the reader splits the bit into 0 and 1 and re-interrogates with the two new queries. The overall process can be explained as a binary tree. In contrast, MQT can perform m-bits arbitration at a time because MQT is not based on a binary tree but on an M-ary tree [16], in which M is 2^m, and furthermore, Manchester coding is capable of tracking a collision to an individual bit. The mapping example for 2 bits is shown in **Table** 1.

Table 1. Example of the mapping table

2 bits in Tag ID	4 bits mapped string
00	0001
01	0010
10	0100
11	1000

The tags which have IDs matched with the query prefix messaged from the reader will respond with the rest part except for the matched part of their IDs. At this time, the first m-bits in the rest ID will be mapped into M-bits. In this paper, this query type is called *mapping-rest query*, designated simply as *mrq*. M-bits string points out which child node of the M-ary tree the tag belongs to. When p designates the number of collision bits in the mapping part, the reader generates new p queries ($qr_1,...,qr_p$), where, p sets of m-bits ($r_1,...,r_p$) can be inversely mapped by using the mapping table or mapping function. An example of the 4-ary tree formed in the identification process is illustrated in Figure 2, and the communication procedure is shown in Table 2. In Figure 2, '10' is an empty node. After receiving the query prefix '00', two tags A (0001 0111) and B (0011 1010) respond with 0010 0111 and 1000 1010, respectively, which produce *0*0 **1* (#2 of Table 2). By using the previous prefix '00' and the received mapping part *0*0 ($p=2$), the reader prepares two new queries '0001' and '0011' ($r_1=01$ and $r_2=11$). M-ary query protocol has two advantages over the conventional query

trees: (i) dividing collision tags into not two but many subgroups (M) so that the total number of queries can be reduced and (ii) identifying empty nodes so that the idle slot can be eliminated.

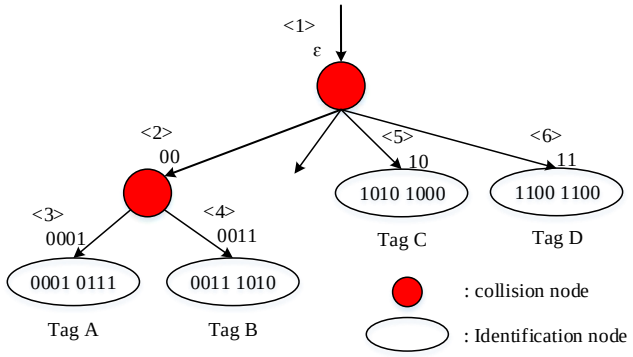


Figure 2. Example of M-ary query tree ($m = 2$).

According to the combined mapping part received from the tags, the reader can identify the locations of a '0' bit and a colliding bit. Such a '0' bit and a colliding bit indicate that the corresponding nodes are an empty node and a readable or collided node, respectively.

Table 2. Communications procedure by using MQT

Slot	Prefix	Mapping	Rest part	Identification
#1	ε	**0*	******	
#2	00	0*0*	**1*	
#3	0001	0010	11	0000 1011 (A)
#4	0011	0100	10	0010 0110 (B)
#5	10	0100	1000	1010 1000 (C)
#6	11	0001	1100	1100 1100 (C)

2.2 Window methodology

The QwT and CwT are the attractive methodologies applying a bit window to QT and CT, respectively. CwT adopts the bit-tracking, while QwT utilizes the CRC in order to judge the slot status. The reader broadcasts a query prefix [$q_1 \ldots q_L$] with the length L by attaching the window size (ws) with the length of $[\log_2 ws]+1$ bits. This bit string ws tells the tag how many bits they should respond, which is calculated in each query cycle. CwT has three possible slot statuses, as follows:

- When at least one collision bit is detected, a collision slot happens. Then, the reader generates two new queries $[q_1 \ldots q_L, w_1 \ldots w_{col-1}, 0]$ and $[q_1 \ldots q_L, w_1 \ldots w_{col-1}, 1]$ by using bit-tracking, where w_{col} indicates the first collision bit in the window part.
- When at least one tag responds, and the condition $L + ws < k$ is satisfied, a go-on slot happens. Then, the reader generates a new query by attaching the received window to the previous query prefix. The window size ws is recalculated using the heuristic function in Eq. (1).
- When the condition $L + ws = k$ is satisfied, a success slot happens. Then, the tag is identified subsequently.

Eq. (1) shows an exponential heuristic function, where β is an adjustable parameter, and it is selected through experiments [18, 19].

$$f(L) = k(1 - e^{-\beta L}) \quad (1)$$

The CwT protocol significantly reduces the number of bits transmitted by tags. As a result, it achieves a considerable energy saving. However, this benefit is accompanied with a certain degree of increase in the numbers of slots and bits transmitted by the reader because the use of bit window forces this protocol to introduce a new type of slot, go-on slot, which is the additional slot to obtain the last part of tag ID. In addition, the window size needs to be recalculated in each query cycle, and the reader's query message contains a bit string to specify the window size [3].

3. PROPOSED IMQCT PROTOCOL

3.1 System transmission model

We use the transmission model defined by GS1 EPC, which corresponds to the EPCglobal Class 1 Gen 2 Specification [21]. Figure 3 illustrates the link timing of the collision, go-on, and success slot. During the identification, time is divided into slots, and each slot begins with the reader's query commands. The reader transmits the continuous-wave (CW) RF signal and the query commands during the time t_R . The tags harvest operating energy from this RF signal and transmit their messages. After receiving the reader's command, the tags generate their responses during the time T_1 . The tag's response lasts during the time t_T . Additionally, T_2 is the time duration from the finish of tag response to reader transmission.

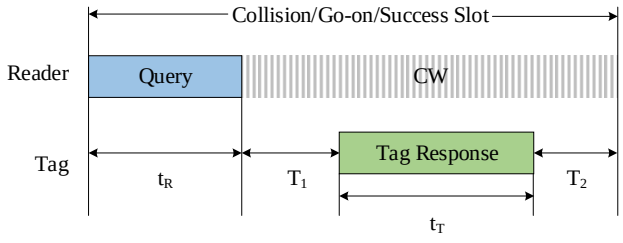


Figure 3. Linking time for collision, go-on and success slot.

According to the above transmission model, the time and energy model are described as follows. The time taken for identifying n tags in the reading area consists of the time required for transmitting the reader's query and tag's response in each slot. Let C_c, C_g and C_s denote the numbers of the collision, go-on and successful slots, respectively, and they can be counted in the reader side. Therefore, the identification time is written as follows:

$$T(n) = \sum_{i}^{C_c+C_g+C_s} (T_{Ri} + T_1 + T_{Ti} + T_2) \quad (2)$$

where T_{Ri} and T_{Ti} refer to the time taken for transmitting the reader's query and tags' response in the i-th slot, respectively. The energy consumption of the reader is expressed by (3) and calculated during the time of transmitting and receiving information. The reader will transmit the RF signal to power up passive tags with power P_{tx} in each interrogation cycle. The reader will consume extra power P_{rx} to receive a tag's response.

$$E(n) = \sum_{i}^{C_c+C_g+C_s} [P_{tx} \times (T_{Ri} + T_1 + T_{Ti} + T_2) + P_{rx} \times T_{Ti}] \quad (3)$$

3.2 Fixed-window methodology

In tree-based protocols, the collision probability of tag response is very high at the beginning of the identification process, and the transmission of the rest bits except prefix causes a long identification delay; moreover, the longer the tag ID is, the more the waste of the transmitted bits is. In MQT, both the M-bits mapping part and ID string are responded in each slot. The occurrence of a collision makes the M-bits string very useful but ID string wasted, and thus, the communication overhead is increased. In order to effectively limit the length of the tag response, the window methodology is applied to the MQT protocol with none-idle slot and m-bits arbitration feature.

The window-resizing methodology makes it possible to decrease the length of the tag response. In other words, the window needs to be resized as a smaller value in the trend of a high probability of collision and as a larger value in the trend of a low probability of collision. In general, the collision prediction is difficult, and its implementation in the reader and the tags are expensive. Therefore, the go-on slot, in which no collision occurs in the window part, is used for the collision prediction in the next query. As the length of query prefix increases during the identification process, the number of tags that belong to the node corresponding to the query prefix decreases, and the collision probability within the window part gradually decreases, finally leading to the go-on slot. Especially in MQT, since the collision tags are divided into many subgroups (M), the collision probability for the next query after the occurrence of the go-on slot is considerably reduced. Therefore, reducing the number of queries by transmitting the whole rest bit is more reasonable rather than increasing the window size gradually. That is, the window size is not changed until the go-on slot occurs, and once the go-on slot occurs, the rest bits are transmitted. It is called fixed-window methodology, differentiating it from the traditional window methodology in which varies the window size in each query cycle.

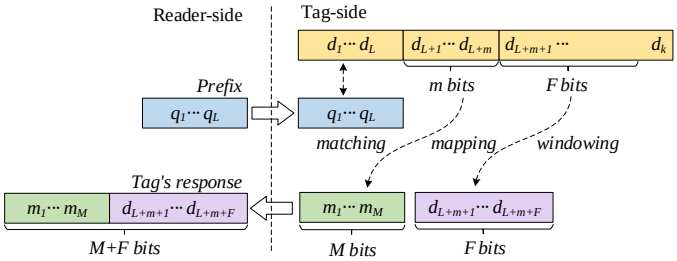


Figure 4. The fixed-window procedure on the collision slot.

The fixed-window size affects the performance of the protocol. If the fixed-window size is large, the number of bits transmitted by the tag increases, and the collision probability in the window part also increases. Thereby, the occurrence probability of go-on slot is reduced, resulting in an increased number of query-response cycles. In particular, if the window size is large enough, the number of bits transmitted by tags might be equal to the performance of the MQT. This fixed-window methodology can be simply implemented in the reader and the tags because the use of the fixed-window does not require the calculation of window size and transmission of the bit string representing it in each query cycle.

The fixed-window is defined as a bit string of the fixed length that the tag must transmit, and the length, F ($1 \le F \le k-L-m$), is a predefined parameter of the system, where k and L are the lengths of the tag ID and query prefix, respectively. Figure 4 shows the communication procedure for the fixed-window query on the collision slot. The introduction of the fixed-window query into MQT allows the tag to transmit only the mapping pattern and fixed-window part ($M+F$), not the rest of its ID ($k-L$ bits). In the paper, this query type is called *mapping-window query* (designated briefly as *mwq*). Fixed-window procedure changes the creation process of the collision and success slot. If two or more tags' windows do not include any different bit, a go-on slot situation ($L+m+F<k$) which would result in a success slot but may not be a success is produced. In this case, they need to be re-queried until a success slot condition ($L+m+F=k$) is guaranteed. When there is no collision in the window part, the reader uses the window part directly to generate new queries, like a common part of CT, and therefore the total query-response cycles can be reduced.

3.3 The proposed IMQCT protocol

IMQCT utilizes two types of the query; that is, *mapping-window query* (*mwq*) and *mapping-rest query* (*mrq*), and their messages contain a query prefix [$q_1 \ldots q_L$], indicating to the tags whether to respond or not. Algorithm 1 and 2 shows the pseudo-code of the IMQCT protocol. The

identification procedure is executed until all the tags in the interrogation area are identified. The query is initialized with type = *mwq* and prefix = 'ε'. The reader broadcasts a query message and waits for the response from the tags. After receiving the reader's query, the tag compares the prefix with its ID $[d_1 \ldots d_L]$ (Algorithm 2 line 3). After successful matching, the tag maps m bits into M bits by using the mapping table (Algorithm 2 line 4). According to the type of query received by the reader, the tag decides the length of the bit string that it should respond, as shown below (Algorithm 2 lines 5–9).

$$r = \begin{cases} F, & queryType = mwq \\ k - L - m, & queryType = mrq \end{cases} \quad (4)$$

In detail, if the type of query is *mwq*, the bit string to be transmitted is the window part $[d_{L+m+1} \ldots d_{L+m+F}]$. Otherwise, the bit string to be transmitted is the rest tag ID $[d_{L+m+1} \ldots d_k]$. Then, the tag transmits the mapping pattern $[m_1 \ldots m_M]$ and the prepared bit string to the reader. After receiving the tag response, the reader retrieves p sets of m-bits from the combined mapping part with the length M by using the mapping table (Algorithm 1 lines 7 and 8), where p is the number of the collided bits in the combined mapping part ($1 \le p \le 2^m$). When a collision occurs in the received bit string except for the mapping part, the reader creates p new queries $[q_1 \ldots q_L m_1 \ldots m_m]$ (Algorithm 1 lines 10-14). When the type of the current query is *mwq*, and no collision occurs in the window part, go-on status is satisfied. At this time, the reader creates p *mrq* queries by appending the m-bits string and window part to the previous query, that is, $[q_1 \ldots q_L m_1 \ldots m_m w_1 \ldots w_F]$ (Algorithm 1 lines 16-21). When the type of the current query is *mrq*, and collision bit is not detected in the received bit string except for the mapping part, the reader identifies p tag IDs (Algorithm 1 lines 16-21). Table 3 illustrates an identification process of IMQCT protocol. In this example, we assume that there are four tags (A, B, C and D) which have (000001100011), (001011101010), (110100011110) and (111100111111) as their IDs, where $m = 2$ and $F = 2$.

Algorithm 1 *Reader Operation*

```
(1)  Initialize: query.type = mwq; query.prefix = 'ε'
(2)  Push query into S
(3)  While S!=NULL do
(4)     query = pop(S)
(5)     broadcast(query)
(6)     [mapPart, respBits]=receiveResponse()
(7)     p=countCollBits(mapPart)
(8)     mBits=demapping(mapPart)
(9)     if isCollision(respBits) then
(10)       for i=1 to p
(11)          query.type=mwq
(12)          query.prefix+=mBits[i]
(13)          push(query)
(14)       end for
(15)    else
(16)       if query.type=mwq then
(17)          for i=1 to p
(18)             query.type=mrq
(19)             query.prefix+=mBits[i]+respBits
(20)             push(query)
(21)          end for
(22)       else if query.type=mrq
(23)          for i=1 to p
(24)             tagID=query.prefix+mBits[i]+ respBits
(25)          end for
(26)       end if
(27)    end if
(28) end while
```

Algorithm 2 *Tag Operation*

```
(1)  Receive a query
(2)  L=getPrefixLength(query)
(3)  if query.prefix=ID[1:L] then
(4)     mapPtn[1:M]=mapping(ID[L+1:L+m])
(5)     if query.type=mwq then
(6)        respBits=ID[L+m+1:L+m+F]
(7)     else if query.type=mrq then
(8)        respBits=ID[L+m+1:k]
(9)     end if
(10)    backscatter(mapPtn+respBits)
(11) end if
```

In Table 3, '*' denotes collision, and the bit string in parentheses indicates the mapping part received. When the reader broadcasts a query *mwq* (ε), all tags respond with their mapping patterns (0001, 0001, 1000 and 1000) and window parts (00, 10, 01 and 11), respectively. At this time, the mapping part and window part received by the reader is (*00*) and '**', respectively (#1 of Table 3). Using demapping, the reader gets two m-bits (00 and 11) by using the mapping table and then generates two query commands *mwq* (00) and *mwq* (11).

Table 3. Identification process by using IMQCT ($M = 2$ and $F = 2$)

Slot	Reader query	Received bits	Slot status
#1	*mwq* (ε)	(*00*) **	collision
#2	*mwq* (00)	(0*0*) *1	collision
#3	*mwq* (0000)	(0010) 10	go-on
#4	*mrq* (00000110)	(0001) 11	success (A)
#5	*mwq* (0010)	(1000) 10	go-on

#6	*mrq* (00101110)	(0100) 10	success (B)
#7	*mwq* (11)	(*0*0) 00	go-on
#8	*mrq* (110100)	(0010) 1110	success (C)
#9	*mrq* (111100)	(1000) 1111	success (D)

In #2 of Table 3, the reader sends a query mwq (00), where the tag A and B matches the prefix (00), then two tags respond with (0001) 01 and (0100) 11, respectively. Since the collision occurs in the window part, the reader gets two m-bits (00 and 10) from mapping part (0*0*) and then prepares two new queries mwq (0000) and mwq (0010). In #3, tag A matches the prefix (0000) and responds with (0010) 10. Since there is no collision bit in the window part, go-on slot status is satisfied. Then, the reader sends mrq (00000110), tag A finally is identified (#4). Other tags are also identified in the same way.

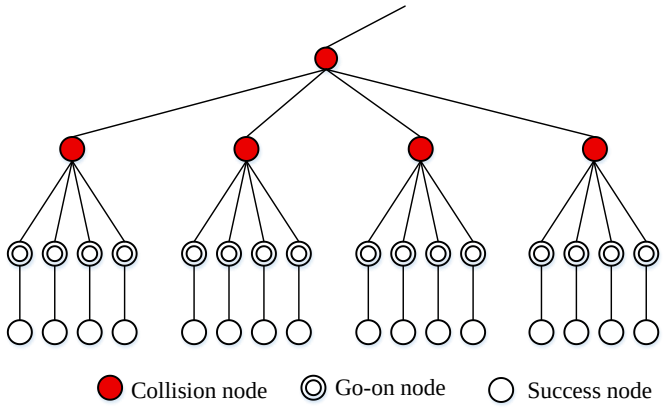


Figure 5. The particular type of the M-ary tree ($m=2$).

In order to simplify the theoretical analysis, we consider a particular type of M-ary tree similar to the perfect M-ary tree. In this tree, all internal nodes except the parent-of-leaf nodes have M children, and each parent-of-leaf node has only one leaf node, and all leaf nodes are in the same depth. The root node is in the depth of 0 and the number of the internal nodes at a depth of h is 2^{mh}. The parent-of-leaf nodes are in depth H ($H=1,2,3\ldots$). Therefore, the parent-of-leaf node corresponds to the go-on slot, while the leaf node corresponds to the success slot. If tag IDs are uniformly distributed, the tree constructed by the distribution may be similar to this particular tree in shape. The number of bits transmitted on the collision and go-on slot is:

$$b_w = M + F \tag{5}$$

The number of bits transmitted on the success slot can be written as follows:

$$b_r = k - m(H+1) - F - m + M \tag{6}$$

Thus, the total number of bits transmitted by tags for the above tree structure can be expressed as given in (7).

$$S = \sum_{h=0}^{H} b_w 2^{mh} + b_r 2^{mH} \tag{7}$$

The finite sum of exponential terms can be computed as

$$\sum_{i=0}^{n-1} a^i = \frac{a^n - 1}{a - 1} \tag{8}$$

Substituting Eq. (8) into Eq. (7), S_{MQfwT} can be written as

$$S_{IMQCT} = \sum_{h=0}^{H-1} b_w 2^{mh} + b_w 2^{mH} + b_r 2^{mH}$$

$$= b_w \frac{2^{mH} - 1}{2^m - 1} + b_w 2^{mH} + b_r 2^{mH} \tag{9}$$

The average tag transmitting bit for one tag identification in IMQCT is

$$A_{IMQCT} \approx \frac{b_w}{2^m - 1} + b_w + b_r$$

$$\approx k + 2M - (H+2)m + \frac{M+F}{2^m - 1} \tag{10}$$

From Eq. (10), we can see that the number of tag transmission bits decreases as the number of tags increases. In particular, the number of tag transmission bits is close to the tag length when $m=2$ and $F=1$.

4. RESULTS AND DISCUSSION

This section presents the simulation results of the proposed protocol using Visual Studio 2013 and Qt 5.5.0. A comparison between IMQCT protocol and tree-based protocols including CT, CwT and MQT is presented here. First, the influence of changing F in IMQCT is shown in the graph. Then all the protocols from state-of-the-art will be compared with IMQCT under different tag sets.

Table. 4 Parameters used in simulations

Parameter	Value
k	128 bits
Tari	6.25μs
Data rate	160kbps
T_1	18.86 μs
T_2	8.13 μs
P_{tx}	825mW
P_{rx}	125mW

It should be noticed that some overhead is not taken into account in our simulation due to the communication latency and the propagation delay from the signal processing on the channel. The protocols are compared in a scenario with one reader and a set of tags varying from 200

to 5000 tags. Tag IDs are assumed as uniformly distributed and are dynamically generated for every simulation iteration. The simulated results were averaged over 100 iterations for accuracy.

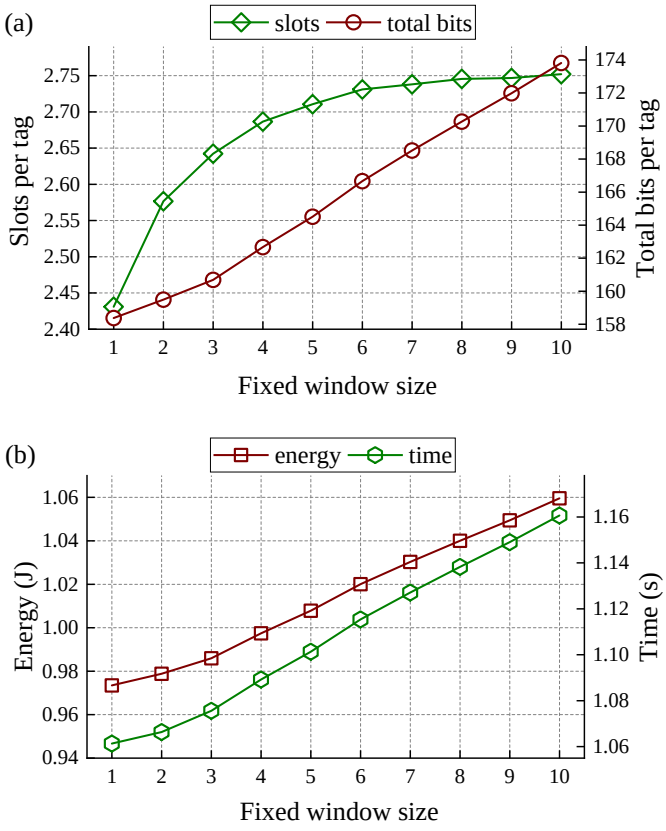
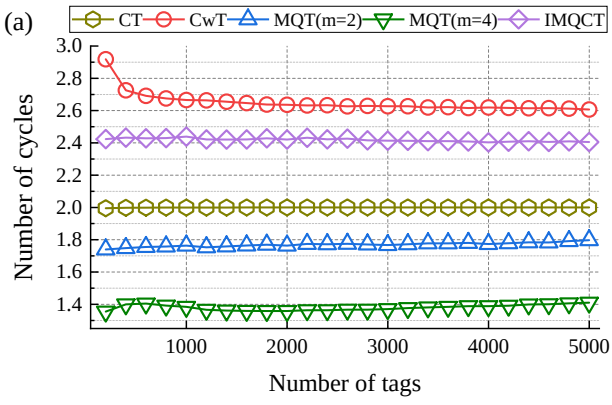


Figure 6. Simulation results per tag: (a) the number of slots and total transmitted bits and (b) consumed energy and time.

Table 4 shows the parameters used in the simulations. *Tari* is the duration of a data-0 ($Tari = 6.25\mu s$) and it influences the other parameters. To simplify the calculation, Bits 0 and 1 have been considered as 1 *Tari*. For CwT, the exponential function is used as a heuristic function and the value of β is set 0.419.



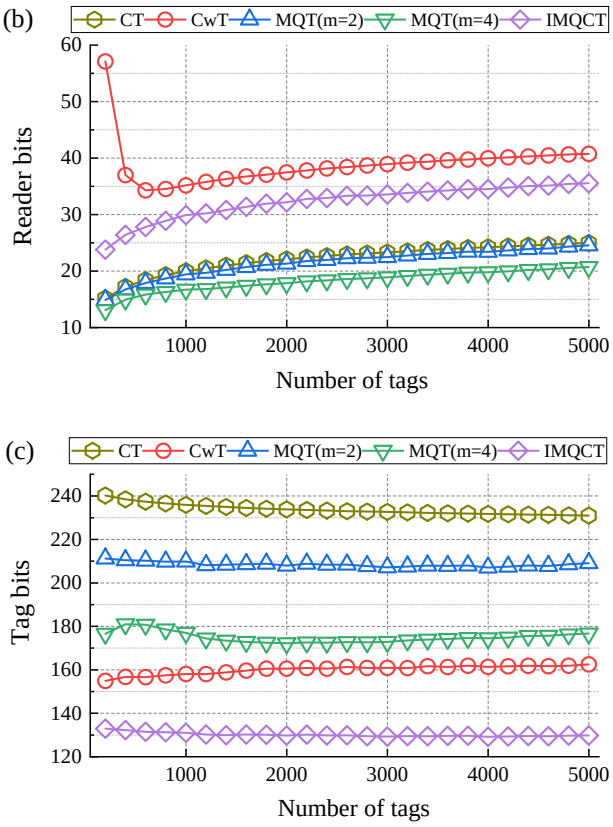


Figure 7. Simulation results per tag: (a) the number of slots, (b) the reader transmitting bits and (c) the tag transmitting bits.

In order to validate the fixed-window methodology, the fixed-window size needs to be adjusted since the number of query cycles and transmitted bits is influenced by the window size. This observation is carried out for varying fixed-window size F and a set of tags ($k = 128$ and $n = 1000$).

Figure 6 shows the obtained result: Figure 6 (a) shows the number of slots and total bits per tag, and Figure 6 (b) shows the energy and time consumed to identify one tag. As can be observed from Figure 6, as F increases, the value of all performance metrics increases. In addition, when $m = 4$, the number of slots is smaller than when $m = 2$. However, the number of total transmitted bits and the energy and time consumed are larger. From the observation above, we can see that the proposed protocol gives the best performance when $m = 2$ and $F = 1$, and thus these parameters are used in simulations.

Figure 7 (a) shows the average number of slots required to identify a given number of tags. From the figure, we can see that MQT ($m = 4$) consumes the least slots, and the window-based protocols (CwT and IMQCT) require more reader bits than other protocols because they introduce the go-on slot, which is the additional slot to obtain the last part of the tag ID. IMQCT consumes fewer slots than CwT, and it is because that IMQCT is based on MQT that can

arbitrate *m*-bits in each cycle. As can be seen in Figure 7 (c), IMQCT protocol outperforms all compared protocols in terms of the number of bits transmitted by tags. More specifically, as described in Eq. (10), when the number of tags increases, the average number of tag transmission bits decreases. Moreover, when $n = 1000$, the average number of bits transmitted by tag is 130.8, which is close to the tag ID length.

Figure 8 compares the bit efficiency of various protocols. The bits efficiency is calculated by the expression $(k / a_t) \times 100$, which means the usefulness of tags' responses and the influence of the collision and go-on slots (here, a_t is the average number of bits transmitted by the reader and tags). As can be observed from the figure, IMQCT achieves the highest bit efficiency, and it is about 77.5% when the tag number is 5000.

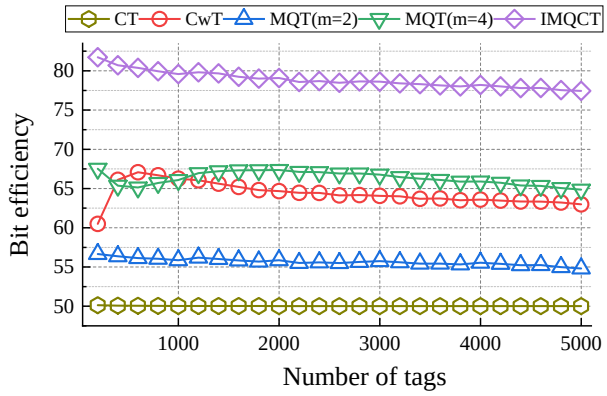


Figure 8. Bits efficiency.

Many anti-collision protocols have been evaluated using the number of slots and transmitted bits, but not the time and energy. The time and energy required to identify all tags in the interrogation area are calculated by using Eq. (2) and (3). Figure 9 shows the identification speed of IMQCT compared to other protocols.

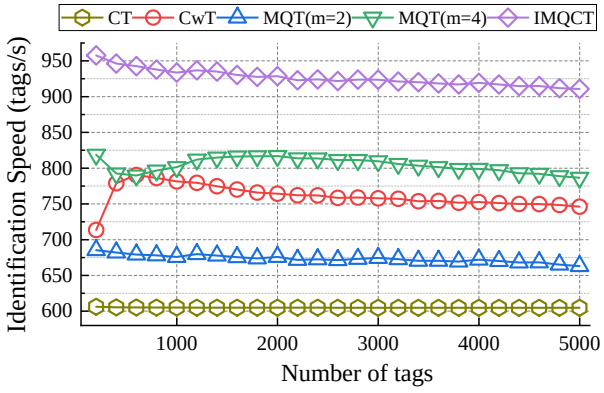


Figure 9. Identification speed.

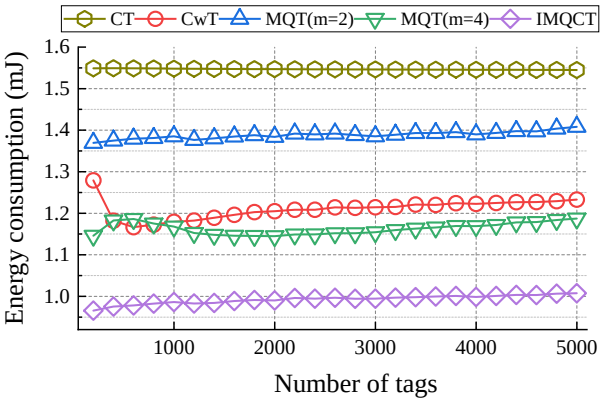


Figure 10. Energy consumption per tag.

Figure 10 shows the simulation result in terms of energy consumed by the reader. In this comparison, IMQCT outperforms the CwT and other tree-based protocols for all the different sets of tags. For example, the proposed protocol saved more energy than CT, CwT, MQT ($m = 2$) and MQT ($m = 4$) up to 34.8%, 18.3%, 28.5% and 15.2%, respectively when the number of tags is 5000. Therefore, the proposed protocol not only achieves a higher identification speed but also saves energy consumption under the same conditions.

5. CONCLUSION

In this paper, a new anti-collision protocol called IMQCT was proposed. IMQCT was designed by applying a fixed-window procedure to the MQT protocol in order to limit the number of bits transmitted by tags. The proposed protocol was compared with the existing several tree-based protocols as regard to relation to the number of slots, transmitted bits, consumed energy and identification speed. Simulation results showed that IMQCT outperformed them in terms of identification speed under low energy consumption. Therefore, IMQCT can be a suitable candidate where energy-efficiency is sought in the large-scale passive RFID system. There are still some issues to be resolved in our future work. Although the proposed protocol has improved the performance of MQT with the introduction of fixed-window, it still suffers from a large number of slots (go-on slot). Our future work will focus on the minimization of the number of slot numbers. In addition, the effect of various tag ID distributions needs to be investigated since IMQCT is based on the uniqueness of the tag IDs.

References

[1] Klair DK, Chin KW and Raad R. A survey and tutorial of RFID anti-collision protocols. IEEE Commun Surv Tut 2010; 12(3): 400-421.

[2] Zhu L and Yum TS. A critical survey and analysis of RFID anti-collision mechanisms. IEEE Commun Mag 2011; 49(5): 214–221.

[3] Cmiljanic N, Landaluce H and Perallos A. A comparison of RFID anti-collision protocols for tag identification. Appl Sci-Basel 2018; 8(8): 1282.

[4] Vogt H. Efficient object identification with passive RFID tags. In: Proceedings of the international conference on pervasive computing 2002; 98–113.

[5] Zhang L, Zhang J and Tang X. Assigned tree slotted aloha RFID tag anti-collision protocols. IEEE T Wirel Commun 2013; 12(11): 5493–5505.

[6] Wu H, Zeng Y, Feng J, et al. Binary tree slotted ALOHA for passive RFID tag anticollision. IEEE T Parall Distr 2013; 24(1): 19–31.

[7] Yang J, Wang YH, Cai QL, et al. A Novel Hybrid Anticollision Algorithm for RFID System Based on Grouped Dynamic Framed Recognition and Binary Tree Recursive Process. Int J Distrib Sens N 2015; 11(8): 641327.

[8] Namboodiri V, Gao L. Energy-aware tag anticollision protocols for RFID systems. IEEE T Mobile Comput 2010; 9(1): 44–59.

[9] Yan X, and Liu X. Evaluating the energy consumption of the RFID tag collision resolution protocols. Telecommun Syst 2013; 52(4): 2561–2568.

[10] Klair DK, Chin KW and Raad R. An investigation into the energy efficiency of pure and slotted aloha based RFID anti-collision protocols. In: Proceedings of 2007 IEEE international symposium on a world of wireless, mobile and multimedia networks 2007; 1–4.

[11] Rahimian S, Noori M and Ardakani M. An energy-efficient adaptive frameless ALOHA protocol. EURASIP J Wirel Comm 2016; 2016:186.

[12] C Law, K Lee, K Y Siu, Efficient memoryless protocol for tag identification, In: Proceedings of the 4th international workshop on discrete algorithms and methods for mobile computing and communications. 75–84 (2000)

[13] Zhang W, Guo YJ, Tang XM, et al. An Efficient Adaptive Anticollision Algorithm Based on 4-Ary Pruning Query Tree. Int J Distrib Sens N 2013; 9(12): 848746.

[14] Jia X, Feng Q and Ma C. An efficient anti-collision protocol for RFID tag identification. IEEE Commun Lett 2010; 14(11): 1014-1016.

[15] Chen Y, Yeh K, Lo N, et al. Adaptive collision resolution for efficient RFID tag identification. EURASIP J Wirel Comm 2011; 2011:139.

[16] Shin J, Jeon B and Yang D. Multiple RFID tags identification with M-ary query tree scheme. IEEE Commun Lett 2013; 17(3): 604-607.

[17] M A Kalache, L Fergani, Performances comparison of RFID anti-collision algorithms. In: Proceedings of international conference on multimedia computing and systems, 808-813 (2014)

[18] Landaluce H, Perallos A and Angulo I. Managing the number of tag bits transmitted in a bit-tracking RFID collision resolution protocol. Sensors 2014; 14: 1010–1027.

[19] Landaluce H, Perallos A, Onieva E, et al. An energy and identification time decreasing procedure for memoryless RFID tag anticollision protocols. IEEE T Wirel Commun 2016: 15(6); 4234–4247.

[20] Cmiljanic N, Landaluce H, Perallos A, et al. Influence of the distribution of tag IDs on RFID memoryless anti-collision protocols. Sensors 2017; 17(8): 2017:1891.

[21] GS1, EPC™ Radio-Frequency Identity Protocols Generation-2 UHF RFID Standard Specification for RFID Air Interface Protocol for Communications at 860 MHz–960Hz Ver. 2.1 (2018)