International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 04, April-2019

An Enhanced Collision Arbitration Algorithm Utilizing Fixed-Window Scheme in RFID System

Sok Kyongjin
Institute of Information Technology
University of Sciences
Pyongyang, Democratic People's Republic of Korea

Pak Unjin
Department of Automation engineering
Kim Chaek University of Technology
Pyongyang, Democratic People's Republic of Korea

Kim Kwanghun
Department of Engineering Machine
Pyongyang University of Mechanical Engineering
Pyongyang, Democratic People's Republic of Korea

Ryu Unsok
School of Information Science
Kim Il Sung University
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 particular, tag collision in a passive RFID system is a serious problem causing performance degradation. Most anti-collision algorithms waste numerous transmitted bits while eliminating or reducing the idle and collision slots. In order to save the reader's energy, it is inevitable to reduce both the number of query-response cycles and the transmitted bits as much as possible. In this study, an enhanced M-ary query tree (EMQT) algorithm was proposed. In order to restrict the length of bits transmitted by tags, a fixed-window scheme is applied to M-ary query tree algorithm which can perform multi-bit arbitration in each cycle without calculating and transmitting the window size in each query cycle. As a result of the simulation, the proposed algorithm is proved to be a considerably improved algorithm in energy and time savings, compared to the existing several tree-based algorithms.**

Keywords— RFID; anti-collision; tree-based; fixed-window

I. 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]. 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]. Therefore, the development of an efficient anti-collision algorithm is very critical and significant in the performance improvement of a passive RFID system.

Anti-collision algorithms can be classified into three broad categories: ALOHA-based, tree-based and hybrid algorithms. Although ALOHA-based algorithms have such advantages as simplicity and good performance, they have some disadvantages such as tag starvation problem in which tags cannot be identified for a long time, and they also suffer a considerable degradation of performance in large-scale systems. [1, 2, 14]. Tree-based algorithms make it possible to successfully identify all the tags in the interrogation area even when the number of tags is enormous [1-3, 12-21]. Finally, hybrid algorithms combine the advantages of ALOHA-based and tree-based algorithms. Hybrid algorithms 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 algorithms, such as collision tree (CT), k-Ary tree-based anti-collision scheme (k-TAS), M-ary query tree (MQT), dual prefix probe scheme (DPPS) and collision window tree (CwT) protocols. In general, the traditional tree-based algorithms such as query tree (QT) [12] and CT [14] generate two queries which differ only in the last bit and then transmit them one by one. Considering this feature, in DPPS, one query containing two prefixes which differ in the last bit is generated depending on the continuity of the first and second collided bits and transmitted, and two time-slots can be finally combined into one slot. As a result, the number of queries is considerably reduced, but a half idle slot occurs [15]. In k-TAS [16] and MQT [17], 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 algorithms. As a result of the comparison between several ALOHA-based and tree-based algorithms, k-TAS shows the best performance regarding the interrogation cycle and the time required for the identification of all tags [18].

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. In general, the reader's energy in passive RFID system consists of the energy to power up all the tags and the energy for the information exchange between reader and tags [1-3, 20]. In order to save the reader's energy, it is inevitable to reduce both the number of query-response cycles and the transmitted bits as much as possible. However, most of the previous tree-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 04, April-2019

based algorithms focus on the elimination or reduction of idle and collision slots. In recent years, the window-based protocols including query window tree (QwT) and collision window tree (CwT) are proved to be capable of limiting the length of tag response by using a bit-window methodology. Such an advantage of QwT and CwT is also accompanied by the increase of the number of query cycles due to the exploitation of a new slot type, go-on slot [3, 19-21].

In this study, we propose an enhanced M-ary query tree (EMQT) algorithm by applying a fixed-window scheme to MQT, focusing on a tree-based algorithm under the static environment in which the mobility of tags is not considered. The fixed-window scheme can effectively limit the length of the tag response, without calculating and transmitting the window size in each query cycle. In addition, the number of query cycles can be reduced than traditional window methodologies because the proposed algorithm is based on MQT which can perform the multi-bits arbitration. As a result of the comparison with the several existing tree-based algorithms, the proposed algorithm shows better performance mainly in terms of energy and time savings. The content of this paper is arranged as follows. Section 2 describes the MQT algorithm and bit window scheme. The EMQT algorithm is presented in Section 3. Section 4 shows the simulation results compared with the several existing tree-based algorithms. Finally, the conclusions are drawn in Section 5.

II. RELATED WORK

A. Bit Tracking and MQT Algorithm

At first, the bit-tracking technology is briefly explained for a better understanding of the proposed algorithm. Manchester coding has got very useful in the numerous algorithms for RFID tag identification because it can accelerate the identification process due to its capability of detecting the locations of collided bits [14-17]. In addition, this coding method is specified in the ISO/IEC 14443 standard. 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 Fig. 1; as a result, the location of collision bits can be detected. In the traditional tree-based algorithms including QT, CT and their variations, the combined response of tags is processed bit by bit.

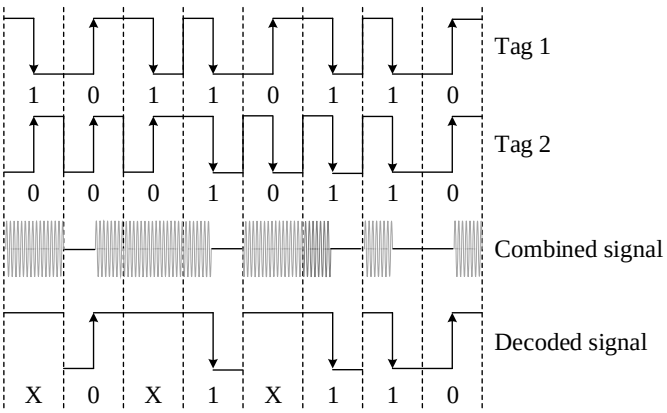

Fig. 1. Example of Manchester encoding (“X” denotes a collision bit).

TABLE I. AN EXAMPLE OF THE MAPPING TABLE

2 bits in Tag ID	4 bits mapped string
00	000**1**
01	00**1**0
10	0**1**00
11	**1**000

If a collision is found in a bit, the reader splits the bit into 0 and 1 and re-interrogates with the two new queries. The overall identification process can be explained as a binary tree. MQT can perform m-bits arbitration at a time since it is based on the mapping function and Manchester coding, and the identification process can be expressed through M-ary tree ($M = 2^m$). The mapping table for 2 bits is shown in Table I. After receiving a reader’s query, the tag compares the query prefix with its ID. If the comparison is true, the tag responds to the reader with its full ID except the same bit string as the prefix. At this time, the first m-bits of the response will be mapped into M-bits. In this paper, this type of query is called *full-response query*, designated simply as FRQ. M-bits string points out which child node of the M-ary tree the tag belongs to. According to the mapping part, which is combined M-bits string received by the reader, 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. Therefore, the reader generates new p ($1 \le p \le 2^m$) sets of queries $\{qr_1, ..., qr_p\}$, where q is the received prefix, p is the number of collision bits in the mapping part, and p sets of m-bits string $\{r_1, ..., r_p\}$ can be inversely mapped by using the mapping table or mapping function.

An example of the 4-ary tree formed in the identification process of the MQT is illustrated in Fig. 2, and the communication procedure is shown in Table II. In the table, ‘*’ denotes collision, and the bit string in parentheses indicates the combined M-bits string received in each time slot. In Fig. 2, ‘10’ is an empty node. After receiving the query prefix ‘00’, Tag A responds with (0**1**00) 1000, and then the reader identifies Tag A because no collision occurs (#2 of Table II). The reader sends a query ‘01’. Tag B and C respond with (00**1**0) 0111 and (0**1**00) 1010, respectively, which produce (0**0) **1* (#3 of Table II). By using the previous prefix ‘01’ and the mapping part (0**0), the reader generates two new queries ‘0101’ and ‘0110’ ($p = 2$, $r_1 = 01$ and $r_2 = 10$). After sending the queries (0101) and (0110) one by one, the reader identifies tags B and C, respectively (#4 and 5 of Table II).

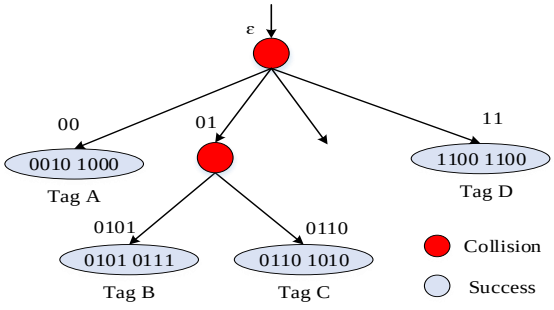

Fig. 2. Example of M-ary query tree ($m = 2$).

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 04, April-2019

TABLE II. COMMUNICATIONS PROCEDURE BY USING MQT.

Slot	Prefix	Received bits	Slot state
#1	ε	(*0**) ******	Collision
#2	00	(0**1**00) 1000	Success (A: 00.10. 1000)
#3	01	(0**0) **1*	Collision
#4	0101	(00**1**0) 11	Success (B: 0101. 01.11)
#5	0110	(0**1**00) 10	Success (C: 0110. 10.10)
#6	11	(0001) 1100	Success (D: 11.00. 1100)

Finally, the reader identifies tag D by sending the query '11' (#6 of Table II). 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.

B. *Window Methodology*

The QwT and CwT is the attractive methodology applying a bit window to QT and CT, respectively. CwT adopts the bit-tracking, while QwT utilizes the CRC in order to judge the status of the slot. 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, calculated in each query cycle. CwT has three possible slot statuses, as follows:

- If 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.
- If no collision occurs as well as 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 prefix. The window size ws is recalculated using the heuristic function in Equation (1).
- If no collision occurs as well as 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].

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

The CwT protocol significantly reduces the number of bits transmitted by tags.

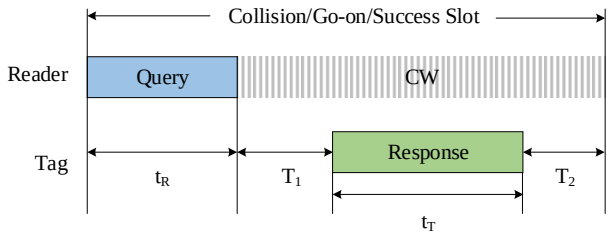

Fig. 3. Linking time for collision, go-on and success slot.

However, this benefit is accompanied by the increase of 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 remaining part of tag ID. In addition, the window size needs to be recalculated in each query cycle, and the reader's query message contains the extra bit string to specify the window size [3].

III. PROPOSED EMQT ALGORITHM

A. *System Transmission Model*

Transmission model is defined according to the EPC global Class 1 Gen 2 Specification. [22]. Fig. 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. T_1 is the time duration from the finish of reader transmission to tag response, and T_2 is the time duration from the finish of tag response to the reader transmission.

According to the above transmission model, the time and energy model are described as follows. The time taken for identifying n tags in the interrogation zone 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. The time to identify all tags can be written as follows.

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

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 Eq. (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.

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

B. *Fixed Window Scheme*

In the tree-based algorithms, the collision probability of tag response is very high at the beginning of the identification process. Therefore, the transmission of the full ID except the same bit string as the received 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 transmitted 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 fixed-window scheme is applied to the MQT protocol with none-idle slot and m -bits arbitration feature, and it is named as EMQT.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 04, April-2019

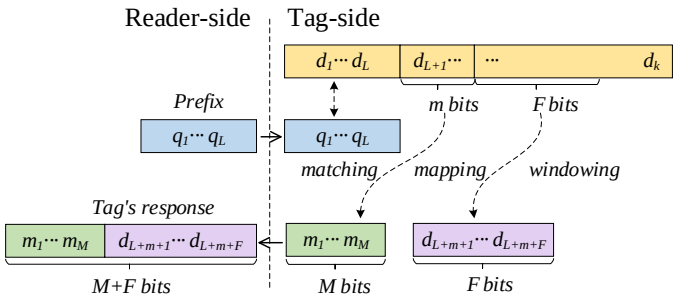

Fig. 4. The fixed-window scheme on the collision slot.

The main feature of the fixed-window scheme is to keep the window size as a constant F (predefined in the system) before the occurrence of a go-on slot. If a go-on slot occurs, it just transmits the full tag ID. It is differentiated from the existing window schemes in which the window size is varied in each query cycle. As the length of query prefix increases, the number of the tags matched to the prefix also decreases, eventually resulting in the go-on slot that no collision is detected in the window part. 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, transmitting the whole remaining ID at a time is more efficient rather than obtaining the remaining ID by expanding the window size for several times. This query type is called *window-response query* (designated briefly as WRQ). The size of the fixed-window affects the performance of the algorithm. If the fixed-window size is large, the number of bits transmitted by the tag increases. In addition, the collision probability in the window part also increases, thereby reducing the occurrence probability of a go-on slot and resulting in an increased number of query-response cycles.

Algorithm 1. *Reader Operation*

```
(1)  Initialize: query.type = WRQ; query.prefix = 'ε'
(2)  Push initial 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[1:p]=demapping(mapPart)
(9)      if isCollision(respBits)        // collision slot
(10)         for i=1 to p
(11)             query.type=WRQ
(12)             query.prefix+=mBits[i]
(13)             push(query)
(14)         end for
(15)     else
(16)         if query.type=WRQ           // go-on slot
(17)             for i=1 to p
(18)                 query.type=FRQ
(19)                 query.prefix+=mBits[i]+respBits
(20)                 push(query)
(21)             end for
(22)         else if query.type=FRQ      // success slot
(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]
(4)      mapPart[1:M]=mapping(ID[L+1:L+m])
(5)      if query.type=WRQ
(6)          respBits=ID[L+m+1:L+m+F]
(7)      else if query.type=FRQ
(8)          respBits=ID[L+m+1:k]
(9)      end if
(10)     backscatter(mapPart+respBits)
(11) end if
```

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. The fixed-window scheme can be easily implemented in the tag and reader because the fixed-window make it possible to avoid any complexity such as the calculation and transmission of the window size.

The fixed-window is defined as a bit string of the constant length that the tag should respond, 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. Fig. 4 shows the communication procedure for the window-response query. The introduction of the window-response query into MQT allows the tag to transmit the mapping pattern and fixed-window part ($M+F$ bits), not the remaining bits of tag ID ($k-L$ bits). Fixed-window scheme changes the creation process of the collision and success slot. If no collision occurs in the combined window bit string as well as the condition $L+m+F<k$ is satisfied, a go-on slot situation, 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 window part is directly appended to the prefix in the same way as in CT, and therefore the total query-response cycles can be reduced.

C. Proposed Algorithm

EMQT utilizes two types of the query: window-response query (WRQ) and full-response query (FRQ), and their messages contain a query prefix $[q_1 \ldots q_L]$, indicating to the tags whether to respond or not. The pseudo-codes of the reader and tag are shown in Algorithms 1 and 2, respectively. The identification procedure is executed until all the tags in the reading area are identified. The query is initialized with type = WRQ 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). If the comparison is true, the tag maps m bits into M bits by using the mapping table (Algorithm 2 line 4). According to the type of received query, the tag decides the length of the response r as follows (Algorithm 2 lines 5–9).

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

In detail, if the type of query is WRQ, the bit string to be transmitted is the window part $[d_{L+m+1} \ldots d_{L+m+F}]$. Otherwise,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 04, April-2019

the bit string to be transmitted is the remaining tag ID $[d_{L+m+1}\ldots d_k]$. Then, the tag transmits the mapping part $[m_1\ldots m_M]$ 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 (M bits) by using the mapping table (lines 7 and 8 in Algorithm 1), 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 WRQ queries by appending the m-bits string to the prefix ($[q_1\ldots q_L m_1\ldots m_m]$) (lines 10-14 in Algorithm 1). When the type of the current query is WRQ as well as no collision occurs in the window part, go-on status is satisfied. At this time, the reader creates p new FRQ queries by appending the m-bits string and window part to the previous query ($[q_1\ldots q_L m_1\ldots m_m w_1\ldots w_F]$) (lines 16-21 in Algorithm 1). If collision bit is not detected in the received bit string except for the mapping part as well as the query type is FRQ, success slot happens, and the reader identifies p tag IDs (lines 16-21 in Algorithm 1). Table III illustrates an identification process of EMQT algorithm. In this example, we assume that there are four tags (A, B, C and D), the IDs of which are (000001100011), (001011101010), (110100011110) and (111100111111), respectively, where $m = 2$ and $F = 2$. In Table III, '*' denotes collision, and the bit string in parentheses indicates the mapping part received by the reader. When the reader broadcasts a query WRQ (ε), all tags respond with their mapping parts (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. Because the collisions are detected in the window part, the reader retrieves two m-bits (00 and 11) by using mapping table and then generates two query commands WRQ (00) and WRQ (11). In #2 of Table III, the reader sends a query WRQ (00); as a result, Tag A and B respond with (0001) 01 and (0100) 11, respectively. Since a collision occurs in the window part, the reader retrieves two m-bits (00 and 10) from the mapping part (0*0*) and then prepares two new queries WRQ (0000) and WRQ (0010). In #3, Tag A transmits (0010) 10 after receiving the query WRQ (0000). Since there is no collision bit in the window part, go-on slot status is satisfied. Then, the reader sends FRQ (00000110), Tag A finally is identified. Other tags are also identified in the same way.

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.

TABLE III. IDENTIFICATION PROCESS BY USING EMQT

Slot	Reader query	Received bits	Slot status
#1	WRQ (ε)	(*00*) **	collision
#2	WRQ (00)	(0*0*) *1	collision
#3	WRQ (0000)	(0010) 10	go-on
#4	FRQ (00000110)	(0001) 11	success (A)
#5	WRQ (0010)	(1000) 10	go-on
#6	FRQ (00101110)	(0100) 10	success (B)
#7	WRQ (11)	(*0*0) 00	go-on
#8	FRQ (110100)	(0010) 1110	success (C)
#9	FRQ (111100)	(1000) 1111	success (D)

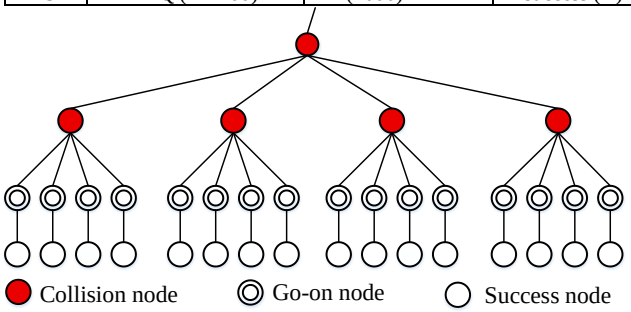

Fig. 5. The particular type of the M-ary tree ($m = 2$).

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:

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

Thus, the total number of bits transmitted by the tag can be expressed as given in Eq. (7).

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

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), the total number of bits transmitted by the tags during the identification, S_{IMQCT}, can be written as

$$\begin{aligned} S_{EMQT} &= \sum_{h=0}^{H-1} b_w 2^{mh} + b_w 2^{mH} + b_f 2^{mH} \\ &= b_w \frac{2^{mH} - 1}{2^m - 1} + b_w 2^{mH} + b_f 2^{mH} \end{aligned} \tag{9}$$

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

$$\begin{aligned} A_{EMQT} &\approx b_f + b_w + \frac{b_w}{2^m - 1} \\ &\approx k + 2M + F - (H+2)m + \frac{M+F}{2^m - 1} \end{aligned} \tag{10}$$

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 04, April-2019

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

TABLE IV. PARAMETERS USED IN SIMULATIONS.

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

IV. RESULTS AND DISCUSSION

This section presents the simulation results of the proposed algorithm using Visual Studio 2013 and Qt 5.5.0. A comparison between EMQT algorithm and several tree-based algorithms including CT, QwT, CwT, MQT and DPPS is presented here. The algorithms are compared in a scenario with one reader and a set of tags varying from 200 to 2000 tags. Tag IDs are unique for each tag, uniformly distributed and have the length of 128 bits. The simulation is performed 100 times for accuracy. Also, it should be noted that some overheads are not considered in this simulation due to the communication latency and the propagation delay from the signal processing on the channel. Table IV shows the parameters used in the simulations. Tari is the duration of a bit '0', which is set as 6.25µs and it influences the other parameters.

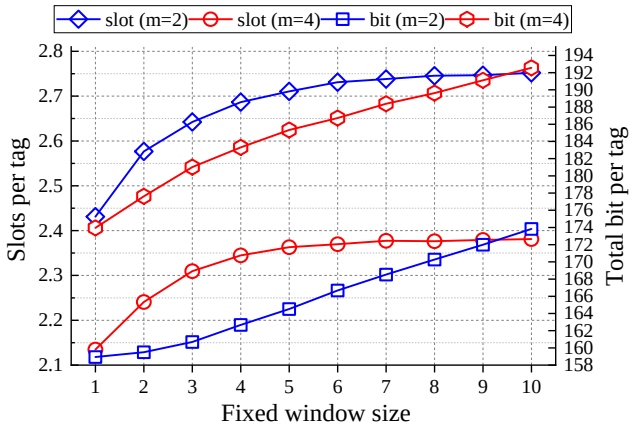

Fig. 6. The number of slots and tag transmitting bits

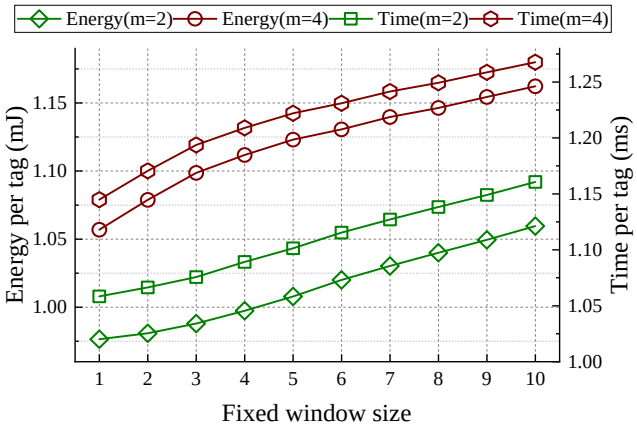

Fig. 7. Energy consumption and identification time.

To simplify the calculation, bits 0 and 1 have been considered as 1 Tari. In CwT, a quadratic function is used as a heuristic function, and the adjustable parameter β is set as 125 [6].

In order to validate the proposed algorithm, the parameters, m and F needs to be adjusted since they influence the number of query cycles and the bits transmitted by tag. This observation is carried out for the different m and F, a set of tags ($k = 128$ and $n = 1000$). Fig. 6 shows the number of slots and the average number of bits transmitted by the reader and tag. Fig. 7 shows the energy and time consumed to identify one tag. As can be observed from Figs. 6 and 7, as F increases the value of all performance metrics increases. In addition, when $m = 4$, the number of slots is smaller than when $m = 2$ while the number of total transmitted bits, energy cost and identification time are larger. From the observation above, we can see that the proposed algorithm gives the best performance for $m = 2$ and $F = 1$, thus these parameters are used in later simulations.

Figs. 8 and 9 show the average slots and the number of bits transmitted by the reader to identify a given number of tags, respectively. From the figures, we can see that DPPS consumes the least slots. Meanwhile, the window-based protocols (QwT, CwT and EMQT) require more query cycles and reader bits than other algorithms because they introduce the go-on slot that is the additional slot to obtain the last part of the tag ID. In addition, EMQT consumes fewer slots and reader bits than QwT and CwT does. It is because that EMQT applies the window scheme to MQT which can perform the multi-bits arbitration at a time.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 04, April-2019

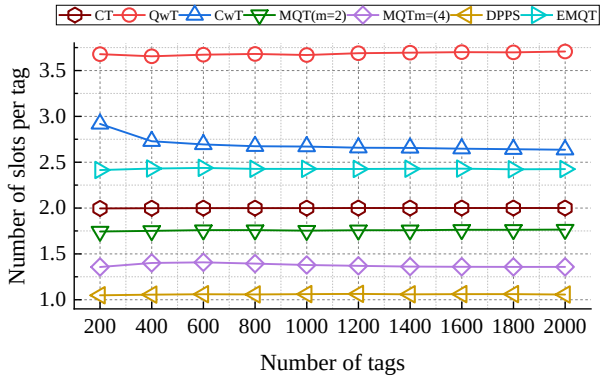

Fig. 8. Average number of slots per tag.

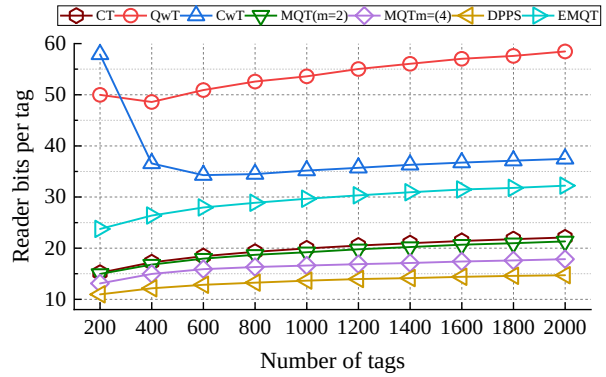

Fig. 9. Average number of bits transmitted by the reader.

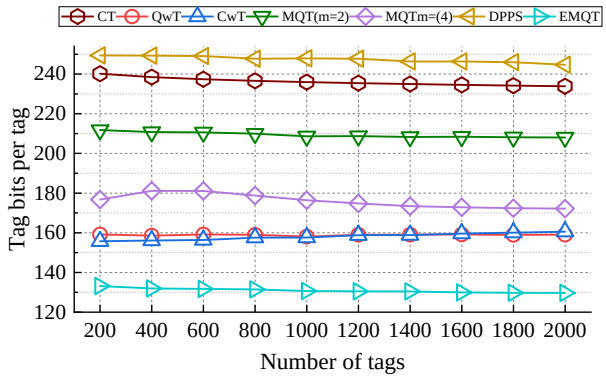

Fig. 10. Average number of bits transmitted by tag.

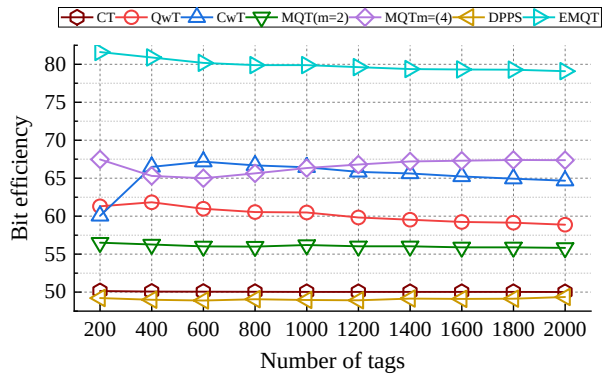

Fig. 11. Bits efficiency.

As can be seen in Fig. 10, EMQT algorithm outperforms the other compared algorithms as regards to 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 $n = 2000$, the average number of bits transmitted by tag is 129.6, which is close to the length of tag ID. The bit efficiency for the different set of tags, shown in Fig. 11, is defined as the percentage ratio of the tag ID length to the number of total transmitted bits per tag. The bit efficiency means the usefulness of tag responses and the influence of the collision and go-on slots. As can be observed from the figure, EMQT achieves the highest bit efficiency for all the set of tags, and it is about 79.1% when the tag number is 2000.

The time and energy required to identify all tags in the interrogation area are calculated by using Eqs. 2 and 3. Figs. 12 and 13 show the identification time and energy cost, respectively. EMQT outperforms all the compared algorithms for all the different sets of tags. For example, when the number of tags is 2000, EMQT consumes 36.1%, 39.4%, 17.8%, 28.5%, 13.5% and 38.7% less energy than CT, QwT, CwT, MQT $(m = 2)$, MQT $(m = 4)$ and DPPS, respectively. Therefore, the proposed algorithm not only achieves a higher identification speed but also saves energy consumption under the same conditions. The window-based algorithms require far fewer tag bits than the other compared algorithms do (Fig. 10). However, QwT shows the worst trend in the aspect of energy cost (Fig. 13) because of requiring a large number of query cycles (Fig. 8). Similarly, DPPS requires the smallest number of query cycles but shows the worst performance in terms of identification time and energy cost since it transmits the numerous numbers of tag bits.

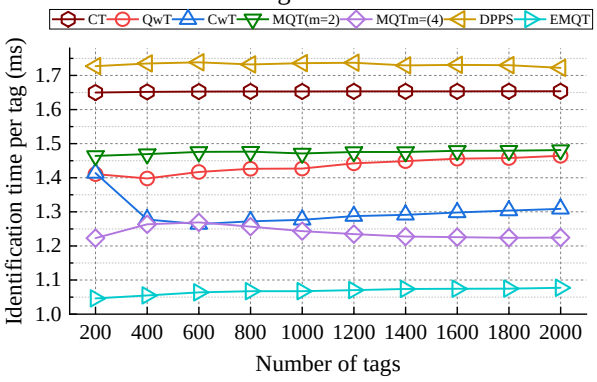

Fig. 12. Identification time per tag.

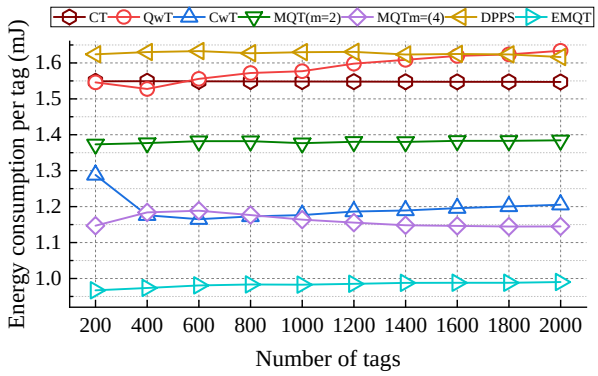

Fig. 13. Energy consumption per tag.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 04, April-2019

Therefore, it can be said that reducing the number of tag transmission bits is also important and significant in the saving of the reader's energy.

There are still some issues to be resolved in our future work. Although the proposed algorithm 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.

V. CONCLUSION

In this paper, an efficient anti-collision algorithm called EMQT was proposed. EMQT was designed by applying a fixed-window procedure to the MQT algorithm in order to limit the number of bits transmitted by tags. The proposed algorithm was compared with the existing several tree-based algorithms as regard to relation to the number of slots, transmitted bits, consumed energy and identification speed. Simulation results showed that EMQT outperformed them in terms of identification speed under low energy consumption. Therefore, EMQT can be a suitable candidate where energy-efficiency is sought in the large-scale passive RFID system.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their helpful comments, which were used to improve the quality of the paper. This work was supported by the National Natural Science Foundation of National Academy of Science (No.2016/231/1318) and Science & Technology Development Fund of the University of Sciences (100/425-63053-248/1).

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, Gu. Y, "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, Zhan. Y, "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, 2000, 75–84.
[13] Zhang. W, Guo. YJ, Tang. XM, Cui. GH, "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] J. Su, Z. Sheng, G. Wen, and V. C. M. Leung, "A time efficient tag identification algorithm using dual prefix probe scheme (DPPS)," IEEE Signal Process. Lett, 2016, vol. 23, no. 3, 386–389.
[16] Chen. Y, Yeh. K, Lo. N, Li. Y and Winata. E, "Adaptive collision resolution for efficient RFID tag identification," EURASIP J Wirel Comm, 2011, 2011:139.
[17] 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.
[18] M. A. Kalache and L. Fergani, "Performances comparison of RFID anti-collision algorithms," In: Proceedings of international conference on multimedia computing and systems, 2014, 808-813.
[19] 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.
[20] Landaluce. H, Perallos. A, Onieva. E, Arjona. L and Bengtsson. L, "An energy and identification time decreasing procedure for memoryless RFID tag anticollision protocols," IEEE T Wirel Commun, 2016, 15(6), 4234–4247.
[21] Cmiljanic. N, Landaluce. H, Perallos. A and Arjona, L, "Influence of the distribution of tag IDs on RFID memoryless anti-collision protocols," Sensors, 2017, 17(8), 2017:1891.
[22] 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)