Hasten Dynamic Frame Slotted ALOHA Algorithm for Fast Identification in RFID System

Siti M. Wasikon and Mustafa M. Deris

Faculty of Information Technology and Multimedia,
Universiti Tun Hussein Onn Malaysia
{mahfuzoh,mmustafa}@uthm.edu.my

Abstract. The problems of identifying a set of tagged objects simultaneously in an RFID network have hampered the adoption of RFID universally. When multiple tags transmit their IDs in a single period of time to a single reader over a shared wireless channel, the tag signals may collide. This collision will disturb the reader's identification process. Hence, we proposed a modified version of Accelerated Frame Slotted ALOHA protocol called Hasten Dynamic Frame Slotted ALOHA (HDFSA) to condense a number of retransmission for one tag, thereby reducing the processing time. In HDFSA, the unread tag will be provided with a new slot in the subsequent reading cycle. The simulation conducted revealed that HDFSA outperformed the present Accelerated Frame Slotted ALOHA up to 78 percent in terms of processing time with less complexity, while preserving the accuracy of tag identification.

Keywords: RFID, Tag Anti-Collision, Unread Tag.

1 Introduction

In the early stage of RFID existence, various study have shown that RFID system provides an efficient and inexpensive mechanism for automatically collecting the identity information of an object [1]. This automatic identification (autoID) device operates by transmitting a signal (radio wave) from the reader to the tags. Every command that broadcast by a reader will be processed by all the tags within the range of reader. The reader then will recognize the objects through the ID that given by tags.

A simultaneous data transmission by several tags to a single reader will lead to mutual interference, and therefore to data loss which also known as tag collision [2] occurred. Tag collision makes RFID loss their usefulness as a quick, flexible, and reliable method to electronically detect [3] a variety of objects in one time. This is especially critical for passive tag due to its limitation to detect collisions or to figure out neighboring tags. Therefore, the need for anti-collision protocol is necessary to enable the recognition of multiple tags in a single time.

In general, there are two types of familiar tag anti-collision for passive tags, namely ALOHA based and tree based protocols. Tree-based protocols such as the binary tree protocol [4], [5] and the query tree protocol [6], [7], [8] operate by traversing the node sequentially. Every tag that transmits their ID in the same time will form a set. A splitting mechanism is applied to the colliding set where it is then will be partitioned into two subsets recursively in turn to be recognized. Thus, the iteration process will relatively cause an identification delay after the splitting procedure from one set until all the tags is being identified. Furthermore, in the worst case of some tree based protocol, the size of tree that the protocol has to traverse might be as equal as the number of bit of serial number inside the tags.

On the other hand, ALOHA based algorithm such as ALOHA [9], slotted ALOHA [10], frame slotted ALOHA [11], [12] and dynamic frame slotted ALOHA [13], [14], [15], [16], [17] is a simple procedure with low complexity. ALOHA based protocol was introduced in order to cut down the number of probability of collision by providing the time slot. The time slot given will allow the tags to transmit their ID in their preferable time which is distinct from each other. Thus, the occurrence of collision will reduce and this offer low complexity and computation [18].

One of dynamic frame slotted ALOHA family, Accelerated Frame Slotted ALOHA (AFSA) algorithm which is proposed in [16], had came out with a good idea of solution to counteract the probability of undetected collision in [14]. AFSA, which based on Enhanced Dynamic Frame Slotted ALOHA (EDFSA) [14], revealed that by reducing the length of bit sequence used in the reservation phase, the average tag reading time had outperformed [14] and [17] more than 40%. However, 'tag starvation problem' as happen to other ALOHA based protocol [19], might caused the unread tags in the first reading cycle not being identified for a long time in the next cycle of reading process.

Hence, we proposed a modified version of AFSA, called Hasten Dynamic Frame Slotted ALOHA (HDFSA) which restrains the unread tag in the first reading cycle from being collided in the next reading cycle. We focus our attention on scheduling of the unread tags in the subsequent reading cycle. By reducing collisions in the subsequent reading cycle, identification process of tags will be faster and tags can be recognized with only a few transmissions of ID. The essential element of HDFSA is a new timeslot that provide specifically for the unread tags in the subsequent reading cycle. HDFSA make use of the new timeslot allocation after the acknowledgement phase. If tags are going not to be acknowledged by the reader, then the unread tags will quickly occupy the exclusive timeslot prepared to allow them send their IDs. The simulations result showed that our proposed algorithm reduced the total delay time for identifying all the tags while preserving the accuracy of tag identification.

The remainder of this paper is organized as follows. Section 2 describes the existing ALOHA based tag anti-collisions protocols and Section 3 introduces the proposed Hasten Dynamic Frame Slotted ALOHA protocol. Section 4 then will present the simulation results and analysis, and finally Section 5 draws conclusions.

2 Aloha Based Tag Anti-collision Protocols

The basic of ALOHA based protocol in RFID system relies on the concept of transmitting the ID randomly whenever reader broadcast the query. If collision occurs, tags will need to wait for random time to retransmit the requested ID.

2.1 Slotted ALOHA (SA) Algorithm

SA is a tag identification method that is introduced to curb the problems of receiving data efficiently in ALOHA protocol. Instead of transmitting data randomly to the reader, the Slotted ALOHA algorithm brings in a number of slots for each tag transmits its serial number [6]. The reader will simply identify the tag when it receives the serial number without collision. However, if the number of tags exceeds the number of slot allocated, the tags are more likely to collide and it cannot guarantee the reasonable response time to identify all the tags [20].

2.2 Frame Slotted ALOHA (FSA) Algorithm

In order to guarantee the response time in SA algorithm, Philips Semiconductor [20] was proposed Basic Frame Slotted ALOHA (BFSA) to schedule the transmission of data. The used of frame in BFSA is advisable to support multiple tags reading by a single reader in a time. A frame was define by [14] as the time interval between requests of reader which consists of a number of slots.

In every tag identification process of BFSA, the fixed frame size and the random number will be broadcasted by a reader to be received by the tags [14]. The random number generated is then used to assign the slot of tags that will be slotted in. If the tag collision occurred, those collided tags have to reply to the next request of reader and this cycle of reading process will be repeated until there is no collision anymore. However, in certain circumstances such as when there are large number of tags involved [14], [15], there is a possibility that no tag is being identified although the read cycle is repetitious. This will reduce the efficiency of identification process. Based on this drawback, a study has proposed a new approach of anti-collision algorithm called Dynamic Frame Slotted ALOHA (DFSA).

2.3 Dynamic Frame Slotted ALOHA (DFSA) Algorithm

DFSA solves the problem in BFSA by dynamically regulating the frame size according to the number of tags. In the tag identification process, DFSA which is introduced by [13] will use information (ie. number of slot, and number of tags collided) from the used slot to determine the frame size in the next reading cycle. This approach indirectly makes the identification process of tags more efficient than existing BFSA. However, the frame sizes indefinitely change according to the number of tag [14], [15], [21], since its have an upper bound that limits the regulation. The performance also could be unsatisfied if the number of tags is higher than the permissible frame size [15] where it will reduce the number of collision [21]. As the technology advance, research on this algorithm is also being actively done to provide better tag allocation.

Enhanced Dynamic Frame Slotted ALOHA (EDFSA) Algorithm. Since that, Enhanced Dynamic Frame Slotted ALOHA (EDFSA) is introduced by [14] with a better adjustment of frame size by grouping the collided tags. By default, if the estimation number of tags is under the threshold, EDFSA does not have to group the unread tags. Otherwise, EDFSA will restrict the number of responding tags by dividing the tags into group nd only one group have to respond to the reader if the number of unread tags is higher than the threshold [15]. EDFSA's new method of grouping the tags offered better performance in reducing the tag collision probability and was improved the system efficiency. Furthermore, this fact has been supported by a few researchers [16], [21] where the system efficiency of EDFSA hovers around 36.8% better than DFSA.

Variant Enhanced Dynamic Frame Slotted ALOHA (VEDFSA) Algorithm. Even though EDFSA offered better performance than DFSA in reducing the tag collision probability, its group solution for unread tags is fixed. This means the group will not change during the whole reading process. Therefore, VEDFSA algorithm is proposed to overcome this problem. VEDFSA restructured the grouping mechanism in EDFSA using dynamic division of tags group. In VEDFSA, the tags in each group are going to give out signal and then being identified by a reader in a different group to achieve the optimal reading. In the next cycle of reading process, the collection of these collided tags will be grouped into a new group and this will decreases the retransmission of unread tags when the numbers of tags are very large [15]. Meanwhile in EDFSA, the unread tags in one group will be recognized repeatedly until its being identified.

Accelerated Frame Slotted ALOHA (AFSA) Algorithm. VEDFSA focused on improvement of retransmissions information but has not improved the inherent problem of tag collision probability in EDFSA. Therefore, V. Sarangan [16] was proposed a new tag anti-collision algorithm called Accelerated Frame Slotted ALOHA (AFSA) which comprised five phases to reduce the number of collision. The five phases are advertisement, reservation, reservation summary, data transmission, and acknowledgement phases respectively as shown in Figure 1 below. AFSA had changed the length of bit sequence in [14] algorithm's into the optimal value and this has resulted a better average tag reading time compared to EDFSA.

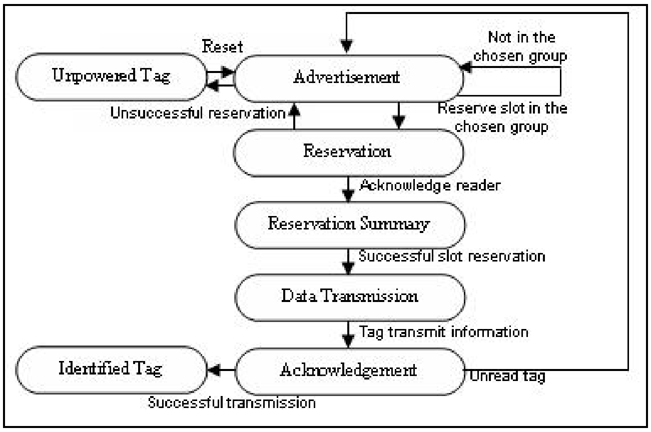

Fig. 1. AFSA framework

Step 1 – Advertisement Phase. For the first read cycle, the number of frame size (*N*) and number of groups (*M*) are broadcasted by reader. Upon receiving the values *N* and *M*, all tags will generate two random numbers. The first random number plays a role to determine either the tag can participate or not this first round. The tags are only allowed to participate if the remainder of modulo operation on first random number with M is zero. The second random number generated using uniform distribution once the tags allowed participating in first round. This second random number will determine the slot where the tags can transmit their data.

Step 2 – Reservation Phase. In the second phase, tags with the second random number generated in Step 1 will let reader knows which slot they have been chosen. Unlikely, in other slotted ALOHA protocols (FSA, DFSA, and EDFSA), tags will sent their data right away to the chosen slot. While AFSA approached proposed to send n bit sequence in their chosen slot, where n is a protocol parameter. For a given value n, there are 2^n possible n bit sequences. Every tag will randomly pick one of these 2^n sequences and transmits the value to the slot in the reader that has been chosen.

If the slot in the reader is succeeds receives the n bit sequence transmitted by tag, then it implies that the tag has successfully reserved this slot for transmitting its data. Since each tag transmits n bit sequence to the slot, the total duration of this phase will be $N*n$ bit times.

Step 3 – Reservation Summary Phase. During this phase, reader will let the tags know their status of reservation. The reservation status will be advertised through a bitmap of length N. For example, assuming that $N = 4$, and the bitmap is 1001, this 1001 n bit sequences indicate the successful slot that have been reserved by tag is slot number 1 and slot number 4. In other words, occurrence of bit 1 in location i indicates that only one tag had chosen slot i. Meanwhile, during slots 2 and 3, the reader did not receive any n bit sequence. This mean the occurrence of bit 0 in location i indicates two possibilities: (i) collisions – where there might be more than one tag chooses the slot. Thus, when the tags transmitted their random number on n bit sequences, collision occurred and consequently, the reader will not be able to decode the receive signal; or (ii) idle – means that the slot is not choose by any of the tags.

Step 4 – Data Transmission Phase. After step 3, all tags will be aware with their status of reservation. Only for those tags that are successful in their reservation are allowed to transmit their data in the next phase namely data transmission phase. Noted that if S is the total number of '1's in the N bit summary bit string, this means that only S data transmissions are possible. Therefore, there are only S data transmission into the chosen slot during data transmission phase, where $S \leq N$.

Step 5 – Acknowledgement Phase. This is the final phase in a round of reading cycle of tag identification. During this phase, reader wills acknowledges tag about the data that have been transmitted by tags. The acknowledgment is sent in the form of a bit '1' indicates that the transmission was received successfully and '0' indicates that the transmission was not successful.

All tags that received positive acknowledgement from the reader will become muted or killed, and will not participate in the next reading process again. The duration of this phase is S bit times.

Through the simulations and analyses conducted, AFSA had minimized the number of tag collisions and had reduced total wastage of bandwidth owing to unoccupied slots. Thus, AFSA gives better tag reading time over existing models.

Every phase in Figure 1, will perform a specific task within some duration time. Let T_{oc} is the total duration of a round reading cycle in AFSA for five different phases. In the complete process of tag reading, T_{oc} is computed as :

$$T_{oc} = T_p^1 + T_p^2 + T_p^3 + T_p^4 + T_p^5 \tag{1}$$

Where T_p^1 is duration for advertisement phase, T_p^2 is duration time for reservation phase, T_p^3 is duration time for reservation summary phase, T_p^4 is duration time for data transmission phase, and T_p^5 is duration time for acknowledgment phase respectively. This means in the first cycle of reading process contains T_p^i phases which is equivalent to :

$$T_{oc} = \sum_{i=1}^{RP} T_p^i \tag{2}$$

where *RP* is number of respective phases involved.

If there were T_{oc} times of reading cycles that the unread tag have been through until it is being identified, then the summation of T_{oc} for *n* number of cycle AFSA is computed as follow :

$$T_{AFSA} = \sum_{j=1}^{n} T_{oc}^j \tag{3}$$

However, the 'tag starvation' problem that discussed in the foregoing research [4] still suffering AFSA algorithm. An improvement still can be made in order to restrain the unread tag from being collide in the subsequent reading cycle. Therefore, we proposed a modified version called Hasten Dynamic Frame Slotted ALOHA (HDFSA) to handle the identification of unread tags in the subsequent reading cycle.

3 Hasten Dynamic Frame Slotted Aloha (HDFSA)

Hasten Dynamic Frame Slotted ALOHA (HDFSA) was inspired based on AFSA algorithm. The basic idea of HDFSA protocol is to improve the processing time of tag identification process without compromising the accuracy of data transmitted. HDFSA implementation retained the enhancements incorporated into AFSA, but modified the subsequent reading cycle operation to include HDFSA mechanism. The processing time of HDFSA algorithm was measured based on the elapsed time calculated during the identification process. While the time-complexity of HDFSA algorithm only consider on the process of HDFSA in handling the unread tags.

3.1 HDFSA Protocol

Figure 2 below illustrates the proposed framework of HDFSA. HDFSA protocol has prevented the unread tags from going into the same phases as in the first reading cycle in AFSA. As a resulted, HDFSA has condensed the number of retransmission for one tag, thereby reducing the time of tag identification process. It is rule of thumb where the number of collision is reduced when reducing the retransmission of tag. HDFSA improves the subsequent read cycles of unread tags by providing a new mechanism once the tag cannot be read.

In Figure 2, if there is no acknowledgement from the reader to the tag in the fifth phase, then this tag is consider as collided tag or unread tag. HDFSA operates by assuming that each unread tags will be provided with a new slot for them to send their information. Before tags are sending their information, booking number has been given to each unread tag which refers as the slot where the tag will send its information. One tag will be given one booking number which is according to the turn that the tag cannot be read. For example, if Tag 5 is the first tag that can not be identify, then Tag 5 will be given number '1' as its booking number which refer to the slot number 1. It is means that every slot will only contain one tag in one time. Thus, during the subsequent cycle of reading process, HDFSA algorithm allows the unread tags to send their data directly to the certain slot which has already assigned according to the given booking number. It is then reduced the number of retransmission of one tag in order to identify again when it is being collided in the first reading cycle.

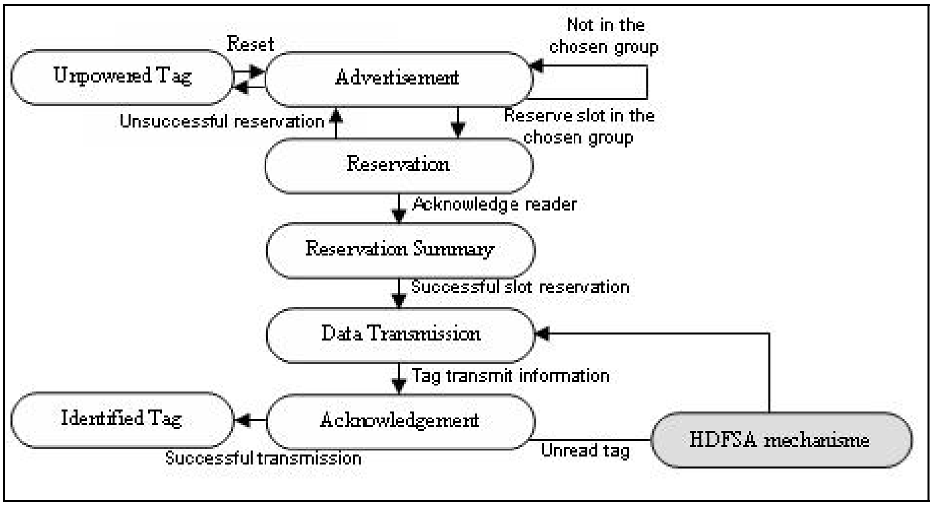

Fig. 2. HDFSA theoretical framework

In HDFSA, when tags are not being identified in the first reading cycle, the reader will prepare a new slot based on the number of unread tags. Every unread tag then will be given with a number called booking number. This booking number is referring to the slot number where the unread tags have to send their information. One tag will be assigned with one booking number. Thus, only one tag will be slotted into the slot and being identified. This will reduce the retransmission of tag and consequently reduce the time used in one identification process.

Let T_{sr} is the total duration of subsequent read cycle, the duration for this cycle should be as follow:

$$T_{sr} = T_p^3 + T_p^4 + T_p^5 \tag{4}$$

Since the unread tag will only through the HDFSA mechanism phase, data transmission and acknowledgement phase in its next reading cycle, the duration of time involves in identifying one tag is reduced as computed in equation 4.

Consider the total duration of first reading cycle like equation 1. Thus, the total duration T_H of one process of HDFSA algorithm are computed as (5) and (6) equation.

$$T_H = T_{oc} + T_{sr} \tag{5}$$

$$T_H = \sum_{i=1}^{RP} T_p^i + \sum_{i=3}^{RP} T_p^i \tag{6}$$

Thus, for equation (6) and (3) we can say that the total duration of HDFSA is less than total duration of AFSA i.e., $T_H < T_{AFSA}$

Algorithm 1. HDFSA tag anti-collision algorithm operation

```
BEGIN
Step0. Initialization
Step 1. Advertisement Phase
  for all tag, tag < TOTALTAGS
    Reader r1 broadcast N and M
    (where N=frame size, M=group number)
    Tag generate 2 random number rnd
            if first rnd mod N = 0
                TagObject[i] participate in the cycle
            if TagObject[i] participate
                TagObject[i].2nd rnd
                (where second rnd refer to number of slot choose)
  calculate duration time for advertisement phase
Step2.Reservation Phase
  for all tag, tag < TOTALTAGS
    TagObject[i] broadcast the slot number to r1
    TagObject[i] sent n bit sequences to the chosen slot
            if slot[i] = n bit sequence
                slot[i] successfully reserved
    calculate duration time for reservation phase
Step 3. Reservation Summary Phase
    r1 let tag know reservation status
   calculate duration time for reservation summary phase
Step 4. Data Transmission Phase
  for all tag, tag < TOTALTAGS
    TagObject[i] transmit data
  calculate duration time for data transmission phase
Step5. Acknowledgment Phase
  for all tag, tag < TOTALTAGS
    TagObject[i] acknowledge r1
  calculate duration time for acknowledgement phase
    if Ack.TagObject[i] = 1
            TagObject[i] = sleeping
    else
            prepare new_slot
            Nsleep
            fsize = Nsleep
            TagObject[i].booking_number
            (where booking number equivalent to slot number)
            TagObject[i] transmit data
            TagObject[i] acknowledge reader
END
```

Fig. 3. HDFSA algorithm

3.2 HDFSA Algorithm

From Figure 2, the most important part of HDFSA algorithm is the way this algorithm handle the unread tag once it has been failed to be identify after through the first round of reading cycle. A clear step can be seen in step 33 of HDFSA algorithm that illustrated by Figure 3.

After the last stage of acknowledgement phase in the first reading cycle, a new slot will be prepared by HDFSA to allocate the unread tag. For every tag that has been read, reader will send a positive acknowledgement to the tag and become muted (line 31). Meanwhile, for the tags that are failed to be read, reader will send a negative acknowledgement. Therefore, in HDFSA, a new slot is created for this specific unread tag. The algorithm of HDFSA begins by preparing a new slot in Step 32 once the tag is not being acknowledged by the reader. In Step 33, a new slot is prepared to allocate the unread tag. This unread tag will be given a booking number that represent the number of a slot that they will transmit their data such present in Step 36. The booking number is generated according to the turn that tag being not identified. For example, if tag C is the first tag that cannot be identified then the reader will generate booking number '1' for tag C which means a new slot for tag C is slot number 1. By using this booking number, tag C will then be sent their data directly to the allocating slot as in Step 37. In Step 38 this tag will then be acknowledged by reader that it was being identified.

3.3 Computational Complexity

Based on HDFSA algorithm shows in Figure 3, time complexity is used to evaluate the complexity of algorithm proposed.

Time complexity can be expressed as the number of operations that algorithm have to execute when input has a particular size. We evaluate the complexity of algorithm exclusively on the segment where the tag is started being unread which begins from Step 30 as illustrates in Figure 3 and Figure 4.

In the HDFSA algorithm (see Figure 3), to handle the unread tag, HDFSA has to perform one loop such in Step 30. It is then followed by twenty two operations inside the loop. Thus, 22n+2 operation is used in HDFSA algorithm and the time complexity is *O(n).*

In contrast with AFSA algorithm, there are 7 *for* loops to execute as illustrates in Figure 4 below. 5 loops refer to the five phases in AFSA algorithm and the rest of two loops refer to initialization process. AFSA algorithm handles the unread tag inside the *while* loop. Inside the *while* loop there are $14+14n$ operations involved. For first time through the *while* loop, it takes $1 + (14 + 14n)$, which $14n + 15$. Second time through the *while* loop, it takes $14(N - 1) + 15$ operations, $14(N - 2) + 15$ until $14(1) + 15$ the end of operations. The operations performed to handle the unread tag in AFSA can be simplified as below.

$$\begin{aligned}&(14n+15)+(14(n-1)+15)+\ldots\ldots+(14(2)+15)+(14(1)+15) =\\&14[n+(n-2)+\ldots..+2+1]+15n=14n(n+1)/2+15n\\&\quad = 7n^2+7n+30n\\&\quad = 7n^2 + 37n\end{aligned}$$

Thus, $7n^2 + 37n$ operation are used to handle the unread tags in AFSA and the time complexity is $O(n^2)$ where it is larger than HDFSA algorithm.

Algorithm 2. AFSA tag anti-collision algorithm operation

```
1   BEGIN
2   Step 0. Initialization
3   Step 1. Advertisement Phase
4     for all tag, tag < TOTALTAGS
5         Reader r1 broadcast N and M
          (where N=frame size, M=group number)
6         Tag generate 2 random number rnd
              if first rnd mod N = 0
8                     TagObject[i] participate in the cycle
9             if TagObject[i] participate
10                    TagObject[i].2nd rnd
                      (where second rnd refer to number of slot choosed)
11  culculate duration time for advrtisement phase
12  Step 2. Reservation Phase
13    for all tag, tag < TOTALTAGS
14        TagObject[i] broadcast the slot number to r1
15        TagObject[i] sent n bit sequences to the chosen slot
16            if slot[i] = n bit sequence
17                    slot[i] successfully reserved
18  culculate duration time for reservation phase
19  Step 3. Reservation Summary Phase
20        r1 let tag know reservation status
21  culculate duration time for reservation summary phase
22  Step 4. Data Transmission Phase
23    for all tag, tag < TOTALTAGS
24        TagObject[i] transmit data
25  culculate duration time for data transmission phase
26  Step 5. Acknowledgment Phase
27    for all tag, tag < TOTALTAGS
28        TagObject[i] acknowledge r1
29  culculate duration time for acknowledgement phase
30        if Ack.TagObject[i] = 1
31            TagObject[i] = sleeping
32  while (!done)
33        Estimate tag count
34        Set frame size
35        Set group
36        Repeat Step 1
37        Repeat Step 2
38        Repeat Step 3
39        Repeat Step 4
40        Repeat Step 5
41  END
```

Fig. 4. AFSA algorithm

4 Results and Analysis

In this section, we compare the performance of HDFSA with AFSA based on the elapsed time taken by both algorithms.

Table 1 below presents the comparison of the elapsed time in millisecond (msec) between HDFSA and AFSA algorithm. The experiment conducted considered five different input value of number of tags to be identified. From Table 1, Figure 5 illustrates the number of comparison into graph. It is noted that the elapsed time of AFSA algorithm gradually increased when the number of tags is significantly high. This is due to the number of phases that the unread tags have to recursively through once it is failed to be identified in the first reading cycle.

However, HDFSA require less processing time as shows in Table 1 where the elapsed time to complete the identification process is less as compared to AFSA algorithm. Figure 5 illustrated that the elapsed time of AFSA algorithm is gradually increased with the number of tags. Mean while HDFSA's elapsed time is remained stable with the number of tags. For example, from Table 1 and Figure 4, when there are 300 of tags presented in the interrogation area of reader, AFSA algorithm acquire 80,000 millisecond to complete one process to identify all tags, meanwhile in AFSA only 10,000 millisecond required to perform one complete process to identify 300 of tags.

This showed that HDFSA algorithm is more efficient compared to AFSA algorithm up to more than two orders of magnitude. Therefore, HDFSA algorithm outperform AFSA algorithm approximately 85 percent.

Table 1. Comparison of the elapsed time between HDFSA and AFSA

Number of Tag	HDFSA(msec)	AFSA(msec)
100	10,000	40,000
200	10,000	56,000
300	10,000	80,000
400	14,000	118,000
500	18,000	138,000
600	18,000	164,000
700	20,000	192,000

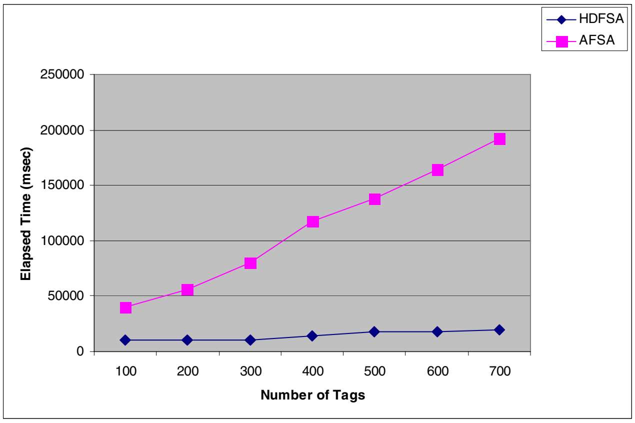

Fig. 5. Simulation results for elapsed time of HDFSA and AFSA

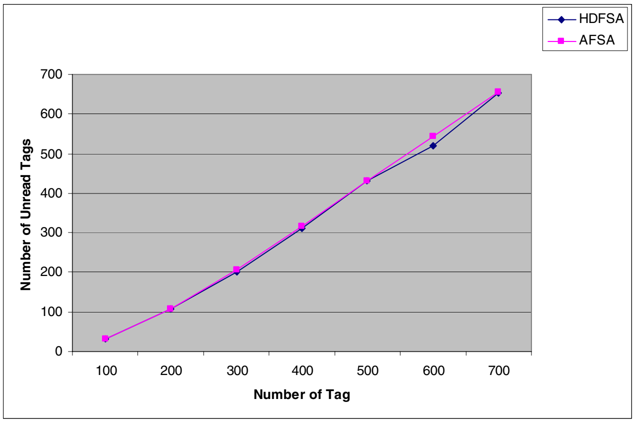

Fig. 6. The number of unread tags in one process

Figure 6 shows the average number of unread tags that obtained in one identification process for HDFSA and AFSA algorithm by increasing the number of tags. It is obviously that HDFSA algorithm performs as well as AFSA algorithm with almost the same average number of unread tags in each process. Thus, the accuracy of HDFSA algorithm is remain as AFSA algorithm.

5 Conclusion

This research was focused on tag collision problem that occurred among the passive tags during identification process in RFID system. A new approached inspired from AFSA algorithm, called HDFSA was proposed. HDFSA is likely reduced the processing time with a new technique introduced in handling the unread tags. HDFSA provide a new slot for the unread tags to allow tags directly transmit their data instead of through the five phases as in AFSA algorithm. The simulation of RFID static reading system developed with HDFSA tag anti-collision revealed that HDFSA outperformed the present AFSA algorithm up to 78% in terms of processing time with less time complexity, while preserving the accuracy of tag identification.

References

1. Finkenzeller, K.: RFID Handbook. John Wiley & Sons, Inc., New York (1999)
2. Shepard, S.: RFID: Radio Frequency Identification. McGraw-Hill, New York (2005)
3. Krebs, D., Liard, M.J.: White Paper: Global Markets and Applications for Radio Frequency Identification (2001)
4. Myung, J., Lee, W.: Adaptive Binary Splitting for Efficient RFID Tag Anti-Collision. IEEE Communications Letters 10, 144–146 (2006)
5. Ho-Seung, Kim, J.-H.: Anti-Collision Algorithm Using Bin Slot in RFID System. In: Proceeding of the TENCON (2005)

6. Seol, J.-M., Kim, S.-W.: Efficient Collision-resilient RFID Tag Identification using Balances Block Design Code. In: Sixth IEEE International Conference on Computer and Information Technology (CIT), p. 220. IEEE Xplore, New York (2006)
7. Law, C., Lee, K., Siu, K.-Y.: Efficient Memoryless Protocol for Tag Identification. In: The 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications Boston. Massachusetts, United States (2000)
8. Hush, D.R., Wood, C.: Analysis of Tree Algorithms for RFID Arbitration. In: The IEEE International Symposium on Information Theory. Cambridge, MA, USA (1998)
9. Abramson, N.: The ALOHA System: Another Alternative for Computer Communications. In: The AFIPS Joint Computer Conferences, Houston, Texas (1970)
10. Metcalfe, B.: Steady-state Analysis of A Slotted and Controlled Aloha System with Blocking. ACM SIGCOMM Computer Communication Review 5, 24–31 (1975)
11. Vogt, H.: Efficient Object Identification with Passive RFID Tags. In: Mattern, F., Naghshineh, M. (eds.) PERVASIVE 2002. LNCS, vol. 2414, p. 98. Springer, Heidelberg (2002)
12. Schoute, F.C.: Dynamic Frame Length ALOHA. IEEE Transactions on Communications 31(4), 565–568 (1983)
13. Cha, J.-R., Kim., J.-H.: Novel Anti-collision Algorithms for Fast Identification in RFID System. In: The 11th International Conference on Parallel and Distributed Systems, ICPADS 2005 (2005)
14. Lee, S.-R., Joo, S.-D., Lee, C.-W.: An Enhanced Dynamic Frame Slotted ALOHA Algorithm for RFID Tag Identification. In: The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous 2005 (2005)
15. Peng, Q., Zhang, M., Wu, W.: Variant Enhanced Dynamic Frame Slotted ALOHA Algorithm for Fast Object Identification in RFID System. In: Proceeding of the IEEE Int. Workshop on Anti-Counterfeiting, Security, Identification (2007)
16. Sarangan, V., Devarapalli, M.R., Radhakrishnan, S.: A Framework for Fast Tag Reading in Static and Mobile Environments. Computer Networks: The International Journal of Computer and Telecommunications Networking 52, 1058–1073 (2008)
17. Khandelwal, G., Lee, K., Yener, A., Serbetli, S.: ASAP: A MAC Protocol for Dense and Time-Constrained RFID Systems. EURASIP Journal on Wireless Communications and Networking 9, 4028–4033 (2007)
18. Klair, D.K., Chin, K.-W., Raad, R.: The Suitability of Framed Slotted Aloha Based RFID Anti-Collision Protocols for Use in RFID-Enhanced WSN's. In: The 16th International Conference on Computer Communications and Networks (ICCCN), Honolulu (2007)
19. Myung, J., Lee, W.: Adaptive Splitting Protocols for RFID Tag Collision Arbitration. In: Proceeding of the MobiHoc 2006, Florence, Italy (2006)
20. Semiconductor, P.: I-Code1 System Design Guide: Technical Report (2002)
21. Bonucelli, M.A., Lonetti, F., Martelli, F.: Tree Slotted ALOHA: A New Protocol for Tag Identification in RFID Networks. In: The International Symposium on World of Wireless, Mobile, and Multimedia Networks, WoWMom 2006 (2006)