Adaptive Splitting and Pre-Signaling for RFID Tag Anti-Collision

Ming-Kuei Yeh, Jehn-Ruey Jiang and Shing-Tsaan Huang

Department of Computer Science and Information Engineering
National Central University
Taiwan, R. O. C.

All correspondence should be addressed to:

Dr. Jehn-Ruey Jiang
Department of Computer Science and Information Engineering
National Central University
Jhongli, Taiwan, R. O. C.

Tel:886-3-4227151
Fax:886-3-4222681
E-mail:jrjiang@csie.ncu.edu.tw

Abstract — In an RFID system a reader requests tags to send their IDs by RF signal backscattering for the purpose of identification. When multiple tags respond to the request simultaneously, tag collisions occur and the tag identification performance is degraded. There are several tag anti-collision protocols proposed for reducing tag collisions. The protocols can be categorized into three classes: ALOHA-based, tree-based and counter-based. ALOHA-based protocols have the tag starvation problem; tree-based protocols have the problem that their performances are influenced by the length and/or the distribution of tag IDs. On the contrary, counter-based protocols do not have such problems. In this paper, we propose a counter-based tag anti-collision protocol, called ASPS, to reduce tag collisions by adaptively splitting tags encountering collisions into several groups according to the estimated number of tags to be split, and to reduce the number of messages sent between the reader and tags by utilizing a pre-signaling bit. We simulate and analyze ASPS and compare it with related ones to show its advantages.

Keywords — Collision resolution, RFID, anti-collision, tag identification

1 INTRODUCTION

The RFID (Radio Frequency IDentification) technique attracts a lot of attention recently due to its automatic identification capability through RF communication [1]. An RFID system consists of a reader and one or more tags. Tags store unique IDs and are attached to objects; a reader recognizes an object by issuing RF signals to interrogate the ID of the attached tag. According to the source of power supply, tags are classified into two types: active tags, which contain a battery and can transmit signals autonomously, and passive tags, which contain no battery and derive energy from the RF field generated by the reader to transmit signals passively. Most RFID tags are passive; they have the advantage over other electronic products that are energized by batteries or other power sources. Furthermore, tags are usually of tiny sizes and low costs. The RFID system is thus suitable for many applications, such as logistic control, supply chain management, and asset tracking, etc.

When a tag and a reader are close enough, they can communicate with each other. For such a situation, we say that the tag is in the *interrogation zone* of the reader. To figure out which tags are within the interrogation zone, a reader initiates an identification procedure (or interrogation procedure) to request tags to send back their IDs. When multiple tags respond to the reader simultaneously, tag collisions occur and no tag can be identified by the reader successful. How to reduce tag collisions to speed up the identification procedure is thus important. There are several tag anti-collision protocols proposed for reducing tag collision. According to [2], they can be categorized into two classes: ALOHA-based protocols and tree-based protocols that include deterministic tree-based and probabilistic counter-based subclasses of protocols.

In ALOHA-based protocols [3-7], tags respond to the reader by transmitting IDs in a probabilistic manner. For example, in slotted ALOHA protocol [4], the whole interrogation procedure period is divided into several time slots, and each tag randomly chooses a time slot for transmitting its ID to the reader. ALOHA-based protocols are simple; however, they have the *tag starvation problem* that a tag may never be successfully identified because its responses always collide with others'.

The basic idea of the tree-based protocol is to repeatedly split the tags encountering collisions into subgroups until there is only one tag in a subgroup to be identified. The tree protocols do not have the tag starvation problem. In order to emphasize the different mechanisms for performing the tag-splitting based on either static tag IDs or dynamic counters, we classify the tree-based protocols into deterministic tree-based [8-11] and probabilistic counter-based [12-14] subclasses of protocols. The deterministic tree-based protocol relies on tag IDs and thus has the problem that its performance is influenced by the tag ID length and/or distribution, while the probabilistic counter-based protocol has not. We hence focus on counter-based protocols in this paper.

This paper presents a novel counter-based tag anti-collision protocol, called ASPS, using two schemes, adaptive splitting and pre-signaling, to reduce tag collision. By predicting the number k of tags to be split, ASPS adaptively splits tags into k groups. It is likely that each group has only one tag to be identified successfully. In this way, collision is reduced significantly. Furthermore, ASPS utilizes a pre-signaling bit to reduce the number of messages sent between the reader and tags. The tag identification delay is thus reduced. We simulate and analyze ASPS and compare it with related ones to show its advantages.

The rest of this paper is organized as follows. Some related work is introduced in Section 2. In Section 3, we describe ASPS protocol by elaborating the concepts of adaptive splitting and pre-signaling. In Section 4, we simulate and analyze ASPS and compare it with related protocols. And finally, conclusion is drawn in Section 5.

2 RELATED WORK

In this section, we introduce some representative ALOHA-based, deterministic tree-based and probabilistic counter-based tag anti-collision protocols.

2.1 ALOHA-Based Protocols

ALOHA-based protocols try to stagger tag response times in a probabilistic manner to reduce collisions. Below, we introduce some ALOHA-based protocols: ALOHA [3], slotted ALOHA [4], frame slotted ALOHA [5], and dynamic frame slotted ALOHA [6-7] protocols.

In ALOHA protocol [3], on receiving the reader's interrogation request, each tag in the interrogation zone independently chooses a random back-off time and responds its tag ID to the reader at that time. If an ID is received by the reader without collision, it can be identified properly and acknowledged by the reader. A tag with acknowledged ID will stop responding to the reader. On the other hand, an unacknowledged tag will repeatedly select a random back-off time to send its ID until it is identified and acknowledged by the reader. In slotted ALOHA protocol [4], the random back-off time must be a multiple of a pre-specified slot time. If collisions occur in a slot, the reader will notify the colliding tags to re-select a response time randomly. As shown in [15], the performance of slotted ALOHA protocol is twice that of ALOHA protocol since there is no partial collision of tag ID responses in the former protocol.

Frame slotted ALOHA protocol [5] is similar to slotted ALOHA protocol. However, to limit the response time, frame slotted ALOHA protocol divides the whole interrogation procedure into a set of frames. Each frame has a fixed number of time slots, and a tag sends its ID to the reader in only one randomly chosen slot during a frame period. One drawback of frame slotted ALOHA protocol is that its performance will degrade when the number of slots in the frame does not properly match with the number of tags in the interrogation zone. Dynamic frame slotted ALOHA protocols [6-7] try to eliminate the drawback by dynamically adjusting the frame size according to the estimated number of tags. They are therefore have better performance slotted ALOHA protocol. But they need many communication rounds to optimize the frame size before the identification process [6]. Under the assumption that tag IDs are with the same series in production (i.e., tags have the continuous tag ID numbers), paper [16] proposed LoF (Lottery Frame) protocol to reduce the number of communication rounds from O(n) to O(log n) with the help of the geometric distribution hash function, where n is the total number of tags in the interrogation zone.

In general, ALOHA-based protocols are simple and have fair performance. However, some ALOHA-based protocols have the tag starvation problem that a tag may never be identified when its responses always collide with others'.

2.2 Deterministic Tree-Based Protocols

Deterministic tree-based protocols rely on tag IDs to repeatedly split colliding tags into subgroups until there is only one tag in a subgroup to be identified successfully. Below, we introduce two representative tree-based protocols: query tree [8] and bit-by-bit binary tree [9] protocols.

In query tree protocol (QT) [8], a reader first broadcasts a bit string *S* of a specified length. The tag with an ID whose prefix matches with S will respond its whole ID to the reader. If only one tag responds at a time, the tag is identified successfully. But if multiple tags respond simultaneously, the responses collide. In such a case, the reader appends string *S* with bit 0 or 1 and broadcasts again the longer bit string (i.e., *S*0 or *S*1). In this manner, the colliding tags are divided into two subgroups. If there is only one tag in a subgroup, it can be identified successfully. The reader keeps track of the request strings needed to broadcast with the help of a stack and perform tag identification procedure until all tags are identified. QT protocol is a memory-less protocol because it does not require tags to be equipped with additional writable on-chip memory. QT protocol does not have the tag starvation problem and its identification delay is affected by the distribution and the length of tag IDs. Specifically, if the tags have long and continuous IDs, the request bit string will grow very quickly for identifying all tags. The delay time of the identification procedure will then increase significantly.

In bit-by-bit binary tree (BBT) protocol [9], on receiving a reader's interrogation request, each tag responds with the first bit of its tag ID. The reader then records and broadcasts 1 (resp., 0) if the received bit is 1 (resp., 0 or a colliding signal). Only the tags with the first bit being 1 (resp., 0) will respond with its next ID bit; other tags will go into a sleep mode. The above procedure will repeat bit by bit until the last ID bit is reached. The reader can then identify and mute one tag, and reset tags in the sleep mode to go through the interrogation procedure from some ID bit position. The bit-by-bit procedure is performed recursively and all tags can be identified. BBT protocol requires tags to be equipped with writable on-tag memory so that tags can keep track of the inquiring bit position. Like QT protocol, BBT protocol has no tag starvation problem and its performance is dependent on tag ID distribution and/or length.

2.3 Probabilistic Counter-Based Protocols

Probabilistic counter-based protocols rely on dynamically changing counters to split colliding tags. Below, we introduce two probabilistic counter-based protocols, ISO/IEC 18000-6B tag anti-collision protocol [13] and ABS (Adaptive Binary Splitting) protocol [12].

The well known ISO/IEC 18000-6B standard [13] proposes a probabilistic counter-based tag anti-collision protocol (later we just name it *ISO/IEC 18000-6B protocol* for short). In the protocol, each tag maintains a counter which is initially 0. Every tag with counter value 0 can transmit its tag ID to respond to the reader's interrogation request. When a collision occurs, the reader will notify all tags of this. And the tags with counter values larger than 0 will increase their counters by 1, while the tags with counter value 0 will randomly add 0 or 1 to their counters. In this way, the colliding tags (i.e., the tags with counters value 0) are split into two subgroups. The splitting procedure will be repeated until there is only one or no tag with counter value 0. In the former case, the tag with counter value 0 can be identified successfully. And in both cases, the reader sends a command to inform all unidentified tags to decrease their counters by 1. In this way, every tag will be the unique one to have counter value 0 and be identified successfully.

Adaptive Binary Splitting (ABS) protocol [12] is proposed to improve ISO/IEC 18000-6B protocol by keeping tags' counter information of the last interrogation round. A tag in ABS protocol keeps two counters. The first counter (Allocated Slot Counter, ASC) is similar to that of ISO/IEC 18000-6B protocol, and the second counter (Progressed Slot Counter, PSC) is to keep track of the number of tags identified successfully. The two counters are initially 0 in the first round, but only PSC is reset to be 0 in following rounds. Tags with ASC equal to PSC can transmit their tag IDs to respond to a reader request. When there is only one response, the responding tag can be identified and each tag increases PSC by one. When there is no response, all tags with ASC larger than PSC decrease ASC by one. When collisions occur, the tags with ASC larger than PSC then increase ASC by 1, while the tags with ASC equal to PSC randomly generate a random bit, 0 or 1, and add it to ASC. Note that tags with ASC less than PSC do not increase ASC; they even do not attempt to transmit their IDs until the tag interrogation round is finished. After all tags are identified in a round, they have unique and successive ASC values. These values can be reserved for use in the next tag interrogation round to speed up the interrogation procedure. Even if there are tags joining or leaving after the last interrogation round, ABS protocol can work properly. As shown in [12], the performance of ISO/IEC 18000-6B protocol is improved significantly by ABS protocol.

In general, probabilistic counter-based protocols do not have the problem of tag starvation. Furthermore, they have the stable property that their performances are not affected by the tag ID distribution or ID length.

3 THE PROPOSED PROTOCOL

In this section, we propose a novel probabilistic counter-based tag anti-collision protocol, called ASPS, which uses two schemes, adaptive splitting and pre-signaling, to reduce tag collision and the number of messages sent between a reader and tags. Below, we introduce the two schemes respectively.

3.1 Adaptive Splitting

In this subsection, we introduce the *adaptive splitting* scheme for speeding up the identification procedure. When there are tags whose responses collide, a typical counter based tag anti-collision protocol, such as ISO/IEC18000-6B protocol, split colliding tags into two subgroups no matter how many colliding tags are. The idea of adaptive splitting scheme is to estimate the number k of colliding tags and to split the colliding tags into k groups to speed up the identification procedure.

There are two phases in the adaptive splitting scheme. In the first phase, the adaptive splitting scheme obeys ISO/IEC 18000-6B protocol until the first tag is identified successfully. It then enters the second phase, in which the reader estimates the number of colliding tags of a specific counter value. Unlike other solutions that estimate the total number of tags before the identification process (e.g., Kodialam and Nandagopa's protocol [6]), ASPS scheme needs not go through a separate tag quantity estimation stage. Below, we explain how a reader does the estimation.

We propose the *primitive splitting group* (*PSG*) concept for the adaptive splitting scheme. As the first tag is identified, the tags having the same counter value are assumed to be in the same *PSG*. The *PSG* groups are indexed by 1, 2, …, *max_idx* at the end of the first phase according to the top-to-down and right-to-left order in the counter-based splitting tree (refer to Fig. 1). Note that *max_idx* is the index of the *PSG* to which the first-identified tag belongs.

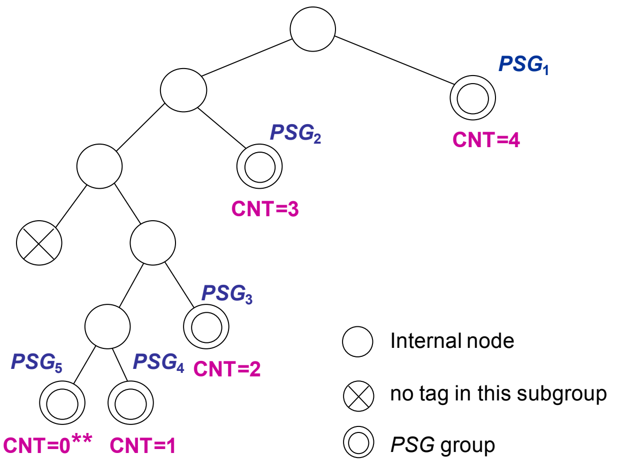

Figure 1. An example of *PSG* groups and their associate counter (CNT) values in a counter-based splitting tree (the first identified tag belongs to PSG_5, i.e., $max\_idx = 5$)

As shown in [17], if there are k tags in a group to be identified, then it has the best performance to directly split the tags into k subgroups. In the second phase, the reader estimates the number of the tags in each *PSG* (in the order from index max_idx-1 to index 1) and splits the tags according to the estimated number. The estimation is based on the observation that there are about the same number of tags associated to the sub-trees rooted at two sibling nodes in the splitting tree, since colliding tags are expected to be split into 2 groups of approximately equal sizes. For example, in Fig. 2, the sub-tree rooted at node PSG_3 has two associated tags, and the sub-tree rooted at the sibling node of node PSG_3 also has two associated tags. Therefore, the reader can easily estimate the number k_e of tags in PSG_e just by figuring out the number N_e of tags associated with the sub-tree rooted at the sibling node of PSG_e. Actually, N_e equals the total number of tags in the *PSG* groups indexed by e+1, e+2, ... , *max_idx*. We have

$$N_e = \sum_{i=e+1}^{max\_idx} |PSG_i| \qquad (1)$$

We now can use N_e as an estimation of k_e and split tags in PSG_e into N_e (instead of 2 in ISO/IEC 18000-6B protocol) subgroups directly. This can be done by making each tag in PSG_e randomly choose a value between 0 and (N_e-1) as its new counter value. At the same time, unidentified tags, i.e., tags in PSG_{e-1}, $PSG_{e-2}, \ldots$, PSG_1, should add (N_e-1-c) to their counter values, where c is the counter value of the tags in PSG_e before splitting.

By the adaptive splitting scheme, we expect to split tags in PSG_e into N_e subgroups such that each subgroup has exactly one tag (refer to Fig. 2). But it may not be so lucky due to the probabilistic characteristic of tag splitting and due to the fact that PSG_e may not have exactly N_e tags. If there are two or more tags in one subgroup, ISO/IEC18000-6B protocol is applied to identify those tags. Fortunately, as shown in [18], it will not have too many tags in one subgroup if N_e approximates k_e. Therefore, it will be efficient enough to identify tags in a single subgroup by recursively applying ISO/IEC18000-6B protocol.

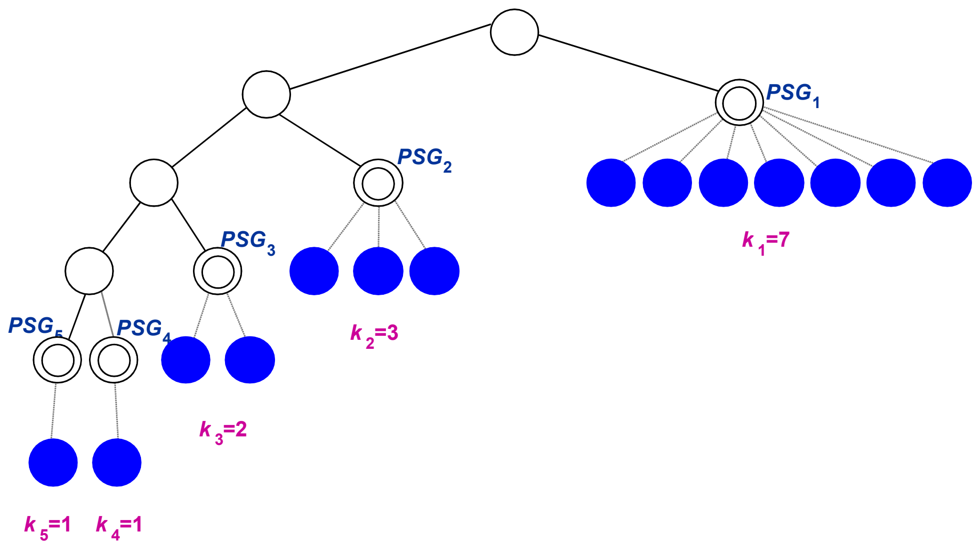

Figure 2. An example of the adaptive splitting scheme

3.2 Pre-Signaling

To enhance the performance of the identification procedure, we propose the idea of *pre-signaling* under the assumption that the reader can distinguish three cases of identical responses from multiple tags by signal strength inspection. The cases are 1) no-response, 2) one-response and 3) multiple- response. Note that it is possible for a reader to detect multiple tags responding the same bit at the same time since tags' response time may be biased (for example, +/-15% in ISO18000-6B standard [13]) and the superposition of multiple tags' signals may give some clues [1]. However, we will not go into the details of the hardware design in this paper.

By the anti-collision rule in ISO/IEC18000-6B protocol, only the tags with counter value 0 can respond its ID to the reader in the response window. We slightly modify this to be the pre-signaling scheme that the tags with counter value 1 will respond 1 at the first bit transmission period of the response window. After the first bit transmission period, the tag with counter value 0 will then respond its ID bit by bit normally.

The bit transmitted by the tags with counter value 1 is called *pre-signaling bit*. The value of pre- signaling bit received by a reader can be *null*, *single_1* and *multiple_1*, which correspond to the three tag response cases of no-response, one-response and multiple-response, respectively. According to the value of pre-signaling bit, the reader takes one of the following three actions.

(Case 1) The pre-signaling bit is "null"

In this case, no tag is with counter value 1. After all tags with counter value 0 are identified, the reader will inform all unidentified tags to decrease their counters by 2. This will speed up the original procedure of ISO/IEC18000-6B protocol, in which the reader informs unidentified tags to decrease their counters by one, waits awhile for tags to respond (there is certainly no response in that case), and then informs again all unidentified tags to decrease their counters by one.

(Case 2) The pre-signaling bit is "single_1"

In this case, only one tag is with counter value 1, which implies it can be identified successfully without collision in the next iteration of tag identification procedure. The reader does not take any extra action for this case. It just follows the original procedure of ISO/IEC 18000-6B.

(Case 3) The pre-signaling bit is“multiple_1”

In this case, there are multiple tags with counter value 1. The reader can inform those tags to split into *k* subgroups just after all the tags with counter value 0 are identified, where *k* is the estimated number of tags with counter value 1. To be more precise, the tags with counter value 1 will then be of counter value 0, 1, …, or *k*–1 randomly. This will speed up the original procedure of ISO/IEC18000-6B protocol, in which the reader informs unidentified tags to decrease their counters by one, waits awhile for tags to respond (there are certainly multiple responses in that case), and then informs all tags with counter value 0 to increase their counters by 0 or 1, and informs all tags with counter value greater than 0 to increase their counters by 1.

By the help of pre-signaling scheme, the number of messages sent between a reader and tags can be significantly reduced, which in turn shortens the identification procedure latency. Consequently, with both adaptive splitting and pre-signaling schemes, the performance of tag interrogation can be improved dramatically. We will justify this by extensive simulation experiments in the next section.

3.3 Operation Details

In this subsection, we elaborate the details of ASPS. The reader operation is shown in Algorithm1.To start a new interrogation round, the reader first sends an NR (New Round) command to tags. It then sends an *RC* (Reader Command) command with *RC.status=Request* to ask tags to respond. On receiving the command, the tags with counter value 1 will respond 1 at the first bit transmission period of the response window, and the tags with counter value 0 will then respond its whole ID bit by bit.

When the reader receives responses from tags, it will extract the first bit as the pre-signaling bit and other bits as the tag ID.

There are two phases, Phase I and Phase II, in a round. Phase I continues until the first tag is identified successfully (lines 7-24). In Phase I, the reader follows the original identification procedure of ISO/IEC 18000-6B protocol to identify tags. At the end of Phase I, the reader can figure out the maximum index (viz., *max_idx*), and all *PSG* groups' counter values (viz., *PSG*[*i*].*cnt*), $0 \le i \le max\_idx$. It can also find out the maximum counter value (viz., *max_cnt*) of tags.

In Phase II, the adaptive splitting scheme is enforced. The reader will send an RC command with *RC.status = Collision, Success, Pre_Split* or *Pre_Collision* according to the tag responses. The reader keeps track of the maximum counter value *max_cnt* and terminates Phase II when *max_cnt* is zero. Below, we describe the reader operations. It is noted that we use variable *next_idx* to keep track of the index of the *PSG* that will be next processed by the reader, and we just use "*the next PSG*" to indicate such a *PSG*.

- Lines 28-31: If there is any tag collision, the reader sends an RC command with *RC.status=Collision* to notify tags of the collision.

- Lines 37-42: The reader estimates the tag number *k* in the next *PSG* by formulas (1)-(3), and sends an *RC* command with *RC.status=Pre_Split* and *RC.no=k* to notify tags in the next *PSG* to split into *k* subgroups by setting counters to be between 0 and k–1, and to notify other tags to increase their counters by k–2 (i.e., k–1– the next *PSG* counter value), if both the following conditions hold.
  - there is no tag response collision
  - the counter value of the next *PSG* is 1.
- Lines 45-48: The reader sends an *RC* command with *RC.status=Pre_Collision* to notify that (1) the tag with counter value 0 has been identified successfully, (2) the tags with

counter value 1 should be split into 2 subgroups, and (3) others tags should increase their counters by 1, if all the following conditions hold.

- there is no tag response collision
- the counter value of the next *PSG* is 2 or more
- the pre-signaling bit is"multiple_1"

■ Lines 49-52: The reader sends an RC command with *RC.status= Success* and *RC.no*=1 to notify that (1) the tag with counter value 0 has been identified successfully, and (2) the other tags should decrease their counters by 1, if all the following conditions hold.

- there is no tag response collision
- the counter value of the next *PSG* is 2 or more
- the pre-signaling bit is "single_1"

■ Lines 53-57: The reader sends an *RC* command with *RC.status= Success* and *RC.no=offset* to notify that (1) the tag with counter value 0 has been identified successfully, and (2) other tags should decrease their counters by *offset*, if all the following conditions hold.

- there is no tag response collision
- the counter value of the next *PSG* is 2 or more
- the pre-signaling bit is "null"

Note that offset=1 if the counter value of the next *PSG* is 2; otherwise, offset=2. This is because tags of the next *PSG* of counter value 2 should decrease their counters by 1 instead of 2; otherwise they will directly be of counter value 0 and bypass the adaptive splitting process.

The tag operations are shown in Algorithm 2. We can observe that there is an action corresponding to each command of the reader. If both the reader and tags comply with the algorithms, all tags can be identified properly.

Algorithm 1. ASPS Reader Operation

```
/* NR stands for New Round command
RC stands for Reader Command, and
RC.status is one of Request, Success,
Collision, Pre_Collision or Pre_Split, and
RC.no indicates the estimated number of
tags for splitting */
1   send NR command to start a new round
2   phase =1          //for starting Phase I
3   max_idx = 0   //the maximum index of PSG groups
4   max_cnt = 0   //the maximum counter value of tags
5   reset PSG[ ].cnt //reset counter values of all PSG groups to be 0
6   send RC with RC.status = Request      //to ask tags to respond
7   while phase = 1 do
8           switch (tag_ response)
9                   case tag_collision:
10                          + + max_idx; ++ max_cnt
11                          + + PSG[i].cnt for i=1..max_idx–1
12                          send RC with RC.status = Collision
13                  case one_tag_response:
14                          record the sole ID responded as an identified one
15                          + + max_idx
16                          PSG[max_idx].cnt = 0      //set the counter (cnt) of the last PSG to be 0
17                          send RC with    RC.status = Success and RC.no=1
18                          phase = 2    //for starting Phase II
19                  case no_tag_response:
20                          – – PSG[i].cnt for i=1..max_idx
21                          – – max_idx; – – max_cnt
22                          send RC with RC.status = Success and RC.no=1
23          end switch (tag_ response)
24  end while (phase = 1)
    // Phase II starts
25  next_idx = max_idx – 1;
    //next_idx stands for the index of the next PSG for processing
26  while phase = 2 do
27          switch (tag_ response)
28                  case tag_collision:
29                          + + PSG[i].cnt for i=1...next_idx
30                          + + max_cnt
```

```
31                  send RC with RC.status = Collision
32              case one_tag_response or no_tag_response:
33                  if (only one tag responds ID)
34                    record ID as an identified one
35                  endif
36                  switch (PSG[next_idx].cnt)
37                        case 1:
38                            estimate the number k of tags in PSG[next_idx]
39                            PSG[i].cnt += (k–2) for i=1..next_idx
40                            max_cnt += (k–2)
41                            – – next_idx
42                            send RC with RC.status = Pre_Split and RC.no =k
43                        default:    // PSG[next_idx] is 2 or more
44                               switch (pre_signal)
45                                     case multiple_1:
46                                         – – PSG[i].cnt for i=1..next_idx
47                                         – – max_cnt
48                                         send RC with RC.status = Pre_Collision
49                                     case single_1:
50                                         – – PSG[i].cnt for i=1..next_idx
51                                         – – max_cnt
52                                         send RC with RC.status=Success and RC.no=1
53                                     case null:
54                                         if (PSG[next_idx]=2) offset=1 else offset=2 endif
55                                           PSG[i].cnt –= offset for i=1..next_idx
56                                           max_cnt –= offset
57                                           send RC with RC.status=Success and RC.no = offset
58                               end switch (pre_signal)
59                  end switch (PSG[next_idx].cnt)
60        end switch (tag_ response)
61        if (max_cnt =0) phase = 0 endif //for terminating Phase II
62   end while (phase = 2)
```

Algorithm 2. ASPS operation for a tag

```
1   receive NR command from the reader to start a new round
2   TC = 0 // initialize tag counter TC for a new round
3   identified=false //set tag to be not identified
4   while (not identified)
5         receive command RC from the reader
6         switch (RC.status)
```

```
            case Request:     //do nothing for TC modification in this case
            case Collision:
                if ($TC = 0$)
                    $TC = r$ ($r$ is a random value of 0 or 1)
                else
                    $TC$ += 1
                end if
            case Pre_Collision:
                switch (TC)
                    case 0: identified = true //keep silient till next round
                    case 1: $TC = r$ ($r$ is a random value of 0 or 1)
                    default: $TC$ += 1
                end switch (TC)
            case Pre_Split:
                switch (TC)
                    case 0: identified=ture;  //keep silent till next round
                    case 1: $TC = r$ ($r$ is a random value of 0, 1,.., or $RC.no - 1$)
                    default: $TC$ += ($RC.no - 2$)
                end switch (TC)
            case Success:
                if ($TC = 0$)
                    identified = true  //keep silent till next round
                else
                    $TC$ –= $RC.no$
                end if
        end switch (RC.status)
        //Respond to the RC command
        if (not identified)
            if ($TC = 1$)
                respond the first bit at the first bit transmission period
            else if ($TC = 0$)
                respond tag ID from the second bit transmission period
            end if
        end if
    end while (not identified)
```

4 SIMULATION AND ANALYSIS

In this section, we show simulation and analysis results. We simulate ASPS and compare it with ISO/IEC 18000-6B protocol [13] in terms of the number of tag collisions, the number of messages sent by the reader and the time needed to identify all tags in the interrogation zone. We also compare the system efficiency of ASPS, query tree, frame slotted ALOHA and ISO/IEC 18000-6B protocols for the cases of 200, 400,…, and 5000 tags in the interrogation zone are considered, and 1000 simulation experiments are performed for each case. It may be unreal to identify thousands of passive tags at a time. But in the scenario of active tag identification, the reader can read up to thousands of tags simultaneously (for example, the model 227004-9-iQRW reader produced by GAO RFID Inc. can read 2000 tags at a time). This accounts for the reason we simulate for the cases of thousands of tags.

4.1 The Number of Collisions

In this subsection, we first compare AS (the protocol with only the adaptive splitting scheme), ASPS (the protocol with both the adaptive splitting and the pre-signaling schemes) and ISO/IEC 18000-6B protocol in terms of the number of collisions by simulation experiments. In Fig. 3, there are three curves of each protocol for the best case result (i.e., the one that has the fewest collisions occurred in the 1000 simulations), the worst case result (i.e., the one that has the most collisions occurred in the 1000 simulations), and the average result (i.e., the average number of collisions occurred in the 1000 simulations). As shown in Fig. 3, ASPS outperforms ISO/IEC 18000-6B protocol. For example, the number of collisions of the former is only 20% of the latter's when the number of tags is 5000. Furthermore, the following observations are worth mentioning.

- The worst case results of ASPS are better than the best case results of ISO/IEC 18000-6B protocol.
- Without using the pre-signaling scheme, AS still has fewer collisions than ISO/IEC 18000-6B protocol.

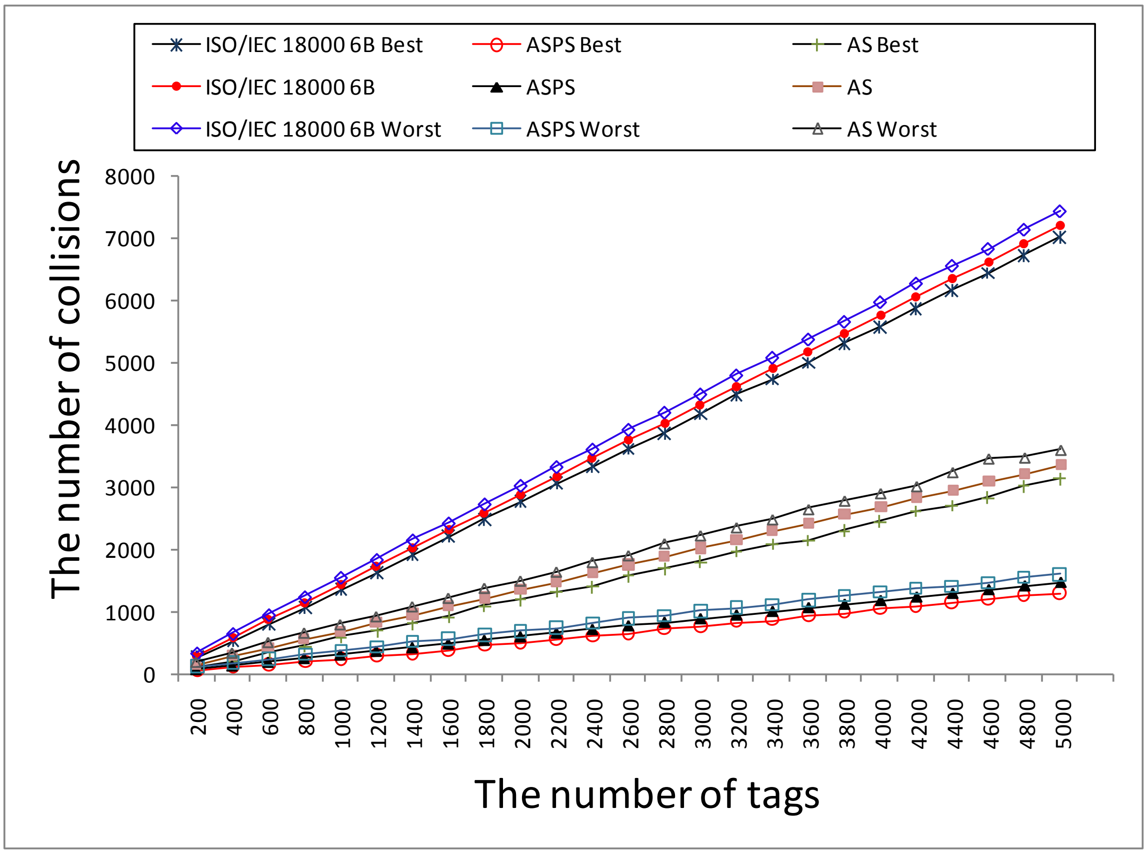

Figure 3. The comparison between ASPS and ISO/IEC 18000-6B protocol in terms of the number of collisions

4.2 The Number of Messages Sent by the Reader

In ISO/IEC 18000-6B protocol, a reader sends a command when it is ready to interrogate tags and listen to the responses from them. The tags with counter value 0 will respond their tag IDs to the reader. The reader will send another command to notify tags of the response status. As we have shown, with the help of the pre-signaling scheme, a single reader command can carry more tag statuses. Therefore, the number of messages sent by the reader is reduced dramatically.

By Fig. 4, we can observe that the number of messages sent by the reader in ASPS is only 55.5% of that in ISO/IEC 18000-6B protocol for 5000 tags. If we disregard essential commands (i.e., we count only the commands caused by tag collisions or no tag responses, but not those for informing successful identification), the number of messages sent by the reader of ASPS is only 31.9% of that in ISO/IEC 18000-6B protocol for 5000 tags. This is because the adaptive splitting and pre-signaling schemes together cut down the number of messages sent by the reader. Similar to Fig. 3, Fig. 4 has there curves for ASPS and ISO/IEC 18000-6B protocol, and the worst case results of the former are better than the best case results of the latter.

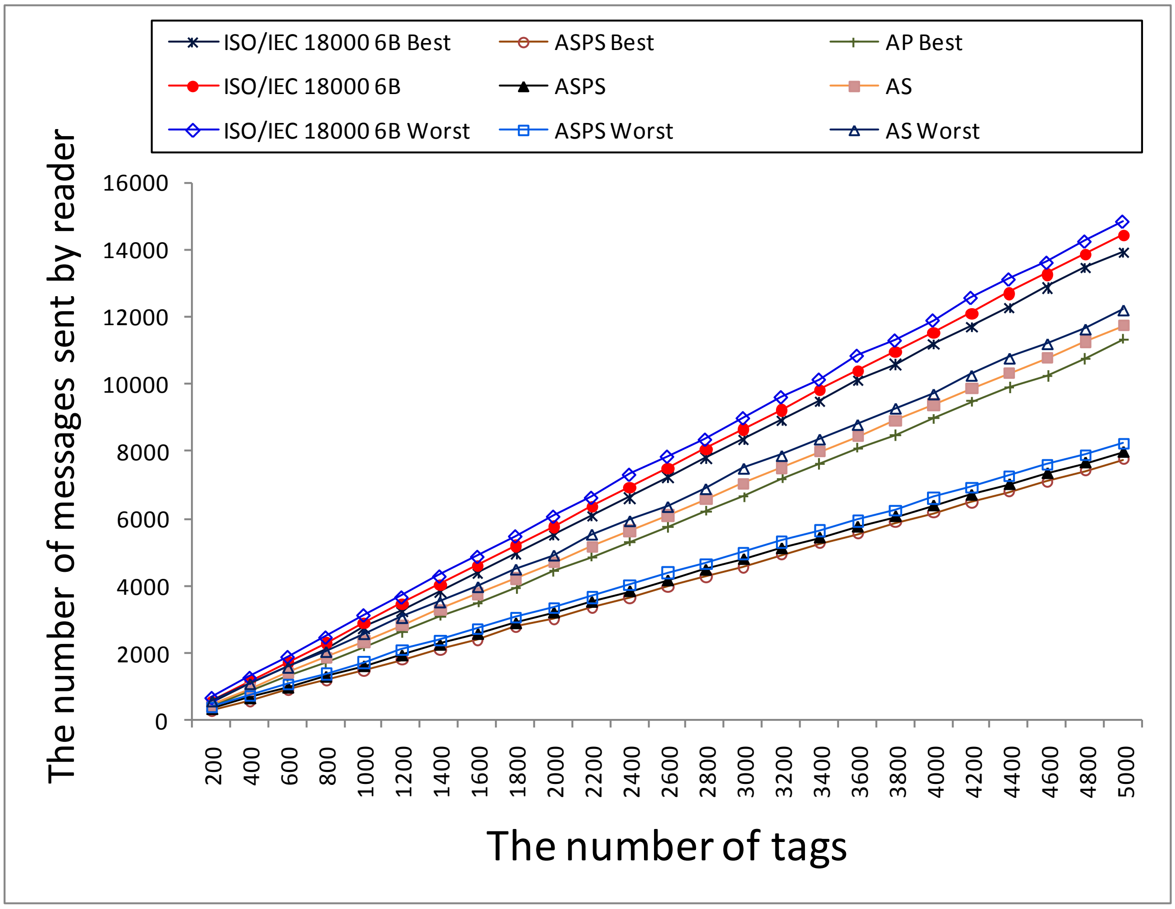

Figure 4. The comparison between ASPS and ISO/IEC18000-6B protocol in terms of the number of messages sent by the reader

4.3 The Tag Identification Delay

We compare ASPS with ISO/IEC 18000-6B protocol in terms of the tag identification delay, which is the elapsed time for a reader to identify all tags in the interrogation zone. The delay time T_d is de fined to be:

$$T_d = (N_C + N_{NC}) \times T_{ID} + N_{CMD} \times T_M + N_{NULL} \times T_W \qquad (2)$$

The notations used in Eq. 2 are explained in Table I. And we follow the parameters suggested by ISO/IEC 18000-6B protocol [13], shown in Table II, to calculate *Td*. By Table II, we set the time for a reader to transmit a command to be 1.5 ms, set the time for a tag to transmit tag ID to be 2.7 ms, and set the elapsed time for a reader to be aware of the null response to be 0.5 ms. Table III demonstrates the delay time to identify all tags in ASPS and ISO/IEC 18000-6B protocol for different numbers of tags. By Table III, we can observe that the delay time of ASPS is only about 60% of that of ISO/IEC 18000-6B protocol.

Table I. The notations used in Eq. 2

N_C	The number of cases with tag collisions
N_{NC}	The number of cases with no tag collisions
N_{CMD}	The number of commands sent by the reader
N_{NULL}	The number of cases of no tag response (null response)
T_{ID}	The elapsed time for a tag to transmit its ID
T_M	The elapsed time for the reader to transmit a command
T_W	The elapsed time for a reader to be aware of null response

Table II. The parameters adopted by ISO/IEC 18000-6B protocol [13]

	Reader forward transmission	Tag backward transmission
Data rate	40 kbps*	40 kbps
Tag identifier	N/A	64 bits
Error detection	16 bit CRC	16 bit CRC
Preamble	25 bits	16 bits
Delimiter	11 bits	N/A
Quiet	N/A	15 bits**
Command set	8 bits	N/A
Tag receiving-to-transmitting turn around time	N/A	ranging from 85 to 460 μs

* 10 or 40 kbps according to local regulations (we assume 40kbps)

** Tags shall not respond for $16 \times T_B - 0.75 \times T_F$, where T_B (resp., T_F) is one symbol period for backward (resp., forward) transmission, which depends on the transmission data rate. We assume T_B and T_F are both 25 μs.

Table III. The tag identification delay comparison

Protocols \ The number of tags	100	200	300	400	500
ISO/IEC 18000-6B	1108.6	2221.4	3340.4	4454.7	5559.8
ASPS	717.7	1390.1	2066.7	2722.9	3391.1

* The delay is measured by ms

4.4 The Error on the Estimated Number of Tags

In this subsection, we analyze the error on the estimated number of tags in a *PSG* group for the adaptive splitting scheme. As described in Section 3.1, a reader in ASPS estimates the number of the tags in each *PSG* and splits the tags according to the estimated number during the identification process. The estimated number k_e of tags in PSG_e is the summation of the number of tags in the *PSG* groups indexed by e+1, e+2, ..., *max_idx*, where the actual number of tags in PSG_{e+1}, PSG_{e+2}, …, PSG_{max_idx} are known before tags in PSG_e are to be identified. That is, the estimated number k_e of tags in PSG_e is N_e, where $N_e = \sum_{i=e+1}^{max\_idx} |PSG_i|$, as defined in Eq. (1). Let V_e be the value of $N_e+|PSG_e|$ (i.e., the actual number of tags in PSG_e, PSG_{e+1},…, PSG_{max_idx}). In the case of $N_e=x$, we have $k_e = x$, $|PSG_e| = V_e–x$, and the estimation error between k_e and $|PSG_e|$ is $\| (V_e–x)–x \|$. We assume that when the tag-splitting procedure splits tags in PSG_e into two groups, a tag will be in either of the two split groups with equal probability. The expected estimation error ERR_e between k_e and the actual number of tags in PSG_e is thus given by

$$ERR_{\mathrm{e}} = \|k_e - PSG_e\| = \sum_{x=1}^{V_e} \|V_e - 2x\| \, \mathrm{C}_x^{V_e} \left(\frac{1}{2}\right)^x \left(1-\frac{1}{2}\right)^{V_e - x} \qquad (3)$$

In Table IV, we show the expected estimation error ERR_e and the estimation error ratio (i.e., ERR_e / V_e) when the actual number of tags is 200, 600, …, 5000. We also show in

Table IV the simulation results of estimation errors, which are derived by averaging outcomes of 1000 experiments for each case. By Table IV, we can see that the simulation and the analysis results are very close and that an estimation error of the number of tags in a *PSG* group is existent, but not too large. It is good enough for tags to be split into subgroups of 0, 1 or few tags, in which ISO/IEC18000-6B protocol is applied to identify the tags. As shown in Fig. 3, ASPS indeed reduces the collisions effectively.

Table IV. The estimation errors (ERR_e) and their ratios for different number (V_e) of tags in PSG_e

V_e	200	600	1000	1400	1800	2200	2600
ERR_e (simulation)	11	20	25	30	33	36	41
ERR_e (analysis)	11	20	25	30	34	37	41
Estimation error ratio (%) (analysis)	5.5	3.3	2.5	2.1	1.9	1.7	1.6
V_e	3000	3400	3800	4200	4600	5000	
ERR_e (simulation)	44	46	48	50	54	55	
ERR_e (analysis)	44	47	49	52	54	56	
Estimation error ratio (%) (analysis)	1.5	1.4	1.3	1.2	1.2	1.1	

4.5 The Effect of Pre-signaling Scheme

The pre-signaling (PS) scheme helps reduce the traffic messages between the reader and tags at the expense of sending one more bit from tags to the reader. In order to evaluate the effect of the PS scheme, we run simulations of AS (without PS) and ASPS (i.e., AS along with PS) for the cases of 100, 200, 300, 400 and 500 tags in the interrogation zone. The number of the commands sent, the number of cases of multiple responses, the number of cases of successful (unique) responses and the number of cases of no response are shown in Table V. We can see that with the help of the PS scheme, the number of the cases of multiple responses and the number of cases of no response can be reduced by half. The commands needed to handle the above cases can be saved, and therefore the number of commands sent is reduced significantly.

Assuming the tag ID is 96 bits long and the command sent by the reader is 8 bits long, we show the numbers of the extra bits used and the bits saved by the PS scheme in Table VI. By Table VI, we can find that the number of extra bits is much smaller than the number of saved bits, which means the PS scheme indeed can improve performance.

Table V. The numbers of different cases and commands sent in the AS and the ASPS schemes

Schemes / Cases / The number of tags	AS (without PS)				ASPS (with PS)			
	Multiple response	Unique response	No response	Commands Sent	Multiple response	Unique response	No response	Commands Sent
100	73	100	68	242	36	100	32	169
200	142	200	136	479	67	200	63	330
300	210	300	204	714	96	300	93	490
400	278	400	271	950	125	400	124	650
500	343	500	337	1,180	155	500	155	811

Table VI. The comparison of the numbers of the extra bits used and the bits saved by the PS scheme

The number of tags	100	200	300	400	500
The number of extra bits	168	330	489	649	810
The number of saved bits	7,592	15,400	23,392	31,200	38,472

4.6 System Efficiency

We compare ASPS, query tree (QT) [9], ISO/IEC 18000-6B [13], and frame slotted ALOHA [6] protocols for 100, 200,…, and 1000 tags in terms of *system efficiency*. In [11], system efficiency is defined as the ratio of the number n of tags to the number s of slots required to identify all the tags for ALOHA-based protocols. To compare different classes of protocols, the number s of slots is assumed to be the number of iterations for some protocols, where an *iteration* is for a reader to send a command and for tags to perform corresponding actions. For example, for counter-based protocols, an *iteration* is for a reader to send a command and for tags to increase/decrease counters or to respond tag IDs. For QT protocol, an *iteration* is for a reader to send a command with some ID prefix and for tags to perform ID prefix matching and ID responding. We perform simulations under the assumption that a frame has *s* time slots initially for frame slotted ALOHA protocol and that tag IDs are uniformly distributed. Fig. 5 shows the simulation results. By Fig. 5, we can see that ASPS apparently outperforms the others in terms of system efficiency.

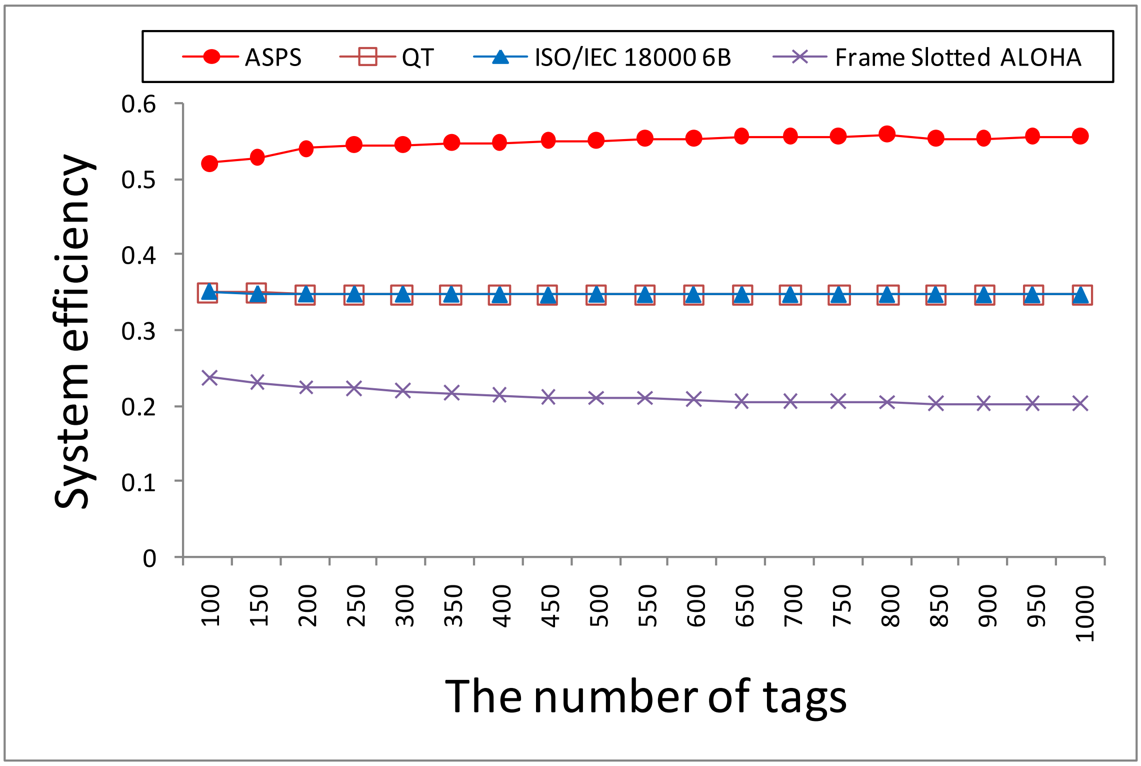

Figure 5. The comparison of ASPS, query tree (QT), ISO/IEC 18000-6B, and frame slotted ALOHA protocols in terms of system efficiency

5 Conclusion

In this paper, we have proposed a novel counter-based tag anti-collision protocol, called ASPS, based on two schemes: adaptive splitting and pre-signaling. By estimating the number *k* of colliding tags, the adaptive splitting scheme tries to directly distribute the tags into *k* subgroups. This will reduce the number of tag collisions effectively. And the pre-signaling scheme is applied to further reduce the number of messages sent between the reader and tags. We have performed simulations for ASPS and compared it with other anti-collision protocols. Simulation results show that ASPS outperforms related protocols in terms of the number of collisions, the number of messages sent by the reader, the tag identification delay, and system efficiency. However, ASPS needs to modify the random number generator design on the tag circuit. Furthermore, the ASPS reader also needs to process a more complex algorithm, as described in Algorithm 1. These are negative points of ASPS.

Both ASPS and ABS protocol [12] are proposed to improve ISO/IEC 18000-6B protocol, ASPS makes improvement within a tag interrogation round; ABS, between interrogation rounds. Therefore, ASPS and ABS can be combined together to further improve ISO/IEC 18000-6B protocol. We are now planning to integrate the two protocols and perform some preliminary simulation experiments to show the integration advantages.

Acknowledgment

This work was supported in part by the National Science Council of the Republic of China (Taiwan) under the grant NSC-96-2221-E-008-007.

References

[1] K. Finkenzeller, *RFID handbook: Fundamentals and Applications in Contactless Smart Cards and Identi- fication,* John Wiley & Sons, 2003.

[2] Jehn-Ruey Jiang and Ming-Kuei Yeh, "Anti-collision protocols for the RFID system," Book Chapter of *RFID and Sensor Networks*, Ed. Yan Zhang et al., Auerbach Publications, Taylor&Francis Group, USA, 2009.

[3] N. Abramson, "The ALOHA System-Another Alternative for Computer Communications," *Proc. of AFIPS Spring Joint Computer Conf.*, Vol. 37, pp. 281-285, 1970.

[4] Leian Liu, Shengli Lai, "ALOHA-Based Anti-Collision Algorithms Used in RFID System," *Proc. of In- ternational Conf. on Wireless Communications, Networking and Mobile Computing (WiCOM 2006),* pp.1 – 4, Sep. 2006.

[5] H. Vogt, "Efficient Object Identification with Passive RFID Tags," *Proc. of 1st*

International Conf. on Pervasive Computing, pp.98–113, 2002.

[6] M. Kodialam and Thyaga Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," *Proc. of ACM Mobicom*, Sep. 2006.

[7] S. Lee, S.D. Joo, and C.W. Lee, "An enhanced dynamic framed slotted aloha algorithm for RFID tag identification, " *Proc. of Mobiquitous 2005*, pp.166-172, 2005.

[8] Feng Zhou et al., "Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems," *Proc. of the 2004 international symposium on Low power electronics and design*, 2004.

[9] H. Choi, J. R. Cha and J. H. Kim, "Fast wireless anti-collision algorithm in ubiquitous ID system," *Proc. of IEEE VTC '04*, 2004.

[10] M. A. Bonuccelli, F. Lonetti, F. Martelli. "Tree Slotted Aloha: a New Protocol for Tag Identification in RFID Networks," *Proc. of the 4th IEEE International Workshop on Mobile Distributed Computing (MDC'06)*, 2006.

[11] Ji Hwan Choi, Dongwook Lee, and Hyuckjae Lee, "Bi-slotted tree based anti-collision protocols for fast tag identification in RFID systems, " *IEEE Communications Letters*, Vol. 10, Issue 12, pp. 861-863, 2006.

[12] J. Myung et al., "Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 6, Jun. 2007.

[13] ISO, *Information Technology Automatic Identification and Data Capture Techniques – Radio Frequency Identification for Item Management Air Interface - Part 6: Parameters for Air Interface Communications at 860-960 MHz*, International Standard ISO 18000-6, Nov. 2003.

[14] Philips Semiconductors, *UCODE*, http://www.semiconductors.philips.com.

[15] L. G. Roberts, "Extensions of Packet Communication Technology to a Hand Held Personal Terminal," *Proc. of AFIPS Spring Joint Computer Conf., vol. 40, pp.* 295-298, 1972.

[16] C. Qian, H. Ngan, and Y. Liu, "Cardinality Estimation for Large-scale RFID Systems," *Proc. of IEEE Int'l Conf. on Perv. Comp. and Comm. (PerCom)*, pp. 30–39, Mar. 2008.

[17] Jae-Ryong Cha and Jae-Hyun Kim, "Novel Anti-collision Algorithms for Fast Object Identification in RFID System," *Proc. of 11th International Conf. Parallel and Distributed Systems (ICPADS'05)*, pp. 63- 67, Jul. 2005.

[18] A. Micic et al., "A hybrid randomized protocol for RFID tag identification," *Proc. of First IEEE International Workshop on Next Generation Wireless Networks (WoNGeN '05)*, Dec. 2005