# An RFID Tag Anti-Collision Protocol Using Parallel Splitting and Merging

Ming-Kuei Yeh
Department of Information Management
National Taipei College of Business
jamesyeh@webmail.ntcb.edu.tw

Shing-Tsaan Huang
Department of Computer Science and Information Engineering
National Central University
sthuang@csie.ncu.edu.tw

Jehn-Ruey Jiang
Department of Computer Science and Information Engineering
National Central University
jrjiang@csie.ncu.edu.tw

## Abstract

*In the RFID system, tag signal collisions occur when multiple tags respond their IDs to the reader simultaneously. In such case, no tag can be identified by the reader successful and the performance of tag identification degraded. How to reduce tag collisions to speed up the identification is thus important. There are several anti-collision protocols proposed for reducing tag collisions. They can be categorized into two classes: ALOHA-based protocols and tree-based protocols; the latter can be further classified into deterministic tree-based and probabilistic counter-based subclasses of protocols. In this paper, we propose a probabilistic counter-based anti-collision protocol, called PSM, to simply split and merge tag groups in parallel to reduce tag collisions for shortening the identification delay. We also conduct simulation experiments for the proposed protocol and compare it with related ones, such as the QT, FS-ALOHA and ISO18000-6B protocols in terms of the number of iterations and system efficiency. To the best of our knowledge, the PSM protocol has the highest system efficiency among the plain protocols that use no special techniques, such as bit-tracking, tag population estimation, and re-identification.*

## 1. Introduction

Recently, the *RFID (Radio Frequency IDentification)* technique [1] has been applied to many applications due to its contactless, automatic identification capability. The front end of an RFID system is composed of readers and tags. When a tag and a reader can communicate with each other, we say that the tag is in the *interrogation zone* of the reader. Because the reader does not know which tags are in its interrogation zone, it initiates an *interrogation procedure* (or *identification procedure*) to request tags to send back their IDs. In cases where multiple tags respond to the reader's request simultaneously, *tag collisions* occur and the reader cannot successful identify any tags. How to reduce tag collisions and speed up the identification procedure is thus important. There are several *anti-collision* protocols proposed for reducing tag collisions. They can be categorized into two classes [2]: *ALOHA-based* and *tree-based* protocols; the latter can be further classified into *deterministic tree-based* and *probabilistic counter-based* subclasses of protocols.

In ALOHA-based protocols, a tag responds to the reader's request by transmitting its ID in an arbitrarily selected time slot. For example, in the frame slotted ALOHA (FS-ALOHA) protocol [3], the period of an interrogation procedure is divided into several frames, each of which is composed of the same number of time slots. On receiving the reader's request command, a tag randomly chooses a time slot to transmit its ID to the reader. If only one tag responds in a slot, it can be identified successfully. A tag not identified successfully will re-select a time slot in the next frame to retransmit its ID. When no tag responds in a frame, all tags are identified successfully and the identification procedure stops. One problem with the protocol is that its performance degrades when the number of slots does not properly match the number of tags. Dynamic frame slotted ALOHA protocols solve this problem by dynamically adjusting the frame size. However, they need to accurately estimate the number of tags, which is not an easy task.

The basic idea of the tree-based [2, 4-9] protocols is to repeatedly split tags that encounter collisions into subgroups until there is only one tag in each subgroup to be identified successfully. The major difference between the deterministic tree-based and probabilistic counter-based protocols is that the former splits the colliding tags according to their static IDs, and the latter, according to dynamically changing counters.

In this paper, we propose a probabilistic counter-based anti-collision protocol, called PSM (parallel splitting and merging), to simply split and merge tag groups in parallel to reduce tag collisions for shortening the identification delay. We also conduct simulation experiments for the proposed protocol and compare it with related ones, such as the QT, FS-ALOHA and ISO18000-6B protocols in terms of the number of iterations and system efficiency. To the best of our knowledge, the PSM protocol has the highest system efficiency among the *plain protocols* that use no special techniques, such as *bit-tracking*, which needs accurate and flexible bit collision detection, and *tag population estimation*, which incurs complex computation, and *re-identification*, which is just suitable for some cases where the reader needs to repeatedly identify similar sets of tags.

The rest of the paper is organized as follows. We describe some representative plain tree based protocols in Section 2.The proposed protocol is elaborated in Section 3, and its performance is evaluated by simulation and is compared with those of related ones in Section 4. Finally, Section 5 concludes the paper.

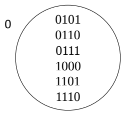
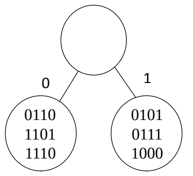
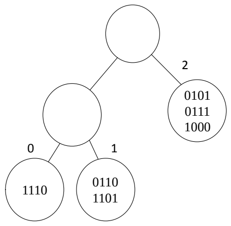
## 2. Related Works

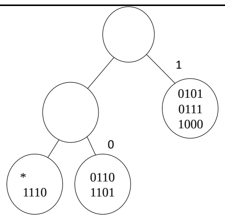
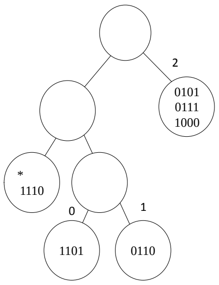
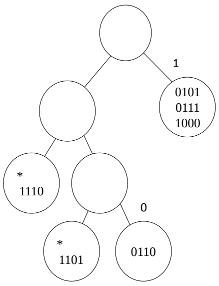
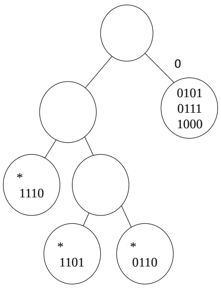
The Query Tree (QT) protocol is a well-known deterministic tree-based protocol. In this protocol, the reader first broadcasts a request string (or an ID prefix) S to tags, and a tag whose ID prefix matches S will send back its remainder ID to the reader. If only one tag responds, the tag is identified successfully. But if multiple tags respond simultaneously, their responses collide to prevent them from being identified successfully. Then, the reader generates two new prefixes by appending 0 and 1, respectively, to S and broadcasts them in order. In this way, the colliding tags are divided into two subgroups ready to respond. The splitting procedure repeats until no tag responds. The reader initially broadcasts two prefixes: 0 and 1, and broadcasts all prefixes generated so that all tags can be identified successfully. The length and the distribution of tag IDs affect the QT protocol's identification delay. For example, if tag IDs are contiguous, the request string grows longer and longer, and the delay then increases significantly.

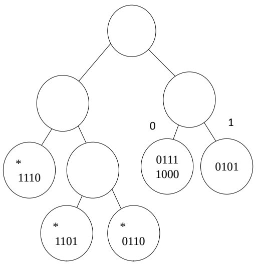
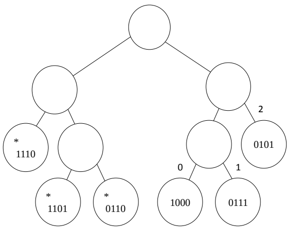
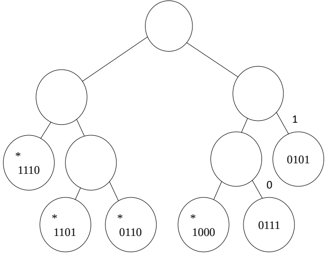
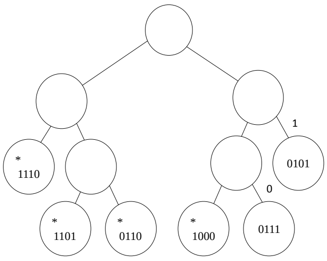
The anti-collision protocol of the ISO/IEC 18000-6B standard (later named the ISO18000-6B protocol for short) [7] is a famous probabilistic counter-based protocol. In this protocol, each tag maintains a counter initially set to 0. Only a tag with a counter value of 0 can return its ID to the interrogation request. When a collision occurs, the reader will notify all of the tags about this. And the unidentified tags with counter values larger than 0 will increase their counters by 1, while the colliding tags (i.e., the unidentified tags with a counter value of 0) will add 0 or 1 randomly to their counters. By this rule, the colliding tags are split into two subgroups. The splitting will continue until no or one response occurs. In the one-response case, the responding tag can be identified successfully. And, in both cases, the reader sends a command to inform all unidentified tags to decrease their counters by 1. The reader keeps track of the largest counter value. When this value reaches 0, all tags are identified successfully and the identification procedure stops.

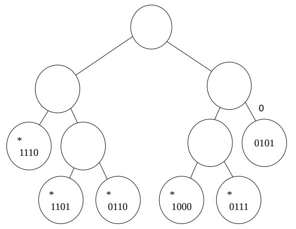
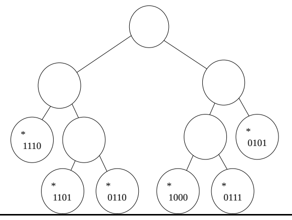
A simple example is given here, in order to observe the identification procedure of the ISO18000-6B anti-collision protocol. We suppose there are 6 tags with tag ID as 0101, 0110, 0111, 1000, 1101 and 1110, in the interrogation zone of reader. The procedure and identification tree are shown in Table 1.

Table 1. The identification procedure of the ISO18000-6B protocol and its identification tree (the tag is identified if tag ID is marked with *)

| Iteration | Reader command | No of tag | Counter value | Choose 0 or 1 randomly | New counter value | Tag ID responded |
|---|---|---|---|---|---|---|
| 1 | initial request | 1 | -- |  | 0 | 0101 |
|  |  | 2 | -- |  | 0 | 0110 |
|  |  | 3 | -- |  | 0 | 0111 |
|  |  | 4 | -- |  | 0 | 1000 |
|  |  | 5 | -- |  | 0 | 1101 |
|  |  | 6 | -- |  | 0 | 1110 |
|  |  |  |  |  |  |  |
| 2 | collision | 1 | 0 | 1 | 1 |  |
|  |  | 2 | 0 | 0 | 0 | 0110 |
|  |  | 3 | 0 | 1 | 1 |  |
|  |  | 4 | 0 | 1 | 1 |  |
|  |  | 5 | 0 | 0 | 0 | 1101 |
|  |  | 6 | 0 | 0 | 0 | 1110 |
|  |  |  |  |  |  |  |
| 3 | successful identification | 1 | 1 | -- | 2 |  |
|  |  | 2 | 0 | 1 | 1 |  |
|  |  | 3 | 1 | -- | 2 |  |
|  |  | 4 | 1 | -- | 2 |  |
|  |  | 5 | 0 | 1 | 1 |  |
|  |  | 6 | 0 | 0 | 0 | 1110 |
|  |  |  |  |  |  |  |
| 4 | collision | 1 | 2 | -- | 1 |  |
|  |  | 2 | 1 | -- | 0 | 0110 |
|  |  | 3 | 2 | -- | 1 |  |
|  |  | 4 | 2 | -- | 1 |  |
|  |  | 5 | 1 | -- | 0 | 1101 |
|  |  | 6 | 0 | -- | -- |  |
|  |  |  |  |  |  |  |

| | | | | | | |
|---|---|---|---|---|---|---|
| | |  | | | | |
| 5 | successful identification | 1 | 1 | | 2 | |
| | | 2 | 0 | 1 | 1 | |
| | | 3 | 1 | | 2 | |
| | | 4 | 1 | | 2 | |
| | | 5 | 0 | 0 | 0 | 1101 |
| | | 6 | -- | -- | -- | |
| | |   2<br>0101<br>0111<br>1000<br>*<br>1110<br>0<br>1<br>1101<br>0110 | | | | |
| 6 | successful identification | 0 | 2 | | 1 | |
| | | 1 | 1 | | 0 | 0110 |
| | | 0 | 2 | | 1 | |
| | | 4 | 2 | | 1 | |
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |   1<br>0101<br>0111<br>1000<br>*<br>1110<br>0<br>*<br>1101<br>0110 | | | | |
| 7 | | 1 | 1 | | 0 | 0101 |
| | | 2 | 0 | -- | -- | |
| | | 3 | 1 | | 0 | 0111 |
| | | 4 | 1 | | 0 | 1000 |
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |   0<br>0101<br>0111<br>1000<br>*<br>1110<br>*<br>1101<br>*<br>0110 | | | | |
| 8 | collision | 1 | 0 | 1 | 1 | |
| | | 2 | -- | -- | -- | |
| | | 3 | 0 | 0 | 0 | 0111 |
| | | 4 | 0 | 0 | 0 | 1000 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |   0<br>1<br>*<br>1110<br>0111<br>1000<br>0101<br>*<br>1101<br>*<br>0110 | | | | |
| 9 | successful identification | 1 | 1 | -- | 2 | |
| | | 2 | -- | -- | -- | |
| | | 3 | 0 | 1 | 1 | |
| | | 4 | 0 | 0 | 0 | 1000 |
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |   *<br>1110<br>2<br>0101<br>0<br>1<br>*<br>1101<br>*<br>0110<br>1000<br>0111 | | | | |
| 10 | successful identification | 1 | 2 | | 1 | |
| | | 2 | -- | -- | -- | |
| | | 3 | 1 | | 0 | 0111 |
| | | 4 | -- | -- | -- | |
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |   1<br>*<br>1110<br>0101<br>0<br>*<br>1101<br>*<br>0110<br>*<br>1000<br>0111 | | | | |
| 10 | successful identification | 1 | 2 | | 1 | |
| | | 2 | -- | -- | -- | |
| | | 3 | 1 | | 0 | 0111 |
| | | 4 | -- | -- | -- | |
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |   1<br>*<br>1110<br>0101<br>0<br>*<br>1101<br>*<br>0110<br>*<br>1000<br>0111 | | | | |
| 11 | successful identification | 1 | 1 | | 0 | 0101 |
| | | 2 | -- | -- | -- | |
| | | 3 | -- | -- | -- | |
| | | 4 | -- | -- | -- | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |   0<br>*<br>1110<br>*<br>1101<br>*<br>0110<br>*<br>1000<br>*<br>0111<br>0101 | | | | |
| 12 | finish | 1 | 0 | -- | -- | |
| | | 2 | -- | -- | -- | |
| | | 3 | -- | -- | -- | |
| | | 4 | -- | -- | -- | |
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |   *<br>1110<br>*<br>0101<br>*<br>1101<br>*<br>0110<br>*<br>1000<br>*<br>0111 | | | | |

Some researches proposed mechanisms, such as Schoute's method [10], Vogt's method [3], Floerkemeier's protocol [11, 12], Popovski's algorithm [13], the Q algorithm [1], Lai's protocol [14], the ASAP protocol [8], and the PS protocol [15], to improve tag identification performance by using *tag population estimation* [16]. For example, the PS (*Parallel Splitting*) protocol [15] tries to improve the ISO18000-6B protocol by reducing the number of iterations with two schemes: the *parallel splitting* and the *adaptive identification-tree height adjustment*, where an *iteration* is for a reader to send a command and for tags to perform corresponding actions, and an *identification tree* is the tree corresponding to an intermediate state of the identification procedure. The first scheme instructs all remaining unidentified tags to left-shift one bit on their counters and then to add 0 or 1 randomly to the counters when collisions occurred. In this way, all unidentified tags are split into subgroups in parallel to speed up tag splitting. After the first tag is identified, the tags are then identified one by one according to the normal identification procedure of the ISO18000-6B protocol. The second scheme tries to fine-tune the effect of parallel splitting by adaptively adjusting the identification tree height to approach a condition where each leaf node contains one tag, keeping as small as possible the number of iterations needed to identify all tags. In the second scheme, the reader keeps track of variables $N_0$, $N_1$, and $N_\mathrm{m}$ to adjust the identification tree height by following rules R1 and R2, where $N_0$, $N_1$ and $N_\mathrm{m}$ are respectively the accumulated numbers of leaf nodes with zero, one and multiple (i.e., two or more) tags during the interrogation procedure.

R1: **If** ($N_0 > 2 \times N_1$ and $(7/24) \times N_1 > N_m$) **Then**
Command all unidentified tags to right-shift their counters one bit.

R2: **If** ($2 \times N_0 < N_1$ and $(5/3) \times N_1 < N_m$) **Then**
Command all unidentified tags to left-shift their counters one bit, and subsequently add zero or one randomly to the counters.

Tag population estimation usually involves complex computation and extra memory, causing some overheads. Furthermore, it is hard to estimate the tag population accurately in some cases (e.g., when the identification procedure just starts up), which makes the identification performance unstable.

The FQT (Fast Query Tree) protocol [17] is proposed to improve the identification performance by reducing the data transmitted between reader and tag. FQT is a variant of QT, while in some manner, it also behaviors like counter based protocol. At the beginning of identification procedure, tag sets its counter SC and pointer PT as 0, and reader sets the length of position PO for request bit as 1. During the identification procedure, the reader firstly broadcasts PO and request bit to the tags in the interrogation zone and tags set PO-1 as PT. Only the tag with SC=0 and the bit of tag ID that pointed by PT is equal to the reader's request bit can respond the reminder tag ID to the reader, while other unidentified tags will increase their SC by 1. When the reader receives the response from tags, it can be no tag, only one tag and multiple tags responses. In the first two cases, except the only one tag is identified successfully, while other unidentified tags decrease their SC by one. In the last case, collision occurred and no tag can be identified successfully. Because the transmitted data is encoded in Manchester code, the reader can find out the collision bits after receiving the post response tag IDs. Therefore, the reader can detect the first collision bit and record the position of this bit as PO for next identification request command.

The FQT protocol can improve the identification performance significantly. But it is hard to detect the collision at each separate bit correctly, because timing synchronization among tiny tags is very challenging [18].

ABS (Adaptive Binary Splitting) protocol [19] is proposed to improve ISO/IEC 18000 6B anti-collision protocol. In ABS protocol, a tag maintains two counters, Progressed Slot Counter (PSC) and Allocated Slot Counter (ASC). With PSC and ASC, a tag can decide if it can respond its ID to a reader request.

PSC is initialized to 0 and increased by 1 when a tag is successfully identified, and it represents the number of identified tags. Tags with ASC equal to PSC can transmit their tag IDs. When no tag responds, all tags with ASC larger than PSC decrease ASC by one. When tag collisions occurred, the reader notifies the collision result to all tags. In such collision case, the tags with ASC larger than PSC increase their ASC by 1, while the tags with ASC equal to PSC randomly add 0 or 1 to their ASC. Otherwise, tags with ASC less than PSC do not increase their ASC because they have already been identified and do not transmit their IDs until the tag interrogation round finishes. After all tags are identified, tags in the interrogation zone have unique and successive ASC values. These values of ASC can be kept for use in the next tag interrogation round to speed up the identification procedure. If there are tags joining or leaving after the last interrogation round, the following actions are taken to adjust the unique and successive ASC values.

- Tags joining:
  When a new coming tag receives the reader's initial command to start a new interrogation round, it sets its PSC to 0 and sets its ASC to a random value between 0 and R which provided by the reader. The new tag's response will collide with that of the old tag with same ASC value R. The processes of ABS protocol mentioned above in the first interrogation round can deal with the collision properly by adjust all tags' ASC counters.
- Tags leaving:
  If no tag responds to a reader request, the reader knows that a tag was left. All tags with ASC larger than PSC will decrease ASC by one to deal with the case.

As shown in [19], the performance of ISO18000-6B tag anti-collision protocol is improved significantly by the ABS protocol. But when the tag population changes greatly in consecutive interrogation rounds, the identification performance of ABS is reduced dramatically.

## 3. Proposed Protocol

The design of RFID tag anti-collision protocol should be as simple as possible, since RFID tag is a tiny device with limited resources. In this paper, we proposes a plain probabilistic counter-based anti-collision protocol, called *parallel splitting and merging (PSM)*, to reduce the tag identification delay without using special techniques, such as bit-tracking (used by FQT protocol, for example), tag population estimation (used by PS, for example) and re-identification (used by ABS, for example).

The basic concept of the proposed PSM protocol is to split and merge in parallel the groups of all unidentified tags. To split tags is through left-shifting tag counters by *u* bits and adding 0, 1, …, or $2^u$–1 randomly to the counters, while to merge tags is through right-shifting tag counters by *v* bits, where *u* and *v* are pre-specified system parameters and may not be identical.

The commands sent from the reader to tags are: *start*, *minus-one*, *split*, *and merge*. After receiving and processing a command, the tags with the counter value 0 will respond their IDs to the reader. The reader sends out "start" command to start a new round of the interrogation procedure and to inform all tags to reset their counters to 0. When multiple tags respond simultaneously, collisions occur and the reader sends out "split" command to make all unidentified tags split. When no tag responds, the reader make unidentified tags merge by sending out "merge" command. When only one tag responds, the reader sends out "minus-one" command to make all tags decrease their counters by 1. In this case, the responding tag can be identified successfully; on receiving the "minus-one" command, its counter will be decreased to be -1, and it will go to *sleep* and keep silent until the next new round.

The intuition of the PSM protocol is as follows. When the first tag is identified, we may expect each leaf node of the identification tree to contain nearly one tag under the condition that adding 0, 1, …, or $2^u$–1 to the counters is truly random. However, a leaf node may contain 0 or more tags in practice. When no tag responds at an iteration, we may assume that the leaf nodes outnumber tags, which in turns implies the identification tree is too high. The reader therefore commands tags to merge themselves. Similarly, when multiple tags respond at an iteration, the reader should command tags to split themselves to increase the tree height and the number of leaf nodes. Figures 1(a) and (b) show the pseudo-code of the reader and tag actions in PSM, respectively. The reader maintains an integer variable N to record the iterations needed to finish the interrogation procedure, while a tag maintains an integer variable C as a counter.

```
Pseudo code for the reader
// u and v are system parameters
Set N=1
Send "start" to tags
While N > 0
    Read IDs from tags
    Switch   (number of responses)
      Case 0: //no response
          If N==1 Then Set N=0
          Else
              Send "merge" to tags
              Set N=N / $2^v$
      Case 1: //one response
          Send "minus-one" to tags
          Set N=N-1
      Case $2^+$: //2 or more responses
          Send "split" to tags
          Set   N=N * $2^u$
```

(a)

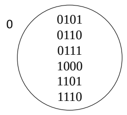
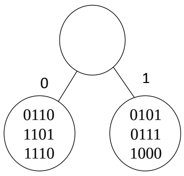
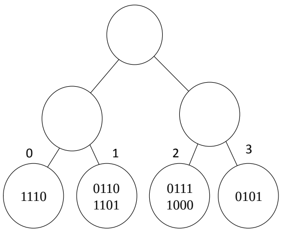
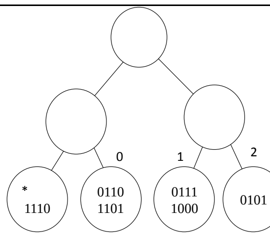
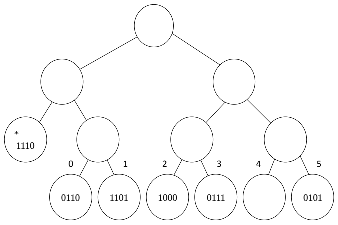
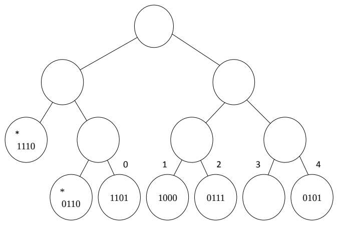
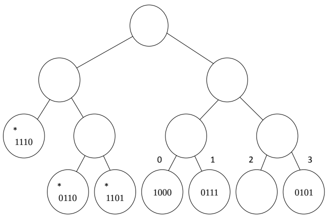
```
Pseudo code for the tag
// u and v are system parameters
While true
    Receive a command from the reader
    Switch (type of the command)
      Case: "start":
          Set C= 0
      Case: "minus-one":
          Set C=C-1
          If C== -1 Then Sleep
      Case: "split":
          Left-shift C by u bits
          Add 0,1,..,or $2^u$-1 to C randomly
      Case: "merge":
          Right-shift C by v bits
    If C== 0 Then Respond Tag ID
```

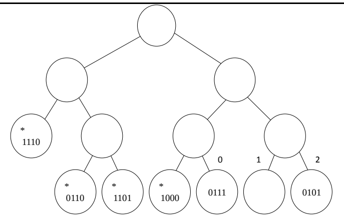
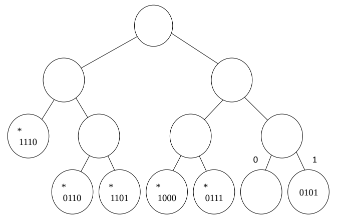
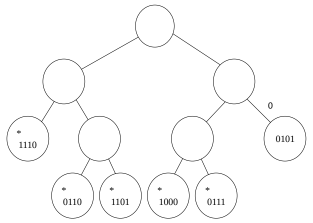
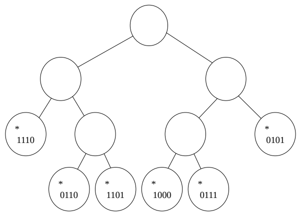
(b)

Figure 1. Pseudo code of the PSM protocol for (a) the reader and (b) the tag

In order to observe the identification procedure of the PSM anti-collision protocol, the example for the ISO18000-6B protocol in section 2 is given again. The procedure and identification tree are shown in Table 2.

Table 2. The identification procedure of the PSM protocol and its identification tree (if a tag is identified successfully, its tag ID is marked with *)

| Iteration | Reader command | No of tag | Counter Value | Choose 0 or 1 randomly | New counter value | Tag ID responded |
|---|---|---|---|---|---|---|
| 1 | initial request | 1 | -- | | 0 | 0101 |
| | | 2 | -- | | 0 | 0110 |
| | | 3 | -- | | 0 | 0111 |
| | | 4 | -- | | 0 | 1000 |
| | | 5 | -- | | 0 | 1101 |
| | | 6 | -- | | 0 | 1110 |
| | |  | | | | |
| 2 | collision | 1 | 0 | 1 | 1 | |
| | | 2 | 0 | 0 | 0 | 0110 |
| | | 3 | 0 | 1 | 1 | |
| | | 4 | 0 | 1 | 1 | |
| | | 5 | 0 | 0 | 0 | 1101 |
| | | 6 | 0 | 0 | 0 | 1110 |
| | |  | | | | |
| 3 | successful identification | 1 | 1 | 1 | 3 | |
| | | 2 | 0 | 1 | 1 | |
| | | 3 | 1 | 0 | 2 | |
| | | 4 | 1 | 0 | 2 | |
| | | 5 | 0 | 1 | 1 | |
| | | 6 | 0 | 0 | 0 | 1110 |
| | |  0<br>1<br>2<br>3<br>1110<br>0110<br>1101<br>0111<br>1000<br>0101 | | | | |
| 4 | collision | 1 | 3 | | 2 | |
| | | 2 | 1 | | 0 | 0110 |
| | | 3 | 2 | | 1 | |
| | | 4 | 2 | | 1 | |
| | | 5 | 1 | | 0 | 1101 |
| | | 6 | 0 | -- | -- | |
| | |  0<br>1<br>2<br>*<br>1110<br>0110<br>1101<br>0111<br>1000<br>0101 | | | | |
| 5 | successful identification | 1 | 2 | 1 | 5 | |
| | | 2 | 0 | 0 | 0 | 0110 |
| | | 3 | 1 | 1 | 3 | |
| | | 4 | 1 | 0 | 2 | |
| | | 5 | 0 | 1 | 1 | |
| | | 6 | -- | -- | -- | |
| | |  *<br>1110<br>0<br>1<br>2<br>3<br>4<br>5<br>0110<br>1101<br>1000<br>0111<br>0101 | | | | |
| 6 | successful identification | 0 | 5 | | 4 | |
| | | 1 | -- | -- | -- | |
| | | 0 | 3 | | 2 | |
| | | 4 | 2 | | 1 | |
| | | 5 | 1 | | 0 | 1101 |
| | | 6 | -- | -- | -- | |
| | |  *<br>1110<br>0<br>1<br>2<br>3<br>4<br>*<br>0110<br>1101<br>1000<br>0111<br>0101 | | | | |
| 7 | successful identification | 1 | 4 | | 3 | |
| | | 2 | -- | -- | -- | |
| | | 3 | 2 | | 1 | |
| | | 4 | 1 | | 0 | 1000 |
| | | 5 | 0 | -- | -- | |
| | | 6 | -- | -- | -- | |
| | |  *<br>1110<br>0<br>1<br>2<br>3<br>*<br>0110<br>*<br>1101<br>1000<br>0111<br>0101 | | | | |
| 8 | successful identification | 1 | 3 | | 2 | |
| | | 2 | -- | -- | -- | |
| | | 3 | 1 | | 0 | 0111 |
| | | 4 | 0 | | -- | |
| | | 5 | -- | -- | -- | |
| | | 6 | -- | -- | -- | |
| | | | | | | |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  |  |  *<br>1110<br>0<br>1<br>2<br>*<br>0110<br>*<br>1101<br>*<br>1000<br>0111<br>0101 |  |  |  |  |
| 9 | no tag response | 1 | 2 |  | 1 |  |
|  |  | 2 | -- | -- | -- |  |
|  |  | 3 | 0 |  | -- |  |
|  |  | 4 | -- | -- | -- |  |
|  |  | 5 | -- | -- | -- |  |
|  |  | 6 | -- | -- | -- |  |
|  |  |  *<br>1110<br>0<br>1<br>*<br>0110<br>*<br>1101<br>*<br>1000<br>*<br>0111<br>0101 |  |  |  |  |
| 10 | successful identification | 1 | 1 |  | 0 | 0101 |
|  |  | 2 | -- | -- | -- |  |
|  |  | 3 | -- | -- | -- | - |
|  |  | 4 | -- | -- | -- |  |
|  |  | 5 | -- | -- | -- |  |
|  |  | 6 | -- | -- | -- |  |
|  |  |  0<br>*<br>1110<br>0101<br>*<br>0110<br>*<br>1101<br>*<br>1000<br>*<br>0111 |  |  |  |  |
| 11 | finish | 1 | 0 | -- | -- |  |
|  |  | 2 | -- | -- | -- |  |
|  |  | 3 | -- | -- | -- |  |
|  |  | 4 | -- | -- | -- |  |
|  |  | 5 | -- | -- | -- |  |
|  |  | 6 | -- | -- | -- |  |
|  |  |  *<br>1110<br>*<br>0101<br>*<br>0110<br>*<br>1101<br>*<br>1000<br>*<br>0111 |  |  |  |  |

## 4. Simulation and Comparison

In this section, we show the simulation results of the PSM protocol and compare them with those of the FS-ALOHA, QT, and ISO18000-6B protocols in terms of the number of iterations needed to identify all tags and the system efficiency. The FS-ALOHA, QT, and ISO18000-6B protocols are typical plain ALOHA-based, deterministic tree-based and probabilistic counter-based protocols using no tag population estimation, bit-tracking, and re-identification techniques.

As mentioned earlier, an iteration is for a reader to send a command and for tags to perform corresponding actions like adjusting counter values and responding tag IDs. For the FS-ALOHA protocol, a time slot is equivalent to an iteration. The simulations are performed for 512, 612, ⋯, 2012 tags in the interrogation zone. We assume a frame in the FS-ALOHA protocol has initially $t$ time slots for a $t$-tag simulation case. We also assume tag IDs are 64-bit long and are uniformly distributed for simulating the QT protocol.

First, PSM is simulated for $2^u$-way splitting/$2^v$-way merging cases (denoted by $2^u/2^v$), where $1 \le u \le 4$ and $1 \le v \le u+2$. Table 3 shows parts of the simulation results, by which we can observe that PSM has better performance by taking $v$ as 1 (i.e., 2-way merging) for $u$=1,…,4, and the 2/2 case needs the fewest number of iterations. Therefore, it is suggested to set $u$=$v$=1 for PSM to achieve better performance.

Table3. The simulation results of the PSM protocol for different splitting/merging ways

| The number of identification iterations | splitting / merging |  |  |  |  |
|---|---|---|---|---|---|
|  | 2/2 | 4/2 | 4/4 | 4/8 | 4/16 |
| 100 | 272 | 292 | 349 | 368 | 404 |
| 200 | 541 | 581 | 700 | 739 | 821 |
| 300 | 808 | 871 | 1049 | 1106 | 1227 |
| 400 | 1078 | 1158 | 1388 | 1478 | 1630 |
| 500 | 1346 | 1451 | 1729 | 1847 | 2037 |
| 600 | 1615 | 1736 | 2083 | 2216 | 2457 |
| 700 | 1884 | 2023 | 2449 | 2584 | 2883 |
| 800 | 2150 | 2315 | 1796 | 2953 | 3300 |
| 900 | 2422 | 2604 | 3253 | 3320 | 3705 |
| 1000 | 2689 | 2892 | 3502 | 3688 | 4116 |
| The number of identification iterations | splitting / merging |  |  |  |  |
|  | 8/2 | 8/4 | 8/8 | 8/16 | 8/32 |
| 100 | 333 | 314 | 496 | 486 | 508 |
| 200 | 668 | 625 | 1063 | 981 | 1023 |
| 300 | 996 | 937 | 1609 | 1469 | 1530 |
| 400 | 1331 | 1250 | 2099 | 1948 | 2044 |
| 500 | 1665 | 1562 | 2574 | 2444 | 2556 |
| 600 | 1997 | 1876 | 3019 | 2937 | 3067 |
| 700 | 2328 | 2188 | 3460 | 3432 | 3583 |
| 800 | 2661 | 2498 | 3916 | 3920 | 4090 |
| 900 | 2993 | 2817 | 4431 | 4402 | 4612 |
| 1000 | 3324 | 3127 | 5007 | 4898 | 5115 |

By Figure 2, we can observe that the PSM protocol has the smallest number of iterations among these plain protocols using no special techniques. The FS-ALOHA protocol has the largest number of iterations needed to identify all tags, and the ISO18000-6B and QT protocols have nearly the same number of iterations.

By Figure 3, we can see that the PSM protocol has the highest system efficiency among all four plain protocols. The FS-ALOHA protocol has the lowest system efficiency due to its fixed size of frames. The system efficiency is defined in [20] as the ratio of the number of total number of tags to the total number of slots or iterations required to identify all tags. The ISO18000-6B and QT protocols have the similar system efficiency.

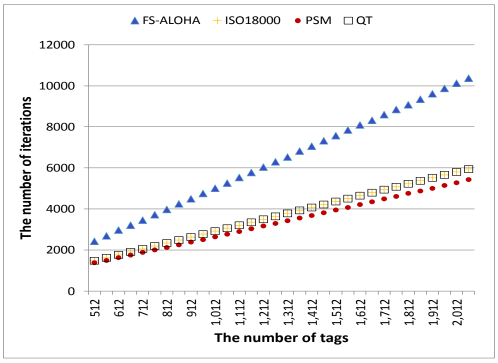


Figure 2. The comparison of plain anti-collision protocols in terms of the number of iterations needed to identify all tags

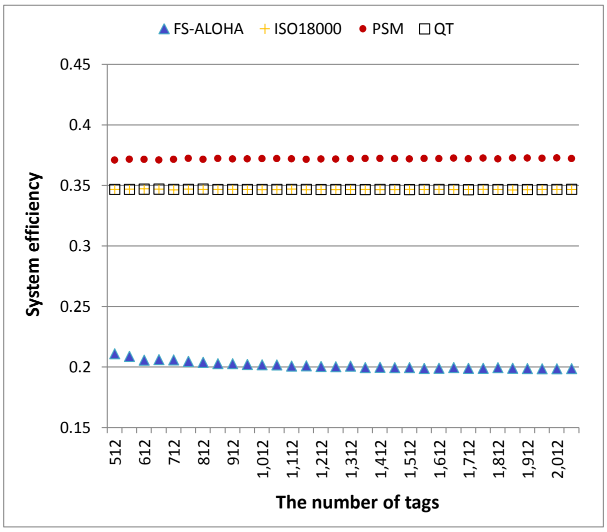


Figure 3. The comparison of plain anti-collision protocols in terms of the system efficiency

## 5. Conclusion

This paper proposes a probabilistic counter-based tag anti-collision protocol, called PSM, to reduce tag collisions for speeding up RFID tag identification by simply splitting and merging groups of tags in parallel. The PSM protocol is simple, since it uses no tag population estimation, which incurs complex computation. It does not use bit-tracking, which needs accurate bit collision detection, and does not use re-identification, which is just suitable for some cases where the reader needs to repeatedly identify similar sets of tags. As shown by the simulation results, PSM can significantly reduce the number of iterations (i.e., identification delay) needed to identify all tags. It has a smaller number of iterations than the FS-ALOHA, QT, and ISO18000-6B protocols that are typical ALOHA-based, deterministic tree-based, and probabilistic counter-based protocols, respectively. To the best of our knowledge, the PSM protocol has the highest system efficiency among the plain protocols that use no special techniques, such as bit-tracking, tag population estimation, and re-identification.

## References

[1] Finkenzeller, K., *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, John Wiley & Sons, 2003.

[2] Yeh, M. K., Jiang, J. R., and Huang, S. T., “Adaptive Splitting and Pre-signaling for RFID Tag Anti-collision,” *Computer Communications*, vol. 32, issue 17, pp.1862-1870, 2009.

[3] Vogt, H., “Efficient Object Identification with Passive RFID Tags,” in *Proc. of Pervasive Computing*, pp.98-113, 2002.

[4] Capetanakis, J. I., “Tree Algorithms for Packet Broadcast Channels,” *IEEE Trans. Inf. Theory*, vol. 25, pp.505-515, 1979.

[5] Hayes, J. F., “An Adaptive Technique for Local Distribution,” *IEEE Trans. Communication*, vol. 26, pp. 1178-1186, 1978.

[6] Tsybakov, B. S., and Mikhailov, V. A., “Free synchronous packet access in broadcast channel with feedback,” *Probl. Pereda. Inf.*, vol. 14, no. 4, pp. 32-59, 1978.

[7] ISO/IEC, “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,” *Final Draft International Standard ISO 18000-6*, 2003.

[8] Khandelwal, G., Yener, A., Lee, K., and Serbetli, S., “ASAP: A MAC Protocol for Dense and Time Constrained RFID Systems,” in *Proc. of ICC*, pp. 4028-4033, 2006.

[9] Myung, J., Lee, W., Srivastava, J., Timothy, K., and Shih, J., “Tag-Splitting: Adaptive Collision Arbitration Protocols for RFID Tag Identification,” *IEEE Trans. Parallel and Distributed Systems*, vol. 18, no. 6, 2007.

[10] Schoute, F. C., “Dynamic Frame Length ALOHA,” *IEEE Trans. Communication*, vol. 31, issue 4, pp. 565-68, Apr. 1983.

[11] Floerkemeier, C., “Transmission Control Scheme for RFID Object Identification,” in *Proc. of IEEE Int’l. Conf. on Pervasive Computing and Communications Workshops*, 2006.

[12] Floerkemeier, C., “Bayesian Transmission Strategy for Framed ALOHA Based RFID Protocols,” in *Proc.* of *IEEE Int’l. Conf. of RFID*, 2007.

[13] Popovski, P., Fitzek, F. H. P., and Prasad, R., “Batch Conflict Resolution Algorithm with Progressively Accurate Multiplicity Estimation,” in *Proc. of ACM DIALM-POMC*, 2004.

[14] Lai, Y. C., and Lin, C. C., "Two Couple-Resolution Blocking Protocols on Adaptive Query Splitting for RFID Tag Identification," *IEEE Trans. Mobile Computing*, vol. 11, issue 10, pp. 1450-1463, 2012.

[15] Yeh, M. K., and Jiang, J. R., "Parallel Splitting for RFID Tag Anti-Collision," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 8, issue 4, pp. 249-260, 2011.

[16] Zhu, L., and Yum, T. S. P., "A Critical Survey and Analysis of RFID Anti-collision Mechanisms," *IEEE Communications Magazine*, vol. 59, no. 5, pp. 214-221, 2011.

[17] Wang, G., Peng, Y., and Zhu, Z., "Anti-collision algorithm for RFID tag identification using fast query tree," in *Proc. of 2011 International Symposium on IT in Medicine and Education (ITME),* Volume 1, pp.396-399, 2011.

[18] Mathys, P., and Flajolet, P., "Q-ary collision resolution algorithms in random-access systems with free or blocked channel access, " *IEEE Trans. Inform. Theory*, vol. 31, no. 4, pp. 217-243, March 1985.

[19] Myung, J., and Lee, W., "Adaptive splitting protocols for RFID tag collision arbitration," in *Proc. of MobiHoc 2006*, pp. 202-213, 2006.

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