A Counter-Based RFID Anti-Collision Protocol Using Parallel Splitting

Ming-Kuei Yeh and Jehn-Ruey Jiang
Department of Computer Science and Information Engineering
National Central University, Taiwan, ROC
{92542010, jrjiang}@csie.ncu.edu.tw

Abstract*—In an RFID system, a reader identifies tags by interrogating them through a shared wireless communication channel. Collisions occur when multiple tags transmit their IDs to the reader simultaneously, degrading the performance of tag identification. How to reduce tag collisions to speed up the identification is thus important. There are several anti-collision protocols proposed for dealing with tag collisions. 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. The ALOHA-based protocol has the tag starvation problem that a tag may never be successfully identified; the deterministic tree-based protocol has the problem that its performance is influenced by the tag ID length and/or distribution. On the contrary, the probabilistic counter-based protocol has no such problems. In this paper, we propose a probabilistic counter-based anti-collision protocol using the idea of parallel splitting to speed up RFID tag identification. The proposed protocol also utilizes an adaptive identification-tree height adjustment mechanism to fine tune the effect of parallel splitting. We analyze and simulate the proposed protocol and compare it with the counter-based protocol adopted by the well known ISO/IEC 18000-6B standard to demonstrate its advantages.

1. Introduction

The RFID (Radio Frequency IDentification) technique [5] attracts a lot of attention recently due to its automatic identification capability through RF communications. An RFID system consists of readers and 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. Most RFID tags do not have on-tag power source; they derive energy from the RF field generated by the reader to drive the circuit. This is an 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 its 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 request simultaneously, signal 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 *anti-collision* protocols proposed for reducing tag collisions. They can be categorized into two classes [17]: ALOHA-based protocols and tree-based protocols that include deterministic tree-based and probabilistic counter-based subclasses of protocols.

This paper presents a novel probabilistic counter-based anti-collision protocol, called *Parallel Splitting (PS)* protocol, to speed up RFID tag identification procedure by splitting tags in parallel. PS protocol utilizes two mechanisms: the *parallel splitting* mechanism and the *adaptive identification-tree height adjustment* mechanism. In the best case, PS protocol needs only $\lceil \lg N \rceil + N - 1$ iterations to identify all N tags, while the related ISO/IEC 18000-6B protocol needs $2N-1$ iterations. We analyze and simulate PS protocol and compare it with ISO/IEC 18000-6B protocol to show its advantages.

The rest of this paper is organized as follows. We describe the anti-collision protocols in Section II and show some observations about tag splitting of the probabilistic counter-based protocol in Section III and describe PS protocol in Section IV. We analyze PS protocol in Section V, and simulate it and compare it with ISO/IEC 18000-6B protocol in Section VI. And finally, conclusion is drawn in Section VII.

2. Anti-collision protocols

2.1. The ALOHA-based protocols

In ALOHA-based protocols [3],[7],[9],[11] tags respond to the reader by transmitting IDs in a probabilistic manner. In ALOHA protocol [1], 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 no collision occurs during a tag's ID response, its ID 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 and send its ID until it is identified and acknowledged by the reader. In slotted ALOHA protocol [12], 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 [14], the performance of slotted ALOHA protocol is twice that of ALOHA protocol since there is no partial collision of tag ID responses in slotted ALOHA protocol.

Frame slotted ALOHA protocol [16] is similar to slotted ALOHA protocol. However, to limit the response time, frame slotted ALOHA protocol [16] 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 [3], [9], [10], [11] try to eliminate the drawback by dynamically adjusting the frame size according to the estimated number of tags. Their performances are therefore better than that of frame slotted ALOHA protocol.

2.2. The deterministic tree-based protocols

In [2], [6], [16], the splitting tree concept was proposed to solve the collision problem. The deterministic tree-based protocols [4], [18] rely on tag IDs to repeatedly split colliding tags into subgroups until there is only one tag in a subgroup to be identified successfully. In query tree protocol (QT) [13],[18], 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., $S0$ or $S1$). 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. Query tree protocol is a memory-less protocol because it does not require tags to be equipped with additional writable on-chip memory. We can observe that QT protocol's 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 protocol [4], on receiving a reader's interrogation request, each tag responds with the first bit of its tag ID. The reader then responds with 0 (or 1), and only the tags with the first bit being 0 (or 1) will respond with its next ID bit. The above procedure repeats bit by bit until there is only one responding tag. The reader can then ask the tag to send remaining ID bits for the purpose of identification. The protocol requires tags to be equipped with writable on-tag memory so that tags can keep track of the identification procedure to respond with a certain bit properly. Unlike QT protocol, bit-by-bit binary tree protocol does not require a reader to send long ID prefixes; the reader and the tag send out only one bit at a time. Consequently, the delay time of the identification procedure is reduced. However, in the case of sparse tags with uniform ID distribution, the identification performance may be worse than that of query tree protocol.

2.3. The probabilistic counter-based protocols

Probabilistic counter-based protocols [8], [11], [13] rely on dynamically changing counters to split colliding tags. In the anti-collision protocol[1] proposed in the well known ISO/IEC 18000-6B standard [8], each tag maintains a counter which is initially 0. Every tag with counter value 0 can transmit its tag ID to respond to the 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.

ABS (Adaptive Binary Splitting) protocol [13] is proposed to improve ISO/IEC 18000 6B protocol by keeping counter information of the last tag 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 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 [13], the performance of ISO/IEC 18000-6B protocol is improved significantly by the ABS protocol.

Among the two types of anti-collision protocols, ALOHA-based protocols are simple and have fair performance. However, they have the tag starvation problem that a tag may never be identified when its responses always collide with others'. Deterministic

[1] For the sake of simplicity, we use "ISO/IEC 18000-6B protocol" to refer to the counter-based anti-collision protocol adopted by the well known ISO/IEC 18000-6B standard.

Tree-based protocols and probabilistic counter-based protocols do not have the tag starvation problem. Yet, the former have the problem that their performance is influenced by the tag ID length and/or distribution, while the latter have not. We thus focus on probabilistic counter-based protocols in this paper.

3. Tag-splitting of the counter-based protocol

As we have shown, in the representative probabilistic counter-based protocol, ISO/IEC 18000-6B protocol, the tags with counter value 0 are split into two subgroups, one for tags with counter value 0 and the other for tags with counter value 1. The splitting procedure continues until (case-1) only one tag is of counter 0 or (case-2) none tag is of counter value 0. In case-1, the tag having counter value 0 will respond to the reader and be identified successfully. It should keep silent until the tag interrogation is finished. In both cases the reader sends a command to inform all tags to decrease their counters by 1. Afterwards, tags with counter value 0 are split again. In this way, all tags can be identified successfully.

For the purpose of observing tag splitting of ISO/IEC 18000-6B protocol, we show below an example of the protocol's identification procedure. We assume there are four tags with unique IDs 0010, 0110, 1001 and 1110. The iterations (steps) of the tag identification procedure and associated identification trees are depicted in Table 1. The iterations are also described as follows.

Iteration 1: At the beginning, the reader requests tags to start a round of tag interrogation. On receiving the request, tags reset their counters to 0. Tag1 (with ID 0010), tag2 (with ID 0110), tag3 (with ID 1001) and tag4 (with ID 1110) respond with their IDs to the reader simultaneously and collisions happen.

Iteration 2: The reader sends a collision-notification command to make all tags randomly add 0 or 1 to their counters. This is the first tag splitting. As shown in Table 1, tags 1 and 4 are with counter value 0, while tags 2 and 3 are of counter value 1 after the splitting. Tags 1 and 4 respond with their IDs simultaneously and collisions occur again.

Iteration 3: The reader sends a collision-notification command to make tags 1 and 4 randomly add 0 or 1 to their counters, while tags 2 and 3 increase 1 to their counters. This is the second tag splitting. Tag 1 is the only tag of counter value 0 after the splitting; it responds with its ID to the reader and is identified successfully.

Iteration 4: The reader acknowledges the identified ID with a success-notification command. The identified tag 1 enters the silent state, and all unidentified tags 2, 3 and 4 decrease their counters by 1. Tag 4 is the only tag of counter value 0; it responds with its ID to the reader and is identified successfully.

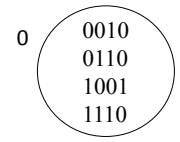
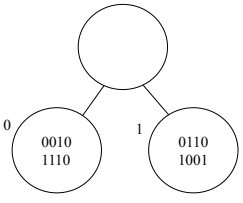
Iteration 5: The reader acknowledges the identified ID with a success-notification command. The identified tag 4 enters the silent state, and all unidentified tags 2 and 3 decrease their counters by 1. Tags 2 and 3 are of counter value 0; they respond their IDs to the reader and collision occurs.

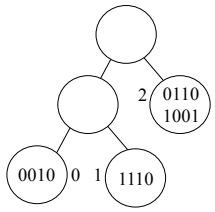
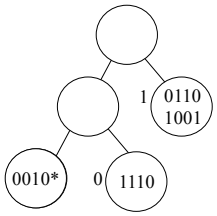
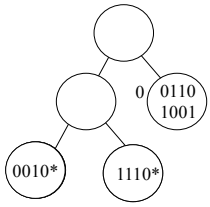
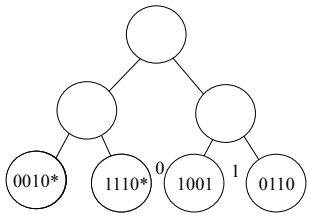
Iteration 6: The reader sends a collision-notification command to make tags 2 and 3 randomly add 0 or 1 to their counters. This is the third tag splitting. Tag 3 is the only tag of counter value 0 after the splitting; it responds with its ID to the reader and is identified successfully.

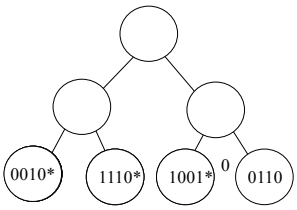
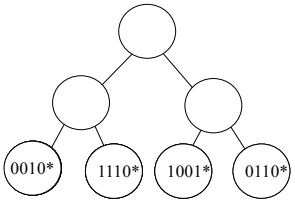
Iteration 7: The reader acknowledges the identified ID with a success-notification command. The identified tag 3 enters the silent state, and the unidentified tag 2 decreases its counter by 1. Tag 2 is the only tag of counter value 0; it responds with its ID to the reader and is identified successfully.

Iteration 8: The reader acknowledges the identified ID with a success-notification command. The identified tag 2 enters the silent state. At that time, all tags are identified and thus will be silent until the reader sends a request to start the next round.

Table 1. The iterations of the identification procedure of ISO/IEC 18000-6B protocol and the associated identification tree (The tag ID marked by '*' means that the associated tag has been identified.)

Iteration	Reader Command	Tag ID	Counter value	Random bit choose	New Counter Value	Tag Response
1	Request	1	--		0	0010
		2	--		0	0110
		3	--		0	1001
		4	--		0	1110
						
2	Collision	1	0	0	0	0010
		2	0	1	1	
		3	0	1	1	
		4	0	0	0	1110
						
3	Collision	1	0	0	0	0010
		2	1		2	
		3	1		2	

		4	0	1	1	
						
4	Success	1	0		--	
		2	2		1	
		3	2		1	
		4	1		0	1110
						
5	Success	1	--		--	
		2	1		0	0110
		3	1		0	1001
		4	0		--	
						
6	Collision	1	--		--	
		2	0	1	1	
		3	0	0	0	1001
		4	--		--	
						
7	Success	1	--		--	
		2	1		0	0110
		3	0		--	
		4	--		--	

						
8	Success	1	--		--	
		2	0		--	
		3	--		--	
		4	--		--	
						

By investigating the tag identification procedure and its associated identification trees of ISO/IEC 18000-6B protocol, we have the following observations:

1. A successfully identified tag corresponds to a leaf node in the identification tree.
2. Each node in the identification tree corresponds to an iteration of the identification procedure.
3. The tag splitting occurs among only the tags with counter value 0. Other tags do not perform splitting but just increase their counters by 1 even though they have the same counter value and their responses to the reader will collide in a coming iteration.

By the first two observations, we can infer that the ISO/IEC 18000-6B protocol needs $2N$–1 iterations for identifying N tags in the best case. This is because a tree with N leaf nodes has a total number of $2N$–1 of nodes (Note that in a worse case, a leaf node may not correspond to any tag ID, which leads to a total number of nodes more than $2N$-1. This occurs if no tag is with the counter value 0 when the reader requests tags to send their IDs.) The one shown in Table 1 is actually a best case example of the tag identification procedure of ISO/IEC 18000-6B protocol. It takes 7 iterations, excluding the first iteration to start a new round, to identify 4 tags.

The last observation further inspires us to develop a counter-based protocol that splits tags in parallel to speed up the tag identification procedure. We propose a protocol, called *parallel splitting (PS)* protocol, to achieve this goal. In the next section, we will elaborate all details of PS protocol.

4. The Proposed Protocol

The basic concept of PS protocol is for unidentified tags to left shift their counters (i.e. multiply the counter values by 2) and then randomly add 0 or 1 to the counters when collisions occur before the first tag is identified. And after the first tag is identified, PS protocol will follow the normal

identification procedure of ISO/IEC 18000-6B protocol. In this way, tags are split into groups in parallel and the delay to identify all tags can be shortened.

In ISO/IEC 18000-6B protocol, only tags of counter value 0 are split, while in PS protocol, all unidentified tags are split simultaneously before the first tag is identified. Fig. 1 shows the comparison of the best case identification trees of ISO/IEC 18000-6B and PS protocols. By Fig. 1, we can observe that PS protocol can generate more leaf nodes than ISO/IEC 18000-6B protocol within the same number of iterations. Since leaf nodes correspond to a tag ID to be identified, we can infer that PS protocol needs fewer iterations for all tags to be identified successfully.

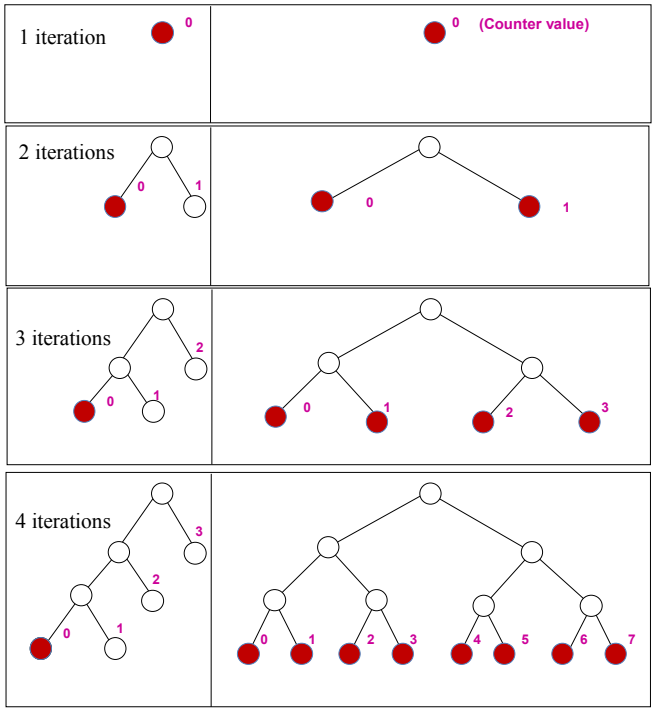


Figure 1. The identification trees of ISO/IEC 18000-6B protocol (left) and PS protocol (right) before the first tag is identified (Note that tag splitting will occur at shaded nodes.)

By Fig.1, we can observe that after H iterations, there will be 2^H leaf nodes in the identification tree in PS protocol. Therefore, the minimum iterations to generate N leaf nodes in the identification tree is $\lceil \lg N \rceil$, which corresponds to the height of a full tree of N leaf nodes. If each leaf node corresponds to only one tag (i.e., all tags have different counters), then no further tag splitting is needed for tags to be identified successfully. However, the parallel splitting stops when the first tag is identified. At that time, if less than $\lceil \lg N \rceil$ iterations have been executed, then the number of leaf nodes will be less than N and there will be several tags residing at the same node, which will call for more tag splitting. On the other hand, if more than $\lceil \lg N \rceil$ iterations have been executed, then there will be some empty leaf nodes (i.e., no tag is with the counter values associated with the nodes). This will lengthen the identification procedure, since an empty leaf node corresponds to an iteration in which no tag responds to the reader. Therefore, a mechanism is needed to adjust the height of the identification tree once the first tag is identified.

Below we introduce the adaptive identification-tree height adjustment mechanism to adjust the height of the identification tree for the purpose of making the number of leaf nodes approximate the number of tags. Since the number of tags is unknown beforehand, we set the tree height according to the ratio of N_0, N_1 and N_m, which are the numbers of leaf nodes with zero tag, one tag and multiple (i.e., two or more) tags, respectively, when the first tag is successfully identified. In the following context, we will show how to derive an approximate ratio of N_0, N_1 and N_m for adjusting the tree height.

Suppose that the number of tags is N. When the first tag is identified, we have the minimum number of iterations for identifying all tags if the identification tree is a full tree. Such a tree has a height of $\lceil \lg N \rceil$, and the number L of leaf nodes of the tree is $2^{\lceil \lg N \rceil}$. We have

$$L = 2^{\lceil \lg N \rceil} \tag{1}$$

Let the probability that there are S tags residing at a leaf node be $p(S)$. Assuming a tag has the same probability to reside at every leaf node, we have

$$p(S) = \binom{N}{S}\left(\frac{1}{L}\right)^S\left(1-\frac{1}{L}\right)^{N-S} \tag{2}$$

We let $p(m)$ be the probability of 2 or more tags residing at a leaf node. We have

$$p(m) = p(2) + p(3) + p(4) + \cdots$$

From Eq. 2, we have

$$\frac{p(3)}{p(2)} = \frac{\frac{N(N-1)(N-2)}{3!}\left(\frac{1}{L}\right)^3\left(1-\frac{1}{L}\right)^{N-3}}{\frac{N(N-1)}{2}\left(\frac{1}{L}\right)^2\left(1-\frac{1}{L}\right)^{N-2}} = \frac{2(N-2)}{3!\,(L-1)} = \frac{(N-2)}{3(L-1)}$$

$$\frac{p(4)}{p(2)} = \frac{\frac{N(N-1)(N-2)(N-3)}{4!}\left(\frac{1}{L}\right)^4\left(1-\frac{1}{L}\right)^{N-4}}{\frac{N(N-1)}{2}\left(\frac{1}{L}\right)^2\left(1-\frac{1}{L}\right)^{N-2}} = \frac{2(N-2)(N-3)}{4!\,(L-1)^2} = \frac{(N-2)(N-3)}{12(L-1)^2}$$

$$\frac{p(5)}{p(2)} = \frac{\frac{N(N-1)(N-2)(N-3)(N-4)}{5!}\left(\frac{1}{L}\right)^5\left(1-\frac{1}{L}\right)^{N-5}}{\frac{N(N-1)}{2}\left(\frac{1}{L}\right)^2\left(1-\frac{1}{L}\right)^{N-2}}$$

$$= \frac{2(N-2)(N-3)(N-4)}{5!\,(L-1)^2} = \frac{(N-2)(N-3)(N-4)}{60(L-1)^2}$$

It is reasonable to assume $L >> 1$ and $N >> 1$. It is thus reasonable to take $L-1$ as L, and to take $N-2$, $N-3$ and $N-4$ as N. Then we have

$$p(2):p(3):p(4):p(5):\ldots:p(i):\ldots$$
$$= 1:\frac{N}{3L}:\frac{N^2}{12L^2}:\frac{N^3}{60L^3}:\ldots:\frac{2\,N^{(i-2)}}{i!\,L^{(i-2)}}:\ldots \quad (3)$$

By Eq. 3, we can observe that $p(4)$, $p(5)$,… are relatively small when compare with $p(2)$. We thus omit the probability that there are more than 3 tags residing at a leaf node. We have

$$\begin{aligned} p(m) &\approx p(2)+p(3) \\ &= \frac{N(N-1)}{2}\left(\frac{1}{L}\right)^2\left(1-\frac{1}{L}\right)^{N-2} + \\ &\quad \frac{N(N-1)(N-2)}{3!}\left(\frac{1}{L}\right)^3\left(1-\frac{1}{L}\right)^{N-3} \\ &= \left(1+\frac{(N-2)}{3(L-1)}\right)\left(\frac{N(N-1)}{2}\left(\frac{1}{L}\right)^2\left(1-\frac{1}{L}\right)^{N-2}\right) \end{aligned} \quad (4)$$

Let the expected numbers of leaf nodes containing none, one and multiple tags be $e(0)$, $e(1)$ and $e(m)$, respectively.

$$\begin{aligned} &e(0):e(1):e(m) \\ &= L\times p(0):L\times p(1):L\times p(m) \\ &= L\left(1-\frac{1}{L}\right)^N : N\left(1-\frac{1}{L}\right)^{N-1} : L\left(1+\frac{(N-2)}{3(L-1)}\right) \\ &\quad \left(\frac{N(N-1)}{2}\left(\frac{1}{L}\right)^2\left(1-\frac{1}{L}\right)^{N-2}\right) \\ &= 2(L-1)^2 : 2N(L-1) : N(N-1)\left(1+\frac{(N-2)}{3(L-1)}\right) \end{aligned} \quad (5)$$

It is reasonable to assume $L>>1$ and $N>>1$, and take $L-1$ as L, and $N-1$, $N-2$ as N. We then have

$$e(0):e(1):e(m) \;=\; 2L^2:2NL:\;N^2\left(1+\frac{N}{3L}\right) \quad (6)$$

By Eq.6, we can calculate the expected ratios of $e(0)$, $e(1)$ and $e(m)$ for different values of L and N. We show such ratios in Table 2.

Table 2. The expected ratios of $e(0)$, $e(1)$ and $e(m)$ for different values of L and N (we assume $L>>1$)

L vs. N	$e(0)$	$e(1)$	$e(m)$
$L=2N$	2	1	7/24
$L=N$	1	1	2/3
$L=N/2$	1/2	1	5/3

With the expected ratios of $e(0)$, $e(1)$ and $e(m)$ in Table 2, we can adjust L to approach N by keeping track of N_0, N_1 and N_m. For example, if the reader finds that $N_0 : N_1 : N_m \cong 2 : 1 : 7/24$ during the identification process, it presumes that $L \geq 2N$ and forces all unidentified tags right-shift their counters one bit to make L approach N. On the other hand, if the reader finds that $N_0 : N_1 : N_m \cong 1/2 : 1 : 5/3$ during the identification process, it presumes that $L \leq N/2$ and forces all unidentified tags left-shift their counters one bit and add one or zero randomly to make L approach N. In summary, we have the following two rules to make L approach N.

Rule 1: If $N_0 > 2\,N_1$ and $N_1 > 3.4\,N_m$, then all unidentified tags right-shift their counters one bit to divide L by 2.

Rule 2: If $2N_0 < N_1$ and $1.7N_1 < N_m$, then all unidentified tags left-shift their counters one bit to multiply L by 2, and subsequently add one or zero randomly to the counters.

Below, we give an overview of PS protocol, which has two phases. In phase I, all tags are split in parallel until the first tag is identified successfully. In phase II, tags are identified one by one according to the normal identification procedure of ISO/IEC 18000-6B protocol. It is noted that the above-mentioned Rule1 and Rule2 may be applied in phase II for adjusting the number of leaf nodes to approach the actual number N of tags even though N is unknown.

Besides the COLLISION and SUCCESS commands used in ISO/IEC 18000-6B protocol, PS protocol utilizes two more commands, PS_COLLISION and PS_SUCCESS, to facilitate parallel tag splitting. We assume each tag has a K-bit counter that can store up to 2^K different counter values. If the number of tag splitting is over K, the counter value will overflow and errors will occur. We use parameter C to record and control the number of tag splitting. When $C < K$, PS_COLLISION command is sent to make all tags split in parallel; otherwise, COLLISION command is sent to make tags split in serial. In this way, the counter value hardly overflows.

The operations of the reader and the tag are described in Fig. 2 and Fig. 3, respectively.

The operation of the reader
(Phase I) /* K is the length of tag counter /* C records the number of PS_COLLISION commands sent in the current round. 1. The reader sends a REQUEST command to make all tags start a new round and reset counter values to 0. The reader waits for responses from tags. 2. If two or more responses are received, then the reader checks if $C < K$. If so, PS_COLLISION command is sent to make all tags split in parallel. If not so, the reader sends COLLISION command to make tags split in serial and the reader goes to Step 5 in Phase II. 3. If no or only one response is received, then the reader sends SUCCESS command to make all unidentified tags decrease their counters by 1 and the reader goes to Step 5. 4. The reader waits for responses from tags. On receiving responses, the reader goes to Step 2.

(Phase II)

5. The reader waits for responses from tags.
6. If collisions occur, the reader performs either a or b
   a. If Rule2 is applicable, then the reader checks if $C < K$. If so, PS_COLLISION command is sent. Otherwise, COLLISION command is sent.
   b. If Rule2 is not applicable, then COLLISION command is sent.
7. If only one or no tag responds, the reader performs either a or b
   a. If Rule1 is applicable, then PS_SUCCESS command is sent and C is decreased by 1.
   b. If Rule1 is not applicable, then SUCCESS command is sent.
8. If there are still tags to be identified, the reader goes to Step 5.

Figure 2. The operation of the reader in PS protocol

The operation of tags

1. All tags reset counter value to be 0 on receiving the REQUEST command sent by the reader.
2. The tags with counter value 0 will send their IDs to the reader.
3. On receiving a command from the reader, tags take actions according to the following cases:
   case command = COLLISION:
   The tag with counter value 0 randomly adds 0 or 1 to counters, while other unidentified tags increase their counters by 1.
   case command = PS_COLLISION:
   All unidentified tags left-shift their counters one bit and randomly add 0 or 1 to the counters
   case command = SUCCESS:
   The tag with counter value 0 will keep silent and become an identified tag until next round of the identification procedure, while other tags decrease their counters by 1.
   case command = PS_SUCCESS:
   All unidentified tags right-shift their counters by one bit.

Figure 3. The operation of tags in PS protocol

5. The analysis of PS protocol

5.1. The minimum number of iterations for identifying all tags

We have the minimum number of iterations to identify all N tags if the identification tree is a full tree when the first tag is identified. Such a tree has a height of $\lceil \lg N \rceil$, and the number of leaf nodes is N. Since $\lceil \lg N \rceil$ iterations are needed in phase I to split tags into N groups and to identify the first tag, and further N–1 iterations are needed in phase II to identify other N–1 tags, we have that $\lceil \lg N \rceil + N - 1$ is the minimum number of iterations to identify all N tags.

5.2. The average number of iterations for identifying all tags

Suppose the identification procedure has gone through H iterations when the first tag is identified. We have the number L of leaf nodes in the identification tree is 2^H. Each leaf node can contain none, one or multiple tags. In the case of none and one tag residing at a leaf node, no more split is needed for tag identification, while in the case of multiple tags residing at a leaf node, more splits are needed.

We define $f(S)$ as the identification iterations needed to identify S tags at a leaf node. For example, $f(2)$ can be utilized to estimate the number of iterations needed to identify 2 tags at a leaf node. Table 3 shows the four possible cases after a split of 2 tags. By Table 3, the total number $f(2)$ of iterations needed to identify 2 tags residing at a leaf node can be easily calculated as

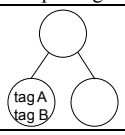
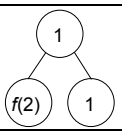
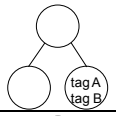
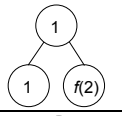
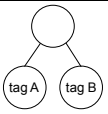
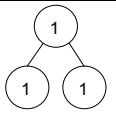
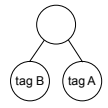
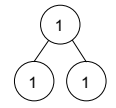
$$\begin{aligned} f(2) = &\left(\frac{1}{2^2}\left(1 + f(2) + f(0)\right)\right) \\ &+ \left(\frac{1}{2^2}\left(1 + f(0) + f(2)\right)\right) \\ &+ \left(\frac{1}{2^2}\left(1 + f(1) + f(1)\right)\right) \\ &+ \left(\frac{1}{2^2}\left(1 + f(1) + f(1)\ \right)\right) \end{aligned} \tag{7}$$

It is obviously, when only one tag residing at a leaf node, the total number $f(1)$ of iteration needed to identify is 1. From Eq.7, we obtain $f(2)=5$.

In order to figure out the general case, we can rewrite the Eq.7 as

$$\begin{aligned} f(2) &= 1 + \left(\frac{1}{2^2}\left(\left(f(2) + f(0)\right) + \left(f(0) + f(2)\right)\right.\right. \\ &\qquad \left.\left. + \left(f(1) + f(1)\right) + \left(f(1) + f(1)\right)\right)\right) \\ &= 1 + \left(\frac{1}{2^2}\left(\left(f(0) + f(2)\right) + 2\left(f(1) + f(1)\right)\right.\right. \\ &\qquad \left.\left. + \left(f(2) + f(0)\right)\right)\right) \\ &= 1 + \tfrac{1}{2^2}\left(\textstyle\sum_{i=0}^{2}\binom{2}{i}\left(f(i) + f(2-i)\right)\right) \end{aligned} \tag{8}$$

Table 3. An example of splitting two tags residing at a leaf node

$f(2)$ tag A tag B		
The identification tree after one splitting	The probability of the occurrence	The number of iterations needed at each node of the identification tree
	$\frac{1}{4}$	
	$\frac{1}{4}$	
	$\frac{1}{4}$	
	$\frac{1}{4}$	

The number of iterations needed to identify S tags at a leaf node for general cases is given below as

$$f(S)=\begin{cases}1 & ,S=0\\ 1 & ,S=1\\ 1+\frac{1}{2^S}\left(\sum_{i=0}^{S}\binom{S}{i}\left(f(i)+f(S-i)\right)\right) & ,S\geq 2\end{cases} \quad (9)$$

As shown in Eq. 2, the probability of S tags residing at one of the L leaf nodes is $p(S)=\binom{N}{S}\left(\frac{1}{L}\right)^S\left(1-\frac{1}{L}\right)^{N-S}$. Combining Eqs. 2 and 9, we can easily calculate the total number $T(N)$ of iterations needed to identify N tags under the assumption that the first tag is identified at iteration H (hence, there are $L=2^H$ leaf nodes). We define $g_j(k)$ as the probability that there are exactly k tags at the first node in 2^j nodes.

$$g_j(k)=\binom{N}{k}\left(\frac{1}{2^j}\right)^k\left(1-\frac{1}{2^j}\right)^{N-k} \quad (10)$$

The probability that the first tag is identified at the jth iteration is

$$h_j(1)=\left(\prod_{i=1}^{j-1}\left(1-g_i(1)\right)\right)\times g_j(1) \quad (11)$$

We have $T(N)$ as

$$T(N)=\sum_{i=1}^{n}\left(\left(\frac{h_i(1)}{\sum_{j=1}^{N}h_j(1)}\right)\left(i+\sum_{S=0}^{N}2^i\times p(S)\times f(S)\right)\right) \quad (12)$$

Because we do not consider the adaptive tree-height adjustment mechanism in Eq. 12, the actual average number of iterations should be less than T(N) in Eq. 12. We will show that by some simulation experiments.

6. Simulation

In this section, we describe simulation experiments that measure the number of iterations needed to identify tags in PS protocol. In the first set of simulations, we perform experiments 1000 times for the cases of N=1025, 1075, 1125,…, 2025 tags. As we have shown, PS protocol performs well if the condition holds that the first tag is identified after $H=\lceil \lg N\rceil=11$ iterations, where H is the height of the associated identification tree and N is the number of tags. However, by the simulation results, we observe that not all experiment instances meet with the condition. To see the effect of the identification tree height H on the number of iterations needed to identify all N tags, we groups experiment instances into 5 groups, denoted as $\lceil \lg N\rceil+\alpha$, where α = –2, –1, 0, 1 and 2. The grouping is according to the height of the identification tree when the first tag is identified. For example, if the identification tree height is $\lceil \lg N\rceil-2$ when the first tag is identified for a case, the case if added into the $\lceil \lg N\rceil-2$ group.

For each group of experiment instances, we derive the average number of iterations needed to identify all tags and plot the results in Figure 4. We can check that the simulation results coincide with the analysis results perfectly in the cases of $\lceil \lg N\rceil$, $\lceil \lg N\rceil-1$ and $\lceil \lg N\rceil-2$ groups. For the other two cases of $\lceil \lg N\rceil+1$ and $\lceil \lg N\rceil+2$ groups, the simulate results are much better than the analysis results. This is because the simulation of PS protocol considers the adaptive mechanism to adjust the height of the identification tree, while the analysis does not. This also demonstrates that the adaptive tree height adjustment mechanism indeed improve performance dramatically.

By Figure 4, we can also observe that α =0 leads to relatively small numbers of iterations for identify all N tags. This is because the number L of leaf nodes approaches the number of tags when α =0 (i.e., when the identification tree height H is $\lceil \lg N\rceil$). On the contrary, the number of iterations needed to identify all tags is relatively large when α =1 or 2. This is because L is twice or four times of N when α =1 or 2 and there are far more iterations needed to identify all N tags for such a number L of leaf nodes.

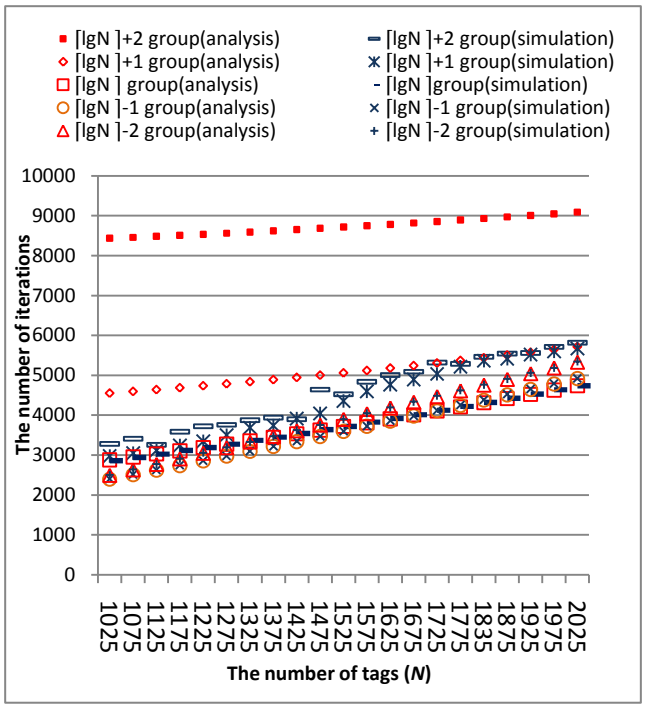


Figure 4. The simulation results and analysis results of the number of iterations needed to identify N tags for different $\lceil \lg N\rceil+\alpha$ groups, where α = –2, –1, 0, 1 and 2

In the second set of simulations, we perform experiments for both PS protocol and ISO/IEC 18000-6B protocol for the cases of N=512, 562, 612,…, 2012 tags. The simulation results are plotted in Figure 5 for the sake of comparison. We can observe that PS protocol outperforms ISO/IEC 18000-6B protocol in terms of the number of iterations needed to identify all tags. We can also observe

that the outperformance is more evident when the number N of tags is larger. In Figure 5, we also plot the analysis result of Eq. 12, which does not consider the adaptive adjustment of the identification-tree height during the identification procedure, in order to again show the advantage of the adaptive identification-tree height adjustment mechanism. It is very obvious that with the help of the mechanism, PS protocol needs fewer iterations to identify all tags.

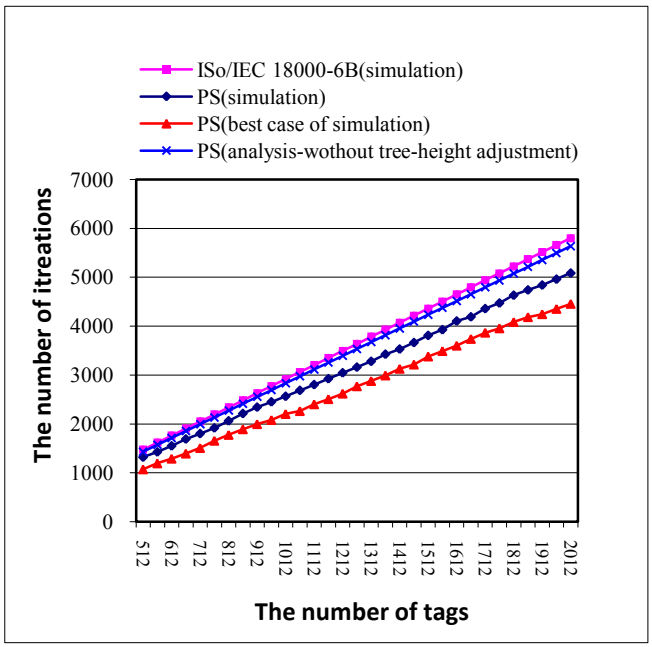


Figure 5. The comparison between PS protocol and ISO/IEC 18000-6B protocol in terms of the number of iterations needed to identify all tags

7. Conclusion

In this paper, we have proposed a probabilistic counter-based anti-collision protocol, called parallel splitting (PS) protocol, using the parallel splitting mechanism to speed up RFID tag identification. Before the first tag is identified, all unidentified tags left shift their counters (i.e. multiply the counter values by 2) and then randomly add 0 or 1 to the counters when collisions occur. In this way, the unidentified tags are split into subgroups in parallel. After the first tag is identified, the tags are identified one by one according to the normal identification procedure of ISO/IEC 18000-6B protocol. We have also embedded an adaptive identification-tree height adjustment mechanism to make the number of leaf nodes approach the actual number of tags to keep the number of iterations needed to identify all tags as small as possible.

We have analyzed PS protocol and simulated it in terms of the number of iterations needed to identify all N tags. We can see that the analysis results coincide with the simulation results perfectly when the identification tree is about $\lceil \lg N \rceil$ height when the first tag is identified. We have also compared the simulation results with those of ISO/IEC 18000-6B protocol. The comparison shows that PS protocol is evidently better than ISO/IEC 18000-6B protocol.

References

[1] N. Abramson, "The ALOHA system - another alternative for computer communications", *Proc. of AFIPS Spring Joint Computer Conf.*, vol. 37, pp. 281-285, 1970.

[2] J. I. Capetanakis, "Tree algorithms for packet broadcast channels", *IEEE Trans. Inf. Theory*, Vol. 25, pp.505–515, 1979.

[3] Jae-Ryong Cha and Jae-Hyun Kim, "Novel anti-collision algorithms for fast object identification in RFID system", *Proc. of the 11th International Conf. Parallel and Distributed Systems (ICPADS'05)*, pp.63-67, 2005.

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

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

[6] J. F. Hayes, "An adaptive technique for local distribution", *IEEE Trans. Commun.*, vol. 26, pp.1178–1186, 1978.

[7] T. W. Hwang et al., "Improved anti-collision scheme for high speed identification in RFID system", *Proc. of ICICIC*, pp449-452, Aug. 2006.

[8] 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, Nov. 2003.

[9] Girish Khandelwal et al., "ASAP: A MAC protocol for sense and time constrained RFID systems", *Proc. of IEEE International Conf. Communications, ICC'06,* Jun. 2006.

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

[11] 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.

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

[13] J. L. Massey, "Collision-resolution algorithms and random-access communications", *Multi. User Communication Systems,* Springer-Verlag, pp. 73-99, 1981.

[14] 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.

[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] B. S. Tsybakov, V. A. Mikhailov, "Free synchronous packet access in broadcast channel with feedback", *Probl. Pereda. Inf*, vol. 14(4), pp. 32–59 ,1978.

[17] H. Vogt, "Efficient Object Identification with Passive RFID Tags", *Proc. of Pervasive Computing,* pp.98–113, 2002.

[18] Ming-Kuei Yeh, Jehn-Ruey Jiang and Shing-Tsaan Huang, " Adaptive splitting and pre-signaling for RFID tag anti-collision", *Computer Communications*, to appear.

[19] Feng Zhou et al., "Evaluating and optimizing power consumption of anti-collision protocols for applications in RFID systems", *Proc. of the International Symposium on Low Power Electronics and Design*, Aug. 09-11, 2004.