A New Collision Resolution Protocol for Mobile RFID Tags

Jaewook Yu
SUNY at Stony Brook
Electrical and Computer Eng.
New York, USA
jwyu@ece.sunysb.edu

Eric Noel
AT&T Labs Research
Middletown
New Jersey, USA
eric.noel@att.com

Wendy Tang
SUNY at Stony Brook
Electrical and Computer Eng.
New York, USA
wtang@ece.sunysb.edu

Abstract

Radio Frequency Identification (RFID) systems have been widely deployed for inventory management, personal identification, and asset tracking. In such environments, RFID tags frequently move in and out of a reader's interrogation zone. Consequently, it is necessary for a reader to keep track of all tags in its interrogation zone in a periodic manner. This requirement not only increases workload of a reader but also generates large amount of traffic from a reader to a management server. In this paper, we propose a selective tag identification protocol which has two sub-frames: an identification frame and an access frame. The proposed protocol improves RFID tag identification performance by introducing a Frame Counter that enables readers to discriminate newly arriving tags and leaving tags. In addition, simulation results show that the proposed protocol reduces the amount of traffic from a reader to a management server by updating only the changes in tag population.

1. Introduction

Radio frequency identification (RFID) is a method to retrieve and store data using a radio channel. A RFID system consists of many tags and a reader. A tag consists of radio-frequency circuits, a CPU, and a small memory. By means of wireless communication between a reader and tags, RFID systems let us track or manage objects in real-time. In addition, passive tags are low cost allowing item-level tagging in retail markets or logistics businesses. As the demand for item-level object managements and tracking increases, RFID systems have gained more attention from such businesses as retail markets and courier services.

One of the most challenging issues in RFID systems is the tag collision problem. A passive RFID system uses only one RF channel and thus it is impossible to avoid collisions during communication between a reader and multiple tags. Tags only listen and respond to requests from the reader and do not interact with the one another [10]. Several collision resolution protocols have been proposed for RFID systems, such as Binary Tree protocol [3, 5, 7], Binary Tree with Bin Slot [4], Tree Slotted ALOHA [1], and Adaptive Binary Splitting (ABS) protocol [8]. These protocols use a similar principle but perform differently under different situations.

The binary tree protocol divides the colliding tags into two groups until only one tag remains [3, 5, 7]. The binary tree protocol is divided into two approaches: a *deterministic* one and a *probabilistic* one. The deterministic binary protocol intersects the tag population using tag ID, while the probabilistic binary tree protocol intersects by having tags choose a number between 0 and 1 randomly.

The RFID standard EPC Class 1 Generation 1 is based on the binary tree protocol and adopts 8 bin slots to reduce collision probability among tags [4]. Here, tags determine their own bin slot using their tag ID and can only access their own bin slot. If only one tag hits the specific bin slot, the tag is identified. However, selection of the appropriate number of bin slots is a challenging problem. If the number of bin slots is far smaller than the tag population, collision spreading effect degrades. On the other hand, if the number is far bigger than the tag population, bin slot hit ratio decreases, resulting in more delay. The protocol has to estimate the tag population to choose the optimal number of bin slots.

Tree slotted ALOHA [1] and Adaptive Binary Splitting (ABS) protocol [8] combine the collision spreading technique and the time slot allocation to improve the collision resolution performance. ABS protocol con-

1-4244-0697-8/07/.00

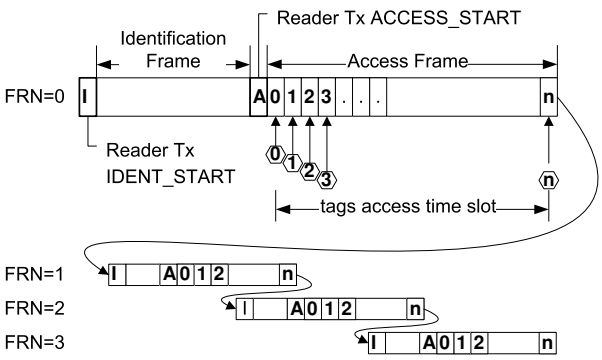


Figure 1. Frame structure of the proposed collision resolution protocol.

siders tag mobility. In this protocol, newly arriving tags choose time slot number between 0 and a maximum slot number that is previously determined. This approach intentionally generates collision between arriving and staying tags to trigger the binary tree collision resolution. However, intentionally generated collisions introduce communication overhead and are costly.

Our goal is to develop a fast and reliable collision resolution protocol that is suitable for mobile tag environments such as logistics and retail markets where both item-level tagging and tight tag tracking are required. In such environments, tag identification procedure should be performed periodically to determine which tags are moved into the reader's interrogation zone or moved out to another reader's interrogation zone. Also, the procedure should be completed within a guaranteed time interval so that the next identification period can start on time. Thus, to guarantee the reliability of the periodic identification, it is required to minimize or avoid nondeterministic aspects of collision resolution protocol in terms of identification time.

We propose a new protocol that considers tag mobility to improve the identification performance in mobile tag environments. Many tag identification protocols have been proposed to improve the performance in terms of the identification speed and the number of iterations to identify tags[1, 4, 6, 8]. However, many of those protocols are designed to identify all tags from scratch whenever the reader performs identification rather than identifying only the changes in tag population resulted from tag mobility. A few protocols considering tag mobility still have an overhead problem that needs to be eliminated for better performance. The proposed protocol introduces a new frame structure to improve the tag identification performance in a mobile tag environment.

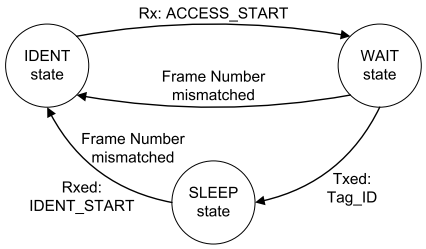


Figure 2. Tag state machine of the proposed collision resolution protocol.

This paper is organized as follows. Section 2 introduces the proposed framed collision resolution protocol. We show the simulation model in section 3 and present the performance analysis in section 4. Finally, section 5 summarizes our study and findings.

2. Framed collision resolution protocol

In mobile tag environments, tags frequently move in and out of a reader's interrogation zone. The tag mobility significantly reduces the performance of a collision resolution protocol in that the newly arrived tags contribute to more collision occurrences. Conventional collision resolution protocols [1, 4, 5, 8] do not consider tag mobility and thus experience performance degradation in mobile tag environments. The proposed protocol, on the other hand, selectively identifies newly arriving tags in a reader's interrogation zone by introducing a frame structure.

The frame structure of our proposed protocol (Figure 1) is designed to separate the collision resolution for newly arrived tags from the time slot access of previously identified tags. The frame is composed of two sub-frames: an identification frame and an access frame. The identification frame is used for identifying newly joined tags which are unknown to a reader. On the other hand, the access frame is for a reader to recognize leaving tags which are moving out of a reader's interrogation zone.

The proposed protocol selectively identifies newly arrived tags by updating and comparing a frame counter. The reader and tag counters are summarized in Tables 1 and 2. Each tag holds a frame counter (FC), and updates it whenever a new identification frame starts. To start the identification frame, the reader broadcasts the IDENT_START message with FRN (the current frame number) and TAG_POP (the number of identified tags during the previous frame). Each tag compares the FRN value from the IDENT_START message with its own

Table 1. Counters on reader

Name	Description
FRN	counting frame number
TAG_POP	counting & storing tag population
NTC	counting num. of newly identified tags

Table 2. Counters on tag

Name	Description
SC	counting ongoing time slot in access frame
FC	counting frame number
TS	store time slot assigned during ident. frame
NTC	counting number of new tags

FC value. Only the tags that have the same FC value as the reader's FRN can communicate with the reader. All the newly joined tags which have different FC are scheduled to participate in the next identification frame by setting their state as IDENT.

During the identification frame, the reader communicates with the tags to identify them. After the identification frame is completed, all tags have a unique Time Slot (TS) value from which each tag determines an accessible time slot. Once an access frame starts, time synchronized tags respond only to their time slot.

Tag operation follows the state machine depicted in Figure 2. Our tag state machine is composed of three states: IDENT, WAIT, and SLEEP state. All the tags in the IDENT state participate in the collision resolution procedure by communicating with the reader. All the tags identified during the previous identification frame or newly identified tags during the current identification frame move to the WAIT state and wait until the access frame starts. Once a tag responds to its own time slot, the tag sets its state to SLEEP and hibernates until receiving IDENT_START message. Tag operation on each of the three states is described in the next section.

Algorithm 1 Reader operation

```
1:  FRN, TAG_POP ← 0
2:  while TRUE do
3:      /* Identification Frame */
4:      TS, NTC ← 0
5:      Tx IDNENT_START
6:      while TS ≥ 0 do
7:          Collects responds from tags
8:          Check collision
9:          if no responds then
10:             TS ← TS − 1
11:             Tx NO_RESPONSE feedback
12:         else if collision then
13:             TS ← TS + 1
14:             Transmit COLLISION feedback
15:         else if one tag responds then
16:             NTC ← NTC + 1
17:             Identify the tag
18:             Tx NO_COLLISION feedback
19:         end if
20:     end while
21:
22:     /* Access Frame */
23:     TS ← 0
24:     Tx ACCESS_START with FRN and NTC
25:     while TS < (TAG_POP + NTS) do
26:         Wait tag response
27:         Check collision
28:         if no response then
29:             TAG_POP ← TAG_POP − 1
30:             Transmit NO_RESPONSE feedback
31:         else if one tag response then
32:             Transmit NO_COLLISION feedback
33:         end if
34:         TS ← TS + 1
35:     end while
36:     FRN ← FRN + 1
37:     TAG_POP ← TAG_POP + NTC
38: end while
```

2.1. Reader operation

Our reader operation is divided into an identification frame and an access frame as described in Algorithm 1. We use a probabilistic binary tree protocol for tag identification [3, 8]. To start the identification, the reader broadcasts the IDENT_START message with FRN and TAG_POP value. In response to the reader's IDENT_START message, tags respond with their TAG_ID. If more than one tag responds, the reader detects a collision. To inform of a collision, the reader sends out a COLLISION feedback message. If only one tag responds to the reader's feedback, the reader identifies the tag. Until all tags are identified, the reader repeats the identification procedure. After the identification frame is done, the reader starts an access frame by sending the ACCESS_START message with FRN and NTC (the number of new tags identified during the current identification frame). Access frame operation is similar to the Tree Slotted ALOHA protocol [1] and ABS protocol [8] in that our proposed protocol detects move-out tags through the access frame. If the reader finds move-out tag, the reader transmits a NO_RESPONSE feedback message to inform a time slot vacancy to the tags and thus the vacancy is filled with another tag in the next access frame. Tag operations for the reader's feedback messages are described through the next sections.

Algorithm 2 Main tag operation

```
1:  if Power-on by a reader then
2:      Wait Reader's IDENT_START message
3:      Read FRN and TAG_POP from message
4:      NTC ← 0
5:
6:      /* Determine State */
7:      if FRN = 0 then
8:          TS, FC ← 0
9:          state ← IDENT
10:     else if FRN > 0 then
11:         FC ← FC + 1
12:         if FC = FRN then
13:             state ← WAIT
14:         else
15:             TS ← 0
16:             FC ← FRN
17:             state ← IDENT
18:         end if
19:     end if
20:
21:     /* Tag Operation State Machine */
22:     while Power On do
23:         /* State Specific Operation */
24:         if state = IDENT then
25:             Run IDENT state tag operation
26:         else if state = WAIT then
27:             Run WAIT state tag operation
28:         else if state = SLEEP then
29:             Run SLEEP state operation
30:         end if
31:     end while
32: end if
```

2.2. Tag operation: main

Algorithm 2 represents the main tag operation. Once a tag is powered on by entering a reader's interrogation zone, the tag operation starts in the IDENT state waiting for the IDENT_START message from the reader. The IDENT_START message carries FRN and TAG_POP. All tags that receive the IDENT_START message with the FRN value of 0, reset TS and FC to 0, and set their state to IDENT. This makes all tags participate in identification procedure. If FRN is larger than 0, each tag increases FC by one and compares the value with FRN.

We assume that each different reader does not have same FRN value at the same time (Section 3.1). Based on this assumption, matched FC and FRN means that the tag is an original resident of the reader. Such tags enter into the WAIT state and wait until the access frame starts. Mismatched FC and FRN means that the tag is a newly arrived tag that has moved from another reader's interrogation zone. The newly arrived tags set their state to IDENT and participate in the collision resolution procedure by performing the IDENT state tag operation. Once all new tags are identified, they have state value of WAIT. During the access frame, tags in the WAIT state respond to their unique time slot and set their state to SLEEP. Tags in the SLEEP state remains quiet until a new frame starts.

2.3. Tag operation: IDENT state

Algorithm 3 describes the tag operation in the IDENT state.

Algorithm 3 IDENT state tag operation

```
1:  while state = IDENT do
2:      if FC! = FRN then
3:          TS ← 0
4:          FC ← FRN
5:          state ← IDENT
6:      else
7:          if TS = 0 then
8:              Transmit TAG_ID
9:          else
10:             state ← WAIT
11:             break
12:         end if
13:     end if
14:
15:     Wait Reader's feedback
16:     if feedback = COLLISION then
17:         if TS = 0 then
18:             bin ← random binary number
19:             TS ← TS + bin
20:         else
21:             TS ← TS + 1
22:         end if
23:     else if feedback = NO_COLLISION then
24:         if TS = 0 then
25:             TS ← TAG_POP + NTC
26:             state ← WAIT
27:         else
28:             TS ← MAX(TS − 1, 0)
29:         end if
30:         NTC ← NTC + 1
31:     end if
32: end while
```

Tags in the IDENT state communicate with the reader in order to be identified and assigned a unique time slot by the reader. Tags set their states to IDENT in the following two cases. Firstly, when a reader is boot up initially, the reader does not have any tags identified and thus all the tags have to be identified. In this

case, the reader transmits the IDENT_START message with FRN=0, and the tags which receive this message set their states to IDENT to participate in identification frame. Secondly, newly arrived tags set their state to the IDENT state. In the previous section, we assume that the FC value of the tag moved from another reader's interrogation zone is always different from the current reader's FRN. So the newly arrived tags with mismatched FC and FRN values are move to the IDENT state. During the identification frame, we use the probabilistic binary tree algorithm for collision resolution which is described in [8].

Algorithm 3, line 16 to 31 describes a tag operation for the binary tree protocol. When the identification frame starts, all unidentified tags are in the IDENT state with TS=0. Only tags with TS=0 transmit their Tag_ID to the reader from which the reader determines collision. If collision occurs, the reader broadcasts a COLLISION feedback message. Colliding tags that receive the COLLISION feedback add a random binary number to their TS and the other tags that do not participate in collision increases TS by 1. If only one tag responds, reader identifies the tag and transmits NO_COLLISOIN feedback. Algorithm 3, line 23 to 31 represents the tag operation on receiving a NO_COLLISION message. Regarding the NO_COLLISION message, the identified tag sets its TS value to (TAG_POP+NTC), where TAG_POP is the number of all tags under the reader's interrogation zone in the previous frame, and NTC is the number of newly identified tags during this identification frame.

The state of the identified tag is set to WAIT, and escapes from the IDENT state. On the other hand, tags in the IDENT state that receive the NO_COLLISION message and have TS values of larger than 0 decrease their TS value by 1. After decrement, one or more tags will transmit their TAG_ID if their decreased TS values are equal to 0. Every tag response is triggered by a reader's COLLISION or NO_COLLISION feedback messages except in the beginning of the identification frame where the IDENT_START message triggers the tag response. If there is no more tag response, the reader completes the identification frame. The identification cycle is repeated until all the new tags are identified. The identification frame is completed once all the new tags are identified. After the identification frame is finished, each tags is assigned the TS value between 0 and (TAG_POP+NTC-1). Tags having TS value between 0 and (TAG_POP-1) are those identified previously. On the other hand, tags having TS value between TAG_POP and (NTC-1) are newly identified tags.

2.4. Tag operation: WAIT state

Algorithm 4 describes the tag operation when the tag is in the WAIT state. Access frame has (TAG_POP-1) time slots. At every time slot, only one tag can respond and thus no collision happens in the access frame because newly arrived tags are separately identified in the identification frame.

Algorithm 4 WAIT state tag operation

```
1:  Wait Reader's ACCESS_START message
2:  Read FRN from the message
3:  SC ← 0
4:
5:  while state = WAIT do
6:      if FC! = FRN then
7:          TS ← 0
8:          FC ← FRN
9:          state ← IDENT
10:     else if TS = SC then
11:         Transmit TAG_ID
12:     end if
13:
14:     Wait Reader's feedback
15:     if feedback = NO_COLLISION then
16:         if TS = SC then
17:             state ← SLEEP
18:         else
19:             SC ← SC + 1
20:         end if
21:     else if feedback = NO_RESPONSE then
22:         if NTC > 0 then
23:             NTC ← MAX(NTC − 1, 0)
24:             if TS = TAG_POP then
25:                 TS ← SC
26:             end if
27:         else
28:             TS ← MAX(TS − 1, 0)
29:         end if
30:     end if
31: end while
```

During the WAIT state, the tag state machine waits until the tag receives ACCESS_START message and does not respond to reader's feedback messages. Once the tags receive the ACCESS_START message tag state machine react to the reader's messages. The tags receiving NO_COLLISION feedback from the reader increase their slot counter by 1 and compare the value with their TS value. If the values are matched, the tag responds to the time slot with its TAG_ID and set their state to SLEEP. On the other hand, if a tag moved out of the reader's interrogation zone, there is no tag response in a time slot which is assigned to the tag.

Algorithm 5 SLEEP state tag operation

```
1:  while state = SLEEP do
2:      Wait Reader's IDENT_START message
3:      FC ← FC + 1
4:      if FC! = FRN then
5:          TS ← 0
6:          FC ← FRN
7:          state ← IDENT
8:      else
9:          state ← WAIT
10:     end if
11: end while
```

Thus, the reader does not receive any tag response. Once a reader detects the absence of a tag, the reader sends out NO_RESPONSE feedback. In response to the NO_RESPONSE feedback, the tag that is identified and assigned to the last time slot during the identification frame is assigned to the empty slot to fill the vacancy. If there is no newly identified tag any more, all the tags still in the WAIT state decrease their TS by 1 to fill the empty slot. Algorithm 4, lines 21 to 30 show the tag operation when the reader detects the absence of a tag.

2.5. Tag operation: SLEEP state

Algorithm 5 shows the SLEEP state tag operation. In the SLEEP state, tags do not answer to the reader's feedback until they receive the IDENT_START message. A tag receiving IDENT_START message increases FC by 1 and compares the value with the FRN value obtained from the IDENT_START message. If the values match, the tag waits ACCESS_START message in the WAIT state. If the values mismatch, the tag places itself in the IDENT state and follows the identification procedure.

3. Simulation

We implemented the ABS protocol and our proposed collision resolution protocol to illustrate the effect of tag movements on the protocol performance. The implementations are written in plain C language, and compiled using GCC (GNU Compiler Collection) on Cygwin™environment. Simulation scenarios are designed to emulate the inventory and retail market environments. In addition, we made assumptions to simplify the simulation model. The following sections describe our assumptions, tag mobility model, simulation parameters, and simulation scenarios.

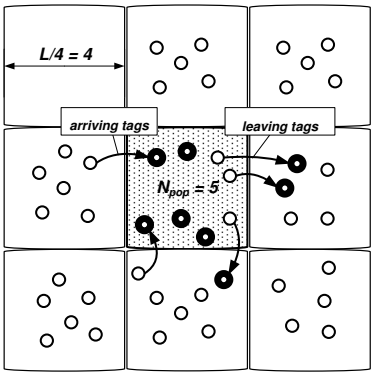


Figure 3. Tag mobility model.

3.1. Assumptions

To make the simulation more tractable, we made assumptions for the reader model, and the tag mobility model. The following describes these assumptions.

- *Assumption I*: Reader's interrogation zones are in rectangular shapes.
- *Assumption II*: There is no reader to reader collision or interrogation zone overlap. Thus, all tags can be a member of only one reader at a time.
- *Assumption III*: All neighboring readers do not have the same FRN at the same time. So tags residing in different reader's interrogation zone cannot have the same FC values at the same time.

3.2. Tag mobility model

To calculate the population changes in an interrogation zone, we define the tag mobility model. The population and movement of tags are important factors that affect the performance of the protocol. Our tag mobility model is presented in Figure 3. In this example (the interrogating reader placed at the center), N_{in}=*2*, N_{out}=*3*, and the perimeter of the interrogation zone, *L=16*. After an identification frame is completed, N_{pop} of the center reader should be 5.

In addition, we define two tag mobility patterns: one for an inventory like case, and the other for retail market like case. For the inventory like case, a number of tags already resides (and thus, have been identified) in a reader's interrogation zone, and some tags are simultaneously moving in or moving out of the reader's interrogation zone. This pattern is used for Scenario I and II which are described in section 3.4. For the retail market like case, the mobility pattern is different from the

inventory model in that each tag takes a random walk in the specific area. Based on Assumption I, we introduce a fluid-flow model [2, 9] for our tag mobility model. In addition to Assumption I, we assume that tags are uniformly distributed with a density of ρ, and walk randomly from one interrogation zone to another with an average velocity of ν. Now, the boundary crossing rate, *C*, is given by

$$C = \frac{\rho \nu L}{\pi} \quad (1)$$

This fluid-flow model has been widely used to model the mobility of cellular phone users [9]. We found that this model is applicable to the tag mobility model in RFID networks where the uniformly distributed tags are crossing the readers' interrogation zones. In our simulation, we used this mobility model for Scenario III which is described in Section 3.4 in detail.

3.3. Parameters

The following parameters are used for simulation.

- *Tag population density (ρ)*: Tags are uniformly distributed in an area. The population of tag is determined by the density, ρ.
- *Tag moving speed (ν)*: Tag moving speed determines the number of tags moving in or out of the interrogation zone.
- *Interrogation zone perimeter (L)*: The perimeter of the interrogation zone.
- *Tag population (N_{pop})*: The number of tags recognized by a reader in the previous frame. As N_{pop} increases, a reader takes more time to recognize the tags.
- *The number of tags moved in (N_{in})*: The number of tags moved into the reader's interrogation zone from the other reader. As N_{in} increases, probability of tag collision increases.
- *The number of tags moved out (N_{out})*: The number of tags moved out of the current reader. As N_{out} increases, probability of tag collision decreases.

3.4. Scenarios

Simulation is performed for three different scenarios. Scenario I and II are designed to emulate inventory like environments that has a few hundreds or thousands of tagged items in stock (for example, N_{pop} of 200 to 600 tags), and multiple readers identify and track the tags, moving in and out. Scenario III simulates the situation where uniformly distributed tags in an area are moving from one interrogation zone to another as in a retail market. In retail markets, customers carry the tagged items and their movements resemble a random walk in any direction.

Table 3. Scenario I

Scenario I	sim. 1	sim. 2	sim. 3
N_{pop}	200	400	600
N_{in}	0∼200	0∼200	0∼200
N_{out}	0	0	0

In Scenario I, N_{in} tags are moving into a zone with no move-out tag ($N_{out} = 0$). In scenario II, we fixed the tag population N_{pop}, while, N_{in} new tags are moving in, and N_{out} tags are moving out simultaneously. Thus the reader has to detect moving in tags and moving out tags. Simulation parameters for Scenario I and II are listed on Table 3 and 4, respectively.

Table 4. Scenario II

Scenario II	sim. 1	sim. 2	sim. 3
N_{pop}	500	500	500
N_{in}	10∼100	10∼100	10∼100
N_{out}	50	100	150

In Scenario III, we consider retail market case where uniformly distributed tagged products are moving randomly with an average velocity, $\nu = 0.3$ m/sec. Based on Figure 3, we assume a retail market has 9 readers, and each total tag populations in this area are assumed to be 3600, 4500, and 5400. In addition, 9 readers have a rectangular shape coverage with perimeter, $L = 16$ meters. Using Equation (1) for the fluid-flow mobility model, we calculate the tags' boundary crossing rate, C, for each simulation setups. Crossing rate, C represents the number of tags move in and out of a reader's interrogation zone. Thus, in this scenario, the number of move-in tags and move-out tags are same.

For example, consider we have total tag population of 3600 tags (400 tags per reader) in the 12x12 square meter area, 20% (=720) of tags among the total tag population are moving with an average speed of 0.3 meter per second, and the boundary length is 16 meters. Now, from the Equation (1), tags' boundary crossing rate, $C = \frac{5 \times 0.3 \times 16}{\pi} = 7.64$. If we assume that a reader runs to identify tags in every 5 seconds, approximately 38 tags cross the boundary of reader's interrogation zone

Table 5. Scenario III

Scenario III $\nu = 0.3m/sec$ $L = 16m$	sim. 1	sim.2	sim. 3
total N_{pop} (tags)	3,600	4,500	5,400
on moving (20%) (tags)	720	900	1,080
ρ (tags/m^2)	5	6.25	7.5
C (tags/sec)	7.64	9.55	11.46
C_{run} (tags/run)	≈38	≈48	≈58

during the 5 seconds interval. Table 5 specifies a simulation parameters for Scenario III.

4. Performance

This section defines the performance metrics that we used for analysis of the simulation results and shows performance analysis for each scenario.

4.1. Performance metric

We define the following two performance metrics used to analyze the performance of the protocol.

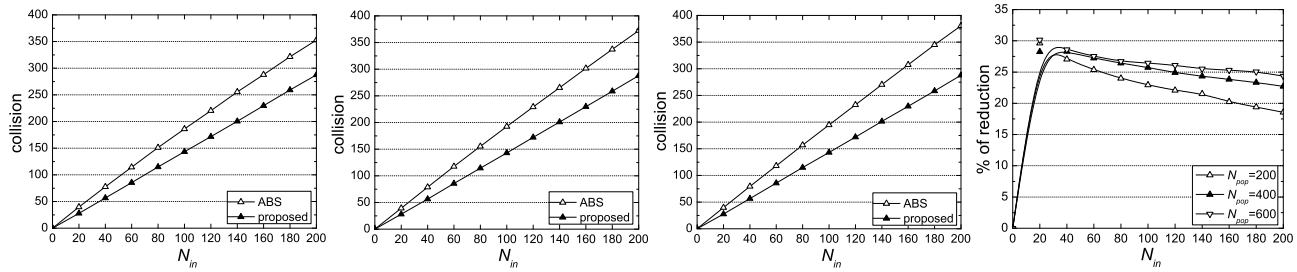
- *The number of collisions*: To identify the tags, protocol resolves the collision. The number of collisions that occur during the identification is a good measurement to understand the performance of the collision resolution protocol. In general, better protocol experiences lower collisions during tag identification.
- *The number of iterations*: Total number of iterations is defined as the total number of feedbacks sent from the reader plus the number of time slots in the access frame. This metric determines the total delay to identify all tags in the reader's interrogation zone.

4.2. Analysis

Analysis for simulation results is based on the performance metrics shown in the previous section. To obtain the desirable accuracy, we performed each scenarios for 1,000 times. Figure 4 shows the simulation results for Scenario I. As given in the Table 3, we consider that 0 to 200 new tags are moving into the reader's interrogation zone (N_{in}=0∼200) without move-out tags (N_{out}=0). In the previous identification period, the reader has identified N_{pop} tags, and tries to start a new identification period. During this identification period, the reader should identify newly arrived N_{in} tags. Figure 4 shows the performance of the protocol in terms of the number of collision occurred during the identification period. Figure 4(a), 4(b), and 4(c) show that if there's no move-in or move-out tag (N_{in}=N_{out}=0), the proposed protocol and ABS does not experience collision but collision occurs if there are move-in tags. As the move-in tags increases, the proposed protocol shows better performance than ABS protocol. Moreover, as the tag population in a reader's interrogation zone increases, the performance gap also increases as shown in Figure 4(a), 4(b), and 4(c). The percentage of collision reduction achieved by the proposed protocol is shown in Figure 4(d). The results show that if a reader has N_{pop} of 600, and N_{in} of 200, the proposed protocol experiences about 24% less collision than ABS protocol. When the new tags are coming in, the proposed protocol successfully put the new tags into the identification frame, rather than intentionally generating collision to trigger the identification mechanism. Figure 5 shows the performance of the protocol in terms of the number of total iterations to identify tags. The number of iterations can be directly interpreted as the total time delay to identify all tags. The results show that the proposed protocol spends about 13% less time to identify tags than ABS protocol when a reader has N_{pop} of 600, and 200 tags come into the reader's zone. In summary, the simulation results demonstrate that the proposed protocol not only reduces collisions and but also requires less number of iterations.

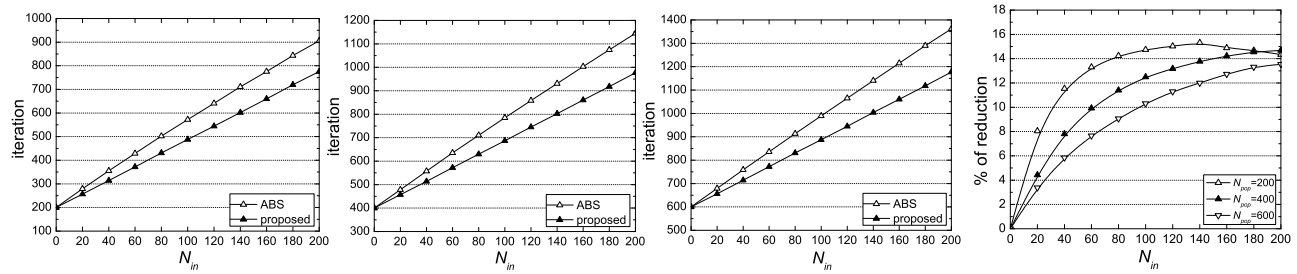
Scenario II simulates the situation that some tags are coming in, and some tags are going out simultaneously. Figure 6 shows the simulation results for Scenario II. In Figure 6(a), a reader has 500 tags identified. In this case, if 50 tags are going out (N_{out}=50) and 30 tags are coming in (N_{in}=30) at the same time, the proposed protocol generates about 20% less collisions than ABS protocol. However, it is observed that the reduction of the total number of iterations achieved by the proposed protocol is around 3% to 8% throughout the simulations for Scenario II (Figure 5(d)). For ABS protocol, some of empty slots generated by leaving tags may be assigned to some of the arriving tags for the identification while the proposed protocol does not use those empty slots for tag identification.

Scenario III simulates the tracking of the tagged items in retail market like environments. Simulation for Scenario III utilizes the fluid-flow model to quantify the tags' movements as shown in Table 5. Intuitively, the number of tags that cross a interrogation zone bound-



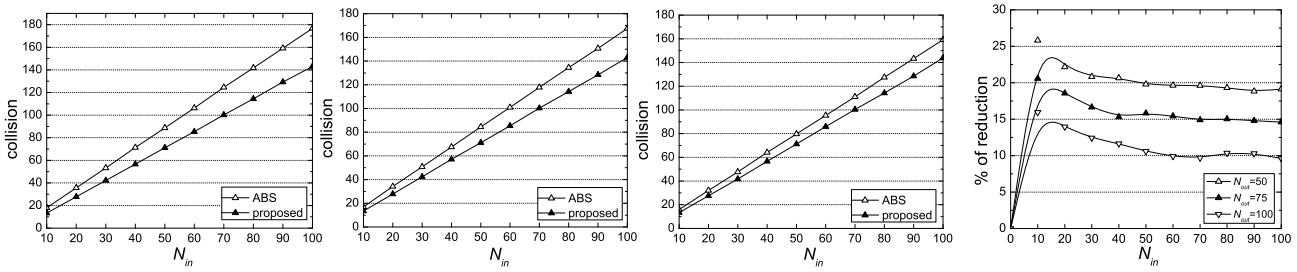
(a) $N_{pop} = 200$ (b) $N_{pop} = 400$ (c) $N_{pop} = 600$ (d) % of collision reduction

Figure 4. Scenario I: Comparison of the number of collisions for different N_{pop} and N_{in} values.



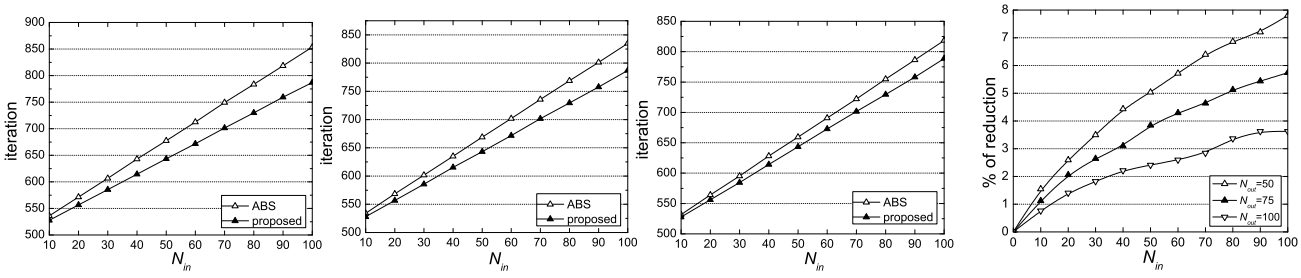
(a) $N_{pop} = 200$ (b) $N_{pop} = 400$ (c) $N_{pop} = 600$ (d) % of iteration reduction

Figure 5. Scenario I: The number of required iterations for different N_{pop} and N_{in} values.



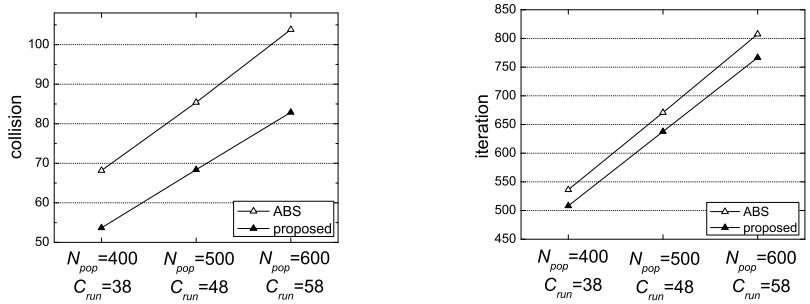
(a) $N_{out} = 50$ (b) $N_{out} = 75$ (c) $N_{out} = 100$ (d) % of collision reduction

Figure 6. Scenario II: The number of collision occurrences for different N_{out} and N_{in} values.



(a) $N_{out} = 50$ (b) $N_{out} = 75$ (c) $N_{out} = 100$ (d) % of iteration reduction

Figure 7. Scenario II: The number of required iterations for different N_{out} and N_{in} values.



(a) number of collision (b) number of required iteration

Figure 8. Scenario III: Comparison of the number of collision occurrences and iterations.

ary are positive proportional to the tag population in the area. In this simulation, every reader operates identification in every 5 seconds to track the movements of the tags. Figure 8 shows the number of collision occurrences and required iterations for three different boundary crossing rates of 38, 48, and 58 crossing tags per 5 seconds.

5. Conclusion

The performance of collision resolution protocol has been one of the most challenging issues in RFID system related research areas. However, most research performed on the collision resolution protocols does not consider tag mobility. This paper focuses on the performance of collision resolution protocol for mobile tag environments. The proposed protocol separates the tag's channel access from the collision resolution which always involves tag to tag collision. Simulation results from the different scenarios demonstrate the performance of the proposed protocol. Also, this paper introduces the tag mobility model to quantify the movements of the tags. With this model, it is possible to expect and simulate the work load applied to the reader. For the future work, we are working on collision resolution technique that dynamically spreads the collisions based on the estimation of tag population.

References

[1] M. A. Bonuccelli, F. Lonetti, and F. Martelli. Tree slotted aloha: a new protocol for tag identification in rfid networks. *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 603–608, 2006.

[2] T. X. Brown and S. Mohan. A mobility management strategy for personal communications systems. *IEEE Transactions on Vehicular Technology*, 46(2):269–278, 1988.

[3] J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transaction on Information Theory*, IT-25(5):505–515, September 1979.

[4] EPCglobal. Epc radio frequency identity protocols class 1 generation 2 uhf rfid protocol for communications at 860-960 mhz. 2004.

[5] K. Finkenzeller. *RFID Handbook*. "John Wiley&Sons", New York, 1999.

[6] D. Hush and C. Wood. Analysis of tree algorithms for rfid arbitration. *IEEE International Symposium on Information Theory*, page 107, August 1998.

[7] J. L. Massey. Collision-resolution algorithms and random-access communications. Technical report, Apr. 1980.

[8] J. Myung and W. Lee. Adaptive binary splitting: a rfid tag collision arbitration protocol for tag identification. *Mobile Networks and Applications*, 11(5):711–722, 2006.

[9] R. Thomas, H. Gilbert, and G. Mazziotto. Influence of the movement of the mobile station on the performance of the radio cellular network. *Proc. 3rd Nordic Seminar*, September 1988.

[10] H. Vogt. Efficient object identification with passive rfid tags. pages 98–113, 2002.