A Middleware Framework for Efficient Identification of Tagged Objects

Byung-Kook Son¹, Chang-Soon Kim¹, Kyung-Lang Park¹, Cheng-hao Quan², and Shin-Dug Kim¹

¹Department of Computer Science
Yonsei University
Seoul, Republic of Korea
²Electronics and Telecommunications Research Institute
Seoul, Republic of Korea

Abstract: RFID is the technology that identifies tagged objects without line of sight. Tagged objects are identified by the readers that are managed by the RFID middleware. Therefore, the performance of identification process for tagged objects can be improved how to manage the readers. In this paper, we propose the middleware framework with an algorithm to manage multi-readers based on this middleware to identify tags efficiently. The proposed middleware assigns the role of readers based on three parameters, i.e., application requirements, the feedback information of readers, and the information of connected readers. To determine the role of readers, the proposed middleware uses the PEG (Pre-Estimation of Group) algorithm. We examine the performance of proposed middleware framework and PEG algorithm by using Gen2 simulator. As a result, when worst case is occurred in a particular group, the performance of tag identification per reader improves about 170%.

Key-Words: RFID, RFID Middleware, Multi-readers, Gen2 protocol, Fast identification system

1 Introduction

RFID is an important technology for ubiquitous computing environment. Recently, RFID technology has used widely in supply-chain management. Specially, because RFID can reduce the cost for product loss by checking of stocks and tracking product location, RFID will be in wide use in warehouse. RFID system consists of three parts, i.e., RFID application that use RFID data, middleware that processes RFID data and RFID reader which identifies tagged objects. To identify tagged objects, RFID reader and middleware are more important than other part.

However, most of RFID issues have been focused on how to process tagged objects, such as data filtering, the elimination of duplicated data, and data grouping [4, 5, 6]. And also in term of the reader, the role of readers is fixed in many cases. In other words, RFID middleware has just been focused on gathering RFID data from the readers. Also, RFID reader just executes already fixed role without any specific role assigned by dynamically from middleware. As a result, the number of tagged objects identified successfully tends to be reduced. Thus the capability of any reader cannot be utilized perfectly. Also, pervious RFID system does not support any efficient method that resolves tag collision when there exist many tags in the sensing range of any reader.

In this paper, we propose a RFID middleware framework for efficient identification of tagged object and PEG (Pre-Estimation of Group) algorithm based on the proposed RFID middleware. PEG algorithm assigns a specific role to each reader, based on the information about tag collision of each group received from the estimation reader, application requirement, and connected readers. Moreover, because PEG algorithm is based on Gen2 [1] protocol, it supports the method that reduces tag collision by dividing many tags into four small groups. Especially, when the worst case that a particular group has many tags, PEG algorithm can resolve this problem. As a result, the performance of identification for tagged objects improves more than the case where readers execute fixed role.

We examine the performance of proposed middleware framework and PEG algorithm by using Gen2 simulator. As a result, when worst case is occurred in a particular group, performance of tag identification per reader improves about 170%.

Rest of the paper is organized as follows: In Section 2, we introduce the Gen2 protocol. Section 3 describes our proposed RFID middleware framework and Section 4 explains the PEG algorithm to identify
tags efficiently. Section 5 provides several experimental results by each algorithm. Finally, we conclude in Section 6.

2 Gen2 protocol
Gen2 protocol [1] is proposed by EPCglobal [7] as a standard protocol for 860 ~ 960 MHz frequency range. Gen2 protocol has some advantages such as fast tag identification, bandwidth efficiency, reliability, reader range, security, and cost. Especially, Gen2 protocol supports the session that can manage a specific tag population. In addition, it is very important factor to the multi-readers operation.

Gen2 protocol has four sessions, denoted S0, S1, S2, and S3 respectively. Each tag has only one session during the inventory round. And the tag communicates with the reader which has the same session. If the tag receives any command from the reader that has different session, the tag ignores the received command. Each session has the inventory flag which has either A or B. When inventory round is beginning, the reader chooses one of the four sessions and sets the inventoried flag as either A or B. The identified tag changes this inventoried flag in session into inverse (A->B, B->A). Therefore, the identified tag dos not response to the same session.

Gen2 protocol has a select command that selects a particular tag population. In addition, it has nine parameters. Before sending the select command, the reader sets the value of the parameters. At this time, the reader sets the parameters to select a particular population. Especially, to select a particular population, the reader has to set the session and mask of the parameters.

3 Middleware frameworks
In this chapter, we describe our middleware framework. The middleware is designed to configure roles of connected readers dynamically with considering application requirements, feedback from readers, and reader information. Thus it consists of task manager, reader event manager, multi-reader manager, and reader information repository. We do not depict common components in order to focus on multi-reader management.

Figure 1 shows our middleware framework and its operation flow. When an application submits a task specification, task manager creates an instance for the application.

An application requirement can be represented in various types, but in this paper, we only focus on fast tag identification. Task manager delivers application requirement to multi-reader manager. Reader event manager delivers the feedback information of readers to the multi-readers manager. And reader information repository gives reader information to multi-reader manager. The reader information repository manages the connected reader information that includes the number of readers, the position of readers, and the sensing range of readers.

Then, multi-reader manager determines the role of readers, generates commands, and sends commands to each reader. Reader operates based on given commands and gives some feedback to multi-reader manager. Based on feedbacks, multi-reader manager reconfigures the role of readers.

4 Efficient identification algorithms
The algorithm to manage multi-readers can be changed by how readers are configured. This algorithm to support multi-reader may cause an effect on the performance of tagged object identification process. In this paper, a case where many readers are allocated linearly without any overlapped sensing.
range of readers is considered to detect many tagged objects fast. In this case, the performance of tagged object identification process can be improved by the proposed algorithm to manage multi-readers. First of all, we examine a single method that just performs partitioning of tags without any algorithm. Then, we examine another method by using backup readers. Finally, we design an algorithm by using an estimation reader that solves the problem caused by algorithm by using backup readers. We use the Gen2 simulator to analyze these algorithms.

4.1 Linear ranges of readers
Partitioning of tags can improve the performance of tagged object identification process by reducing tag collision and the value of Q where is slot-count parameter. Generally, because Gen2 protocol supports four sessions, tags can be divided four groups. Therefore, each group is identified by four readers. If the total number of tags can be identified by using four readers, and each group has the same number of tags, all of tags can be identified. However, because we cannot know any information about the tags before identifying tags, it is impossible that each group can have the same number of tags. That is, a particular group may have many tags than other groups. This case can be considered as the worst case. As a result, a reader in charge of a particular group cannot completely identify all the tags.

4.2 SWF (Success-Well-Fail) algorithm
If each of groups has the same number of tags, there is no problem in the identification process even though any particular algorithm is not used. However, if the worst case is occurred in a particular group, the identification process of tagged objects cannot be completed successfully. To solve this problem, some feedback information of readers can be represented as “SUCCESS”, “FAIL”, and “WELL”. When a reader finishes the identification process of tags completely, it sends “SUCCESS” message. When a reader identifies some given tags, but does not finish it completely, it sends “WELL” message. When it finds out that given tags are too many to identify, it sends “FAIL” message to the multi-reader manager.

Multi-reader manager assigns additional commands to the backup reader based on this feedback information. When multi-reader manager receives “SUCCESS”, multi-reader manager does nothing. When multi-reader manager receives “WELL”, multi-reader manager assigns commands to identify the rest of tags. When multi-reader manager receives “FAIL”, multi-reader manager divides the tags in two groups and assigns two backup readers to each of them.

For instance, if there are four readers such as reader-1, reader-2, reader-3, and reader-4, each reader has its own mask such as, “00”, “01”, “10”, and “11” respectively. Reader-1 identifies tags which have postfix “00”. Reader-2 identifies tags which have postfix “01”. Reader-3 identifies tags which have postfix “10”. Reader-4 identifies tags which have postfix “11”. Suppose that the number of tags with postfix “00” is 200, the number of tags with postfix “01” is 50, the number of tags with postfix “10” is 100, and the number of tags with postfix “11” is 50. Then reader-1 will send “FAIL” message. Reader-2 will send “SUCCESS” message. Reader-3 will send “WELL” message. Reader-4 will send “SUCCESS” message. Multi-reader manager assign the commands to identify the rest of tags. Two backup readers can be assigned to identify the tags unidentifed by reader-1. One backup reader identifies the tags unidentified by reader-3.

Thus, using SWF algorithm with some backup readers, we can identify all the tags even though the worst case occurs. However, if “FAIL” message occurs a lot, we would need a number of backup readers.

4.3 PEG(Pre-Estimation of Group) algorithm

4.3.1 Gen2 protocol and tag partition
Because Gen2 protocol supports the session mechanism, tags can be partitioned by using this session mechanism. Tags can be divided into four populations by using four sessions. In addition to the four populations, we can use the parameters, such as session, inventoried flag, selected flag, and mask in the SELECT command.

![Fig. 2. Tag partition.](image-url)
are four readers. Each of the four readers has a session, denoted as S0, S1, S2, and S3. Also, each of the four readers has its own mask value which are 00, 01, 10, and 11 respectively. If the mask value matches with the tag value, the reader changes the inventoried flag of this tag as A. Otherwise the reader changes the inventoried flag as B. For example, reader-1 with session S0 assigns the session S0 to the group 1 that matches with the mask value. Then, reader-1 changes the inventoried flag of S0 into A, and other groups do not change the inventoried flag of S0. Reader-2, reader-3, and reader-4 can partition tags as the reader1.

4.3.2 Algorithm
SWF (Success Well Fail) algorithm does not know the size of each tag population. Therefore, the middleware has some difficulty to decide the role of the readers, considering whole tagged objects. For example, if a particular population has a few tags, the reader can finish the identification process early and eventually its capability is not utilized properly. So we need an efficient management method to support multi-reader environment.

Fig. 3. The algorithms by using estimation reader.

This method uses a pre-estimation reader, which is located at the staring position of all readers. This pre-estimation reader ties to get the information about each of tag populations and sends the information to the reader event manager. The information includes the number of collisions about each population. Therefore, the size of each population can be specified roughly.

Figure 3 shows the algorithm that is based on the pre-estimation reader. Pre-estimation reader executes the inventory round at a short time to obtain the information of four tag populations. This inventory round of the pre-estimation reader does not identify tags, but checks the frequency of collisions about each of four tag populations. After the pre-estimation reader checks the frequency of collisions about each tag population, this reader sends the result to the middleware. The middleware can determine the tag population for the largest number of collisions. That is, we can know that a particular tag population includes more tags than other populations. Also, the middleware can determine the tag population for the smallest number of collisions. Then the reader-1 is assigned as the role to detect the tag population having the largest number of collisions. And then Q-value, which is larger then others, is assigned to the reader-1 by the middleware to reduce tag collision. The reader-1 executes the identification process of the tagged objects until the tagged objects stay in the sensing range of reader-1. Then, the reader-1 sends the information to the middleware. Then the reader-2 is assigned as the role to detect the tag population for the smallest number of collisions by the middleware. The reader-2 sends its result to the middleware. The middleware reassigns the reader-2 to identify tags which cannot be identified by the reader-1. The middleware assigns the reader-3 and reader-4 as the same as the reader-2.

5 Experimental results
We consider two cases to analyze the performance of PEG algorithm. First, we consider the normal case that each group has the similar number of tags. Second, we consider the worst case that a particular group has more than 300 tags. We examine the performance of tag identification process by each algorithm. In the following graphs, normal is the method that does not use any algorithm for identifying tags.
Figure 4(a) and (b) show the number of readers to identify all tags perfectly by using each algorithm. When the number of tags is 400, any algorithm can identify all tags perfectly by using four readers. That is, there is no difference in the performance of tagged objects identification process. However, when the number of tags is more than 500, normal and SWF algorithms need the same number of readers to identify all tags perfectly. However, PEG algorithm reduces two readers to identify all tags perfectly. Therefore, we can know that the performance of tagged objects identification process improves by using PEG algorithm.

Figure 5 and Figure 6 show the number of identified tags per reader while tags stay in the sensing range of any reader. In these graphs, we can know that the number of identified tags per reader is almost equal regardless of any algorithm.

Figure 7(a) and (b) show the number of readers to identify all tags perfectly by using each algorithm when the worst case is occurred in a particular group. Unlike the normal case, there is difference in the performance of tagged object identification process by each algorithm.

Because the normal does not support the flexible role of each reader, it needs many readers. However, because PEG algorithm supports the flexible role of each reader, it can reduce the number of used readers.

Figure 8 and Figure 9 show the number of identified tags per reader in the worst case. We can know that each reader identifies the unequal number of tags in normal and SWF algorithms.
Fig. 8. Identified tags per readers by each algorithm when Q-value is 8 in worst case.

Fig. 9. Identified tags per readers by each algorithm when Q-value is 9 in worst case.

However, each reader identifies almost the equal number of tags in the PEG algorithm. In other words, the capability of any reader can be utilized perfectly in the PEG algorithm.

6 Conclusion

In this paper, we introduce the middleware framework with an algorithm to manage multi-readers based on this middleware to identify tags efficiently. To identify tags efficiently, RFID middleware has to assign and change the role of each reader dynamically. And each reader executes this role flexibly. To determine the role of each reader, this middleware uses three parameters, i.e., application requirements, the feedback information of readers, and the information of connected readers. In this paper, a case where many readers are allocated linearly without any overlapped sensing range of readers is considered to detect many tagged objects fast. Especially, when the worst case is occurred in a particular group, we propose the PEG algorithm that resolves this problem. This algorithm can reduce the number of readers to identify all tags perfectly and improve the number of identified tags per reader. In the event, the proposed middleware framework with an algorithm to manage multi-readers based on this middleware can improve the performance by about 170%.

References: