Adaptive Distributed Reader Activation Approach for Large-scale RFID Systems

Weiping Zhu*†, Yi Hong*, Vaskar Raychoudhury‡, Run Zhao§ and Dong Wang§
*International School of Software, Wuhan University, P. R. China
†State Key Lab. for Novel Software Technology, Nanjing University, P. R. China
wpzhu@whu.edu.cn, yihongucla@gmail.com
‡Dept. of E&CE, IIT Roorkee, India
vaskar@ieee.org
§School of Software, Shanghai Jiao Tong University, P. R. China
zhaorun@cs.sjtu.edu.cn, wangdong@sjtu.edu.cn

***Abstract*—In recent decades, a growing number of large-scale RFID systems are used in various applications. In such a system, it is not uncommon that multiple concurrent radio communications among RFID readers and tags cause serious inference (called collision in the RFID field). One important kind of method to achieve collision-free communication is to activate RFID readers in different time slots. Existing activation approaches for solving this problem are mainly centralized, which is impractical due to the lack of central server, one-point failure risk, and performance bottleneck. Some distributed algorithms are proposed recently, but failed to consider the adaptiveness of the identification, where all of the RFID readers need to participate in the coordination control even if they do not have communication requirements any more. As a result, the optimal identification performance cannot be achieved. In this paper, we propose an adaptive distributed reader activation approach called ADRA for large-scale RFID systems. We build a fine-grained conflict graph for different kinds of collisions. And then a shared permission based distributed approach is adopted to eliminate those collisions. We guarantee that the RFID readers that do not need to communicate any more are suspended and excluded from the execution of coordination eventually. Extensive simulation results show that our approach outperforms existing approaches in terms of execution time and message overhead.**

***Index Terms*—Adaptive Algorithm; Distributed Reader Activation; Large-scale RFID Systems**

I. Introduction

Radio Frequency Identification (RFID) is a rapidly developing digital identification technology that employs radio to collect identification information from RFID tags [1], [2]. In a typical RFID identification scenario, an RFID reader sends a request to RFID tags, and the RFID tags reply with the information pre-stored in their storages. In recent decades, many applications such as supply chain management, auto-ticking, human and animal tracking, smart hospital, etc. employ more and more RFID readers and form large-scale RFID systems [3]. Compared with classical RFID systems that include a limited number of readers, large-scale RFID systems greatly extend the coverage of system and increase the reliability of identification.

In large-scale RFID systems, it is not uncommon that multiple concurrent radio communications among RFID readers and tags cause serious inference (called collision in the RFID field). Therefore an important problem for large-scale RFID systems is how to eliminate such collisions to achieve correct identification. The collisions can be categorized into three types: tag-tag collision, reader-tag collision, and reader-reader collision [4]. Tag-tag collision denotes that two tags collide when they response to a reader's request simultaneously. Reader-reader collision denotes that, if two or more readers send the requests simultaneously, the tags in their overlapped interrogation region cannot distinguish the requests and cause communication failure. Reader-tag collision denotes that, if reader A resides in the interference region of reader B and both the readers send requests, all the data transmissions from reader B are interfered by the signal sent from reader A. tag-tag collision is usually addressed by anti-collision protocols in the MAC layer [5], [6], [7], while the other two collision need proper activation of readers in the application layer that is the focus of this paper.

Existing approaches on such topic [4], [8], [9], [10] mainly focus on the centralized optimization of RFID reader activation, where all the readers belong to one application and the purpose is to read all the tags only once. Although working well for the target scenario, these approaches encounter following new challenges in large-scale RFID systems. 1) There are usually multiple applications running on top of the RFID system, and a central server is usually unavailable for coordinating those applications. 2) Even if a central server is available, it suffers from one-point failure and computation bottleneck. Some recent works [11], [12] have proposed distributed approaches for activating RFID readers. However, they have not considered that different applications may need different RFID readers for identification. Moveover, these approaches are not adaptive. An algorithm is considered adaptive for our problem if there is no identification request from an RFID reader any more, after a period of time the reader has no need to participate in the execution of the

978-1-4673-9101-6/15 $31.00 © 2015 IEEE
DOI 10.1109/MASS.2015.11

activation algorithm. This promotes the fairness of RFID readers in the coordination of activation. Other advantages of adaptive approaches include reduced message overhead and reduced execution time.

In this paper, we propose the Adaptive Distributed Reader Activation Approach (ADRA) for large-scale RFID systems. We first build a conflict graph for RFID operations considering two kinds of RFID collisions, reader-tag collision and reader-reader collision. Based on it, a three-layer approach is proposed to achieve the collision-free activation of readers. The first layer eliminates reader-tag collision in individual applications, the second layer eliminates reader-tag collision among different applications, and the third layer eliminates reader-reader collision. Permission based mutual exclusion algorithm is used for the detailed implementation. A reader needs to obtain all the permissions of other readers that may cause collisions before it can perform the identification. Permissions is properly allocated among RFID readers for achieving adaptiveness. Extensive simulations are carried out for validating the proposed approach. The results show that our approach outperforms existing approaches. In summary, this paper makes the following contributions.

- We proposed a permission based distributed algorithm to coordinate the reader activation in large-scale RFID systems to achieve collision-free identification.
- The proposed algorithm is an adaptive algorithm which makes the RFID readers without identification requirements not participate in the coordination eventually.
- We conducted extensive simulations and the results show that our approach outperforms existing approaches in terms of execution time and message overhead.

The rest of the paper is organized as follows: Section II reviews the related works. Section III describes the system models used in this paper and Section IV formulates the problem. Our solution is illustrated in Section V. The simulation results for the proposed algorithm are reported in Section VI. Section VII concludes the paper.

II. Related Works

In existing works, different RFID reader coordination approaches are used to eliminate various kinds of RFID collisions. In the EPC C1G2 standard [8], a dense reading mode is proposed to eliminate reader-tag collision. The reader and tags are suggested to work in different channels in that standard. Colorwave [9] coordinates the RFID readers to eliminate reader-reader collision. It colors readers randomly and then allocate the conflicting readers with different colors. The work [13] further restricts the number of colors to the number of available radio channels, which is more practical. Later works further consider both reader-tag collision and reader-reader collision. HiQ [10] schedules the optimal channels and time slots for readers after a period of learning process. Without learning process, the work [4] schedules the channels and time slots by a region splitting and rotation method. Season [14] optimizes the identification in the overlapped interrogation regions of multiple readers in order to minimize the execution time. It adopts a two-phase approach: in the first phase all the tags are read without handling collisions to accelerate the identification process, and then in the second phase the readers with potential collision are coordinated such that some of them send the requests to the tags while the others only listen to the tags. RASPberry [15] investigated the stability problem in large-scale RFID systems. All these works above are centralized approach which suffer from large computation overhead, long latency, one-point failure risk, etc.

Recent research began to investigate distributed solutions. The work [11] utilizes the beacon messages at a common control channel to avoid collisions among readers. The work [16] reduces collisions through the coordination of RFID readers, assuming that the readers can communicate with each other. In these works, all the readers belong to one common application and are involved in single data gathering task. The approaches cannot directly be applied to the scenario with multiple applications. The PRDC [12] is the latest work using distributed approach to coordinate RFID readers for multiple applications. However, that approach is not adaptive. It means that all of the readers need to participate into the coordination anyway even if they finish their identification already. This cause unnecessary message overhead and increased execution time.

In other RFID research, the works [17], [18] developed languages and techniques to specify various complex identification requests from the users. The work [19] designed an efficient distributed RFID storage model for supply chain management. It leverage Bloom filters to save storage space and improve process efficiency of existence queries and path queries. These works are useful for the RFID system with multiple applications hence complementary to this paper.

III. System Model

In this section, we introduce the system models used in this paper.

A. RFID Identification Model

RFID is an advanced digital identification technology based on radio frequency. A typical RFID system consists of RFID readers and RFID tags. During the identification process, a reader sends out an identification request to the tags, and tags reply with pre-stored IDs and associated information.

Like other radio-based devices, RFID reader has its *interrogation region* and *interference region* [1]. Interrogation region is a region within which radio signal is sufficient strong such that the tags in that region can communicate with that reader successfully. Interference region is a region within which radio signal sent from that reader can affect the response of tags in that region. Interference region usually contains the interrogation region.

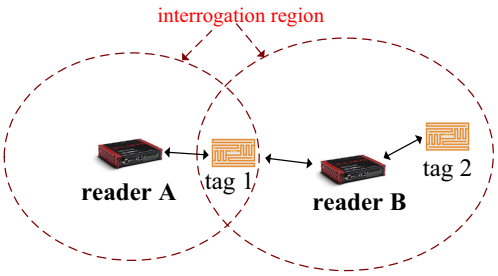

Fig. 1. Reader-reader collision

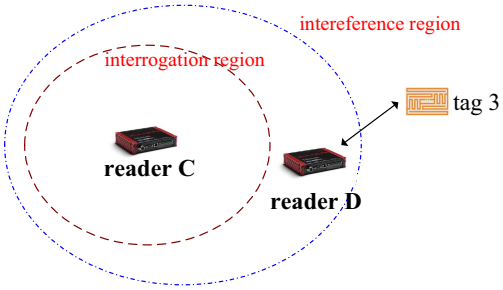

Fig. 2. Reader-tag collision

B. RFID Collision Model

A large-scale RFID system usually includes many RFID readers and tags. The multiple concurrent radio communications among RFID readers and tags cause serious inference (called collision in the RFID field). The collisions generally can be categorized into three types:*tag-tag collision*, *reader-reader collision*, and *reader-tag collision* [4]. Tag-tag collision denotes that two tags collide when they response to a reader's request simultaneously. This kind of collision is well solved by anti-collision protocols in the MAC layer. In this paper, we further investigate the other two kinds of collisions. Reader-reader collision (or R-R collision) denotes that, if two or more readers send the requests simultaneously, the tags in their overlapped interrogation region cannot distinguish the requests and cause communication failure. As shown in Fig. 1, tag 1 cannot be read by reader A and reader B. Reader-tag collision (or R-T collision) denotes that, as shown in Fig. 2, if reader D resides in the interference region of reader C and both the readers send requests, all the data transmissions from reader D (e.g. the data transmission with tag 3) are interfered by the signal sent from reader C and cannot be read. Proper reader activation methods are needed to alleviate the collision. It is noted that reader-tag collision can be solved by assigning different channels to the tag and the reader, but reader-reader collision cannot be solved even by assigning different channels. In this paper, we assume single channel for each reader for the sake of simplicity and further optimization considering multi-channels can be seen in [4].

C. Application Model

There are usually multiple applications running in a large-scale RFID system. Each application can issue one or more identification requests, and each identification request involves tags in different physical places. Such an identification request can be fulfilled by one or more combinations of RFID readers. Our objective is to choose proper combinations of RFID readers and activate them in proper time to complete all the identification requests.

We do not assume that there is a central server for the coordination purpose. This is practical for several reasons: First, the applications may not be willing to exchange information with others due to security concerns or business considerations. Second, the single point failure and performance bottleneck of centralized computing is not suitable for large-scale systems. Third, such a server is usually unavailable or with large cost. A distributed approach is needed to activate the RFID readers.

IV. Problem Formulation

In this section, we formulate the problem after defining *independent region* and *execution time*.

To facilitate the activation of RFID readers, we divide the region covered by the RFID system into several small sub-regions called *independent regions* as follows.

Definition 1(Independent Region): An independent region is the maximum geographic region which is covered by one and only one RFID reader's interrogation region.

We also define the execution time of RFID identification considering multiple requests from different applications:

Definition 2(Execution Time): The execution time of an RFID identification task is defined by the time that the last tag is read for the first time.

We then formulate the problem as follows:

Given a region A which is covered by a set of RFID readers. Some RFID tags to be read are placed in that region. There are multiple applications running on top of the readers, each of which issues identification requests for independent regions in A. We assume that 1) Each RFID reader can determine its own interrogation region and interference region. 2) Each reader know its own location and can communicate with neighboring readers. 3) Tag-tag collision is already solved by the MAC layer, while read-tag collision and reader-reader collisions affect the identification process. Our objective is to achieve the minimum execution time of the RFID identification for all requests.

V. The Solution

In this section, we propose our solution to activate RFID readers adaptively in a large-scale RFID system. We first briefly discuss the basic idea of the solution, and then illustrate the details.

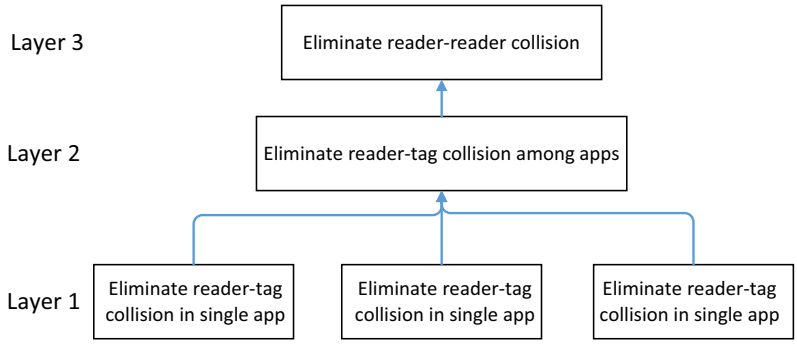

Fig. 3. Structure of our approach

A. Basic Idea

The challenge of the problem is how to coordinate the concurrent requests from readers which may cause reader-reader collision and reader-tag collision. Reader-tag collision is regarded to be more serious than reader-reader collision, since reader-tag collision causes the reader not work at all while reader-reader collision affects only the identification in the overlapped interrogation region. Both the kinds of collisions can be solved by resource mutual exclusion methods but with some differences. For reader-tag collision, the readers themselves are the resources needed to be competed for, while for reader-reader collision, the overlapped independent regions are the resources needed to be competed for.

We designed a three-layer approach to coordinate the activation of RFID readers. The structure of it can be seen in Fig. 3. In the first layer, the coordination is conducted for each application to eliminate reader-tag collision. The conflicted readers are assigned into different sets which will be activated in different time slots. In the second layer, the coordination is conducted among applications to eliminate reader-tag collision. And in the third layer, the coordination is conducted among readers to eliminate reader-reader collision.

We use the permission-based mutual exclusion approach to coordinate the activation: an reader can access the resource (reader itself for reader-tag collision or independent region for reader-reader collision) before it gets all the permissions from other readers that may be interfered with each other. Such approach makes all conflicted readers work in different time duration without overlap. In the following of this section, we illustrate how to follow this idea to activate different readers in a proper way and achieve desirable performance.

B. Data Structure: Conflict Graph

To facilitate the design of our approach, we introduce *conflict graph* [20] as a supporting data structure. Conflict graph describes the conflict relationship among readers to access resources. Conflict graph is an undirected graph whose vertices are RFID readers. If it exists an edge between reader R_1 and reader R_2 in the graph, the identification of R_1 will be affected by R_2. The conflict graph is called *R-R conflict graph* or *R-T conflict graph*, according to the kind of collision (reader-reader collision or reader-tag collision, respectively) with which the graph deals.

TABLE I
MATHEMATIC NOTATIONS USED IN THIS PAPER

Symbol	Meaning
readers	the set of readers in the system
clock	logic clock of the current reader
state	the variable of the current reader denoting the state of accessing resources, which can be "trying", "in" and "out"
C	the readers that may potentially collided with current reader
delayed	the readers whose resource access is delayed by the current reader
REQUEST($clock$,i)	the message issued from reader i to acquire the permission
PERMISSION(i,j)	the permission granted from reader i to reader j
next	the set of readers that will be considered in the next round
R_i (i=1...n)	A list of readers each of which has the interrogation region containing region i (i=1...n)
R	An array including R_i (i=1...n)
$result_i$ (i=1...)	A list of readers that will be activated in time slot i (i=1...)

C. Reader Activation Algorithm

We then propose the Adaptive Distributed Reader Activation Approach (ADRA) to activate the readers to finish the identification tasks. This approach includes two algorithms. The first algorithm corresponds to the first layer of our approach, and serves for eliminating reader-tag collision for individual applications. The other algorithm corresponds to the second and third layer of our approach, and serves for eliminating reader-tag collision among different applications and reader-reader collision. We summarize the mathematical notations used in the algorithms in TABLE I. The details of the algorithms are shown in Algorithm 1 and Algorithm 2.

Algorithm 1 computes the reader set that can be activated without reader-tag collision for individual applications. The

Algorithm 1: Local Collision Elimination Algorithm

Input : *regions*
Output: *result*

1 **foreach** region $i \in$ *regions* **do**
2 | obtain the readers R_i whose interrogation regions contain i
3 | add R_i into *readers*
4 **endfch**
5 **foreach** reader $r \in$ *readers* **do**
6 | calculate its frequency occurred in R
7 **endfch**
8 **while** *readers* $\neq \emptyset$ **do**
9 | obtain $r_{max} \in$ *readers* with the maximum occurrence frequency
10 | add r_{max} into *result*[*i*]
11 | *i*++
12 | **foreach** reader $r \in$ *readers* **do**
13 | | **if** *r* has reader-tag collision with r_{max} **then**
14 | | | *readers* $=$ *readers* $- r$
15 | | | **if** *r* is the unique reader to cover any uncovered region **then**
16 | | | | *next* $=$ *next* $\cup \{r\}$
17 | | | **end**
18 | | **end**
19 | **endfch**
20 | update the occurrence frequency in *readers*
21 **end**
22 **if** not all the regions are covered **then**
23 | put *next* in *readers* and repeat line 5 to 21
24 **end**

input of the algorithm is the independent regions requested by the application, *regions*. And the output is an array *result* whose elements *result*[*i*](*i*=1....) consists of the readers that can be activated in time slot i without reader-tag collision.

We assume that each independent region is equally important. All the readers that cover these regions are put into a list *readers* for further procession (line 1-4). Then the occurrence frequency of each reader in this list is calculated (line 5-7). The occurrence frequency of a reader denotes the number of independent regions it covers. We select the one with the maximum occurrence frequency into the result set *result*[*i*] (line 9-10), and then mark the regions it covers as the covered regions. Some readers are removed from consideration if they have reader-tag collision with the selected reader (line 12-14). If they are the unique readers that can identify some uncovered independent regions, they are put into *next* (line 15-17). After that, we update the occurrence frequency of the left readers in terms of the number of uncovered regions (line 20). The process is repeated until all readers are processed. We then check whether any independent region has not be covered. If in such case, we put all the readers of *next* into *readers*, and repeat the above process (line 22-24). Eventually, the target independent region can be fully covered. The readers in different result set *result*[*1*], *result*[*2*]...., are activated in different time slots, and the readers in *result*[*i*]($i = 1, 2...$) is reader-tag collision-free and can work currently.

We further eliminate the collision among readers in different applications. The readers in *result*[*i*] need to get a two-fold right to perform operations: get the first right considering reader-tag collision among applications, and then get second right considering reader-reader collision. Algorithm 2 is used

Algorithm 2: Adaptive Mutual Exclusion Algorithm

Input : *result*[*i*]

Function: mutualExclusiveAccess()
1 **foreach** reader $r \in$ *result*[*i*] **do**
2 | **if** *r* has R-T collision(or R-R collision for 2rd process) with others **then**
3 | | acquireResource()
4 | **end**
5 **endfch**

Function: acquireResource()
1 *state* $=$ *trying*
2 **if** *i* is in the 1st process *or* *i* is not involved in the 1st process before **then**
3 | *lrd* = *clock*+1
4 **end**
5 **foreach** $j \in C$ **do**
6 | send REQUEST($clock, i$) to p_j
7 **endfch**
8 **if** *i* is in the 1st process **then**
9 | **foreach** j has R-T collision with *i* **do**
10 | | *compete*$_j$=*true*
11 | **endfch**
12 **end**
13 wait for $(C = \varnothing)$
14 *state* $=$ *in*
15 read the tags from the region

Function: releaseResource()
1 *state* $=$ *out*
2 **foreach** $j \in$ *delayed* **do**
3 | send PERMISSION($\{i,j\}$) to p_j
4 **endfch**
5 $C =$ *delayed*
6 *delayed* $= \varnothing$

Function: processRequest(REQUEST(k, j))
1 *clock* = max(*clock*, *k*)
2 *prio*= (*state*=*trying*) and $(lrd, i) < (k, j)$ and (*compete*$_j$=*true*)
3 **if** (*state*=*in*) *or* (*prio* $=$ *true*) **then**
4 | *delayed* = *delayed* $\cup\{j\}$
5 **else**
6 | send PERMISSION($\{i,j\}$) to p_j
7 | $C = C \cup \{j\}$
8 | **if** *state*=*trying* **then**
9 | | send REQUEST(*clock*, i) to p_j
10 | **end**
11 **end**

Function: processPermission(PERMISSION($\{i,j\}$))
1 $C = C - \{j\}$;

to grant the rights based on mutual exclusive resource access. The algorithm is a permission based approach, which we modified from [21]. We begin from Function *mutualExclusiveAccess* which will be invoked twice, the first time for the right granting based on the R-T conflict graph and the second time for right granting based on R-R conflict graph.

In the algorithm, each reader is in the state of *trying*, *in*, or *out*. *Trying* means that the reader is interested in this resource and tries to get the permissions. *In* means that all the permissions needed are obtained and now the reader can begin the identification process. *Out* means that the reader is not interested in the resource anymore (e.g. the identification is finished). Each reader needs to get the permission from other potential conflicted readers. We use a token residing between a pair of readers to denote such permission: if a reader gets such a token, it obtains the permission from its

partner. Initially, we assume that the permission is randomly placed in a pair of readers. Let *C* include the readers from which the permission need to be get. It is noted that if a reader already has the permission of its partner (due to random placement of the permission), the partner is not included in *C*.

In Function *acquireResource*, the reader first sets the state to *trying*, updates the local logic clock, and then sends the permission request REQUEST to all readers in C. After that, it waits until all the permissions are received. Since multiple readers may compete for common resources, the permission is granted based on their priorities. When a reader receives a permission request, it computes the priority in Function *processRequest*. If the reader is in the identification process (*state=in*), it has a higher priority. If the reader is competing for the resource (*state=trying*), the priority is granted based on the requesting time and the ID of the reader, denoting by the pair (*lrd*, *i*). (*lrd*, *i*) has a total order among all the readers so no deadlock happens. The requests with lower priorities are delayed and stored in *delayed*. Otherwise, the reader sends the permission to the requester, and sends itself request to it. After a reader finishes the identification, the permission is released in function *releaseResource*. It updates the state to *out* and then sends out the permissions to all the delayed requests. When the permission is received, as shown in Function *processPermission*, *C* shrinks accordingly. Each reader eventually can obtain all the permissions and get into the state *in* at function *acquireResource*. After the reader gets the two-fold right, the identification process can be started. Some special handling is put to the local clock update and the priority determination. As shown in Function *acquireResource* line 2-4, the clock will be updated once when both reader-tag collision and reader-reader collision between a pair of readers need to be resolved. It is to eliminate the possible deadlock in this case. As shown in Function *processRequest* line 2, when only reader-reader collision need to be resolved, the priority will not consider the R-T permission (using $compete_j$ to denote).

We adopt the shared permission mechanism in the algorithm for achieving adaptiveness. Classical permission based algorithm needs to gather all the permissions from the readers that may collide with, which is not adaptive for algorithm execution [20]. Using shared permission mechanism, a permission is shared by a pair of readers that may collide with each other, and the permission has no need to return once granted. Therefore if one reader has no identification request any more, its permission will be eventually granted to its partner and no further permission request is needed [21]. The adaptiveness promotes equal benefit and duty of different readers, and also improves identification performance in terms of message overhead and execution time.

D. Discussion of the Approach

We first prove the correctness and adaptiveness of the proposed approach.

Theorem 1: ADRA can guarantee that all the readers are activated without any reader-tag collision and reader-reader collision.

Proof: We prove this by showing that reader-tag collision and reader-reader collision cannot exist when following ADRA to activate readers. ADRA first divides the readers into different sets using Algorithm 1, and then further allocates the readers in one set into different time slots using Algorithm 2. In Algorithm 1, the readers in one set cannot have reader-tag collision, because whenever a reader is put into the set, *result*[*i*]($i = 1$....), all of the readers with reader-tag collision with it are excluded from the set (Algorithm 1, line 14). This ensures that reader-tag collision among readers in one application cannot exist. Possible reader-tag collision among readers in different applications is also avoided by the coordination of Algorithm 2. A reader needs to get a permission from another reader requested by other applications if there is a possible reader-tag collision between them (Function *mutualExclusiveAccess*, line 2). Since all of the reader-tag collisions among readers in individual application is eliminated by Algorithm 1 already, here only the reader-tag collisions among readers in different applications are handled. For a pair of conflicted readers, the one with smaller (lrd, i) will obtain the permission while the other one will be delayed (Function *processRequest*, line 2-6). After that, the reader performs a similar process to obtain the permission to read the independent region considering reader-reader collision. The uniqueness of permission between a pair of readers guarantees that the readers with possible reader-reader collision cannot be activated together. To conclude, all of the readers are activated without any reader-tag collision and reader-reader collision if ARDA is used for coordination. ■

Theorem 2: ADRA can guarantee that all of the identification requirements from different applications can be complete eventually.

Proof: The readers are activated concurrently for different applications if no reader-tag collision and reader-reader collision exist. When reader-tag collision and reader-reader collision exist, some readers in one application may be blocked by some readers in other applications. We show that the blocked readers will be eventually activated and then their identifications can be finished. It is noticed that the reader follow a two-step process to get permissions before performing identification, one based on R-T collision conflict graph and the other based on R-R collision conflict graph. There are two cases. In the first case, the pair of readers may have both reader-reader collision and reader-tag collision, which is hold if the sum of their radiuses of interrogation regions is smaller then the radius of one reader's interference region. Then the pair of readers have links in both R-T collision conflict graph and the other based on R-R collision conflict graph. Without losing generality, we assume that reader A gets the permission from reader B based on R-T collision conflict graph. It can be inferred

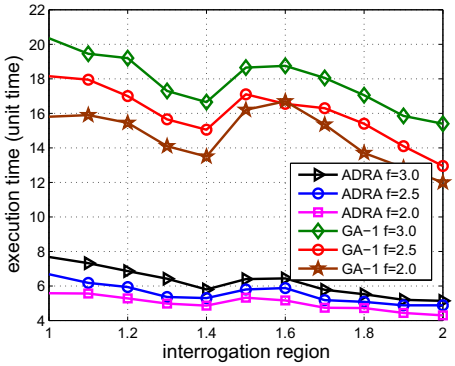

Fig. 4. The execution time of GA-1 and ADRA at different interrogation regions and a fixed interference region

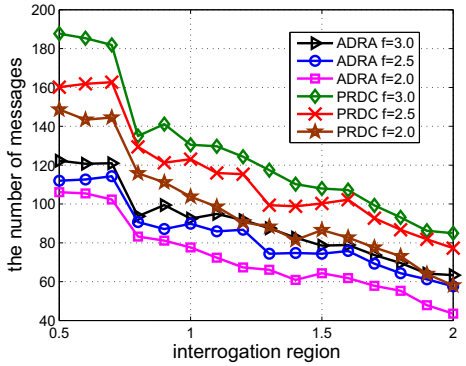

Fig. 5. The message overhead of ADRA and PRDC at different interrogation regions and a fixed interference region

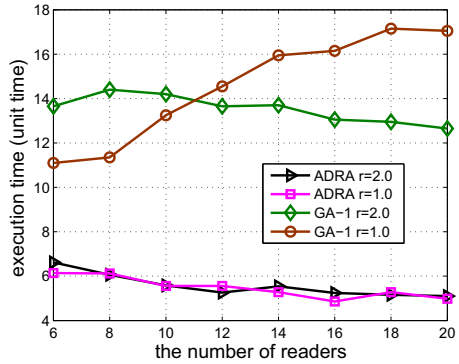

Fig. 6. The performance of GA-1 and ADRA when varying the number of readers

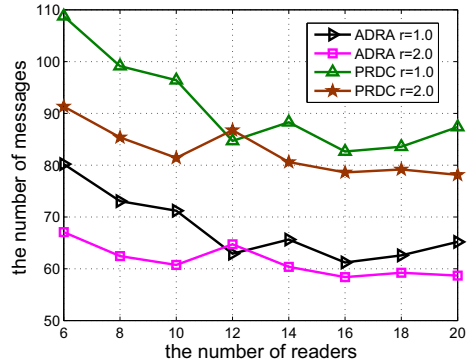

Fig. 7. The message overhead of ADRA and PRDC when varying the number of readers

that reader A has a smaller (lrd, i) (Function *processRequest*, line 2-6). When reader A request the permission for reading independent region, the sending (lrd, i) has not been changed and still smaller then B (Function *acquireResource* line 2-4). Reader A performs identification and then releases both permissions to reader B. In the second case, the pair of readers have only reader-tag collision or reader-reader collision. The reader-tag collision can be easily handled since only the 1st process of Algorithm 2 is executed. If they have reader-reader collision only, we assume that the permission is placed at reader A without losing generality. Since the R-T permission in this case is not a concern, we can allow the reader A send the permission to reader B if reader B's (lrd, i) is smaller (Function *processRequest*, line 2-6). It is noted that here the reader A has the $compete_B$ to be *false* since no reader-tag competition between them. After the permission is granted, reader B will finish the identification and then send the permission to the reader A to finish its identification eventually. ■

Theorem 3: ADRA can guarantee that the readers without identification requirement in any application will be excluded from coordination eventually.

Proof: This is to validate the adaptiveness of the proposed approach. This is achieved by Algorithm 2. When a reader does not need to perform identification anymore, its permission shared with others will be granted to other readers since its state is set to "out" and the priority is lower then others (Function *processRequest*, line 2-3). When one such permission is sent to its partner, no "REQUEST" will be sent (Function *processRequest*, line 8-10) and no "PERMISSION" will be received anymore (Function *acquireResource*,line 7). Therefore after all the permissions are granted to others, the reader can be stopped and does not need to participate in the coordination for others. When it has the identification requirement again, it can actively send the request to others and begin a new coordination process. ■

We then discuss the computation complexity of the proposed approach. Algorithm 1 is a centralized algorithm executed in one single application, which is evaluated using the number of key computations. Algorithm 2 is a distributed algorithm executed in each RFID reader, which is evaluated using the number of messages exchanged for key operations.

For the Algorithm 1, we need $O(n \times m)$ computations where *n* is the number of readers and *m* is the number of independent regions. This algorithm can be finished off-line. In this paper, we focus on the performance of online execution (i.e. distributed execution) hence not further optimize Algorithm 1, which is left for future work. For the Algorithm 2, the number of messages involved in one identification is $2|C|$ or $4|C|$ where $0 \leq |C| \leq n-1$, depending on that both reader-tag collision and reader-reader collision exist or only one of them exists. The exact number of messages depends on the permission distribution among readers.

VI. Performance Evaluation

Simulations are carried out to validate the effectiveness of our approach. We compare the performance of our approach ADRA with that of GA-1 algorithm proposed in [4] in terms of execution time. That approach achieves good performance when scheduling the readers of a large-scale RFID system in a centralized way. We also compare the message overhead of ADRA with that of PRDC [12] that employs similar distributed processing idea but not adaptive.

A. Simulation Setup

In the simulation, a number of RFID readers are randomly placed in a room of $10m \times 10m$. Each reader knows its own interrogation range r, interference region f, and the location. The readers also can communicate with neighboring readers. The room is divided into multiple independent regions according to the possible reader-tag collision and reader-reader collision. In the system, there are multiple requests issued from different applications. Each request includes the same number of independent regions. We vary different parameters to check the performance of the approaches. 100 runs simulations are repeated to get each data point of the figures with the confidence level of 0.95.

B. Impact of Interrogation Region

We compare the performance of different approaches varying the interrogation region r and making the interference region fixed to be specific value f. The results are shown in Fig. 4-5. We set the number of readers as 10, the number of requests as 5, and the independent regions included in a request as 4.

As shown in Fig. 4, the execution time of both GA-1 and ADRA trends to decrease when the interrogation region increases. The decreasing is because that at a fixed interference region, the reader with a larger interrogation region can read more tags and then save time. The performance of GA-1 is fluctuated because it is an approximation algorithm and the approximation ratio varies in different configurations. The execution time of ADRA is always much less than that of GA-1. When $f = 2$, the execution time of ADRA is reduced by 64.0%-69.1%, compared with GA-1.

We then compare the performance of ADRA and PRDC. As shown in Fig. 5, the message overhead of both ADRA and PRDC trends to decrease when the interrogation region increases, which is consistent with the results shown in Fig. 4. For both approaches, the message overhead increases when *f* increases from 2.0 to 3.0. The message overhead of ADRA is much less than that of PRDC with the same *f*. This is because that ADRA is adaptive and hence less permission requests are sent out for coordination. When $f = 2$, the number of messages sent by ADRA is reduced by 24.3%-29.2% compared with PRDC.

C. Impact of Number of Readers

The number of readers is another important factor affecting the execution time of identification. We vary the number of readers from 6 to 20 and compare the performance of different approaches (under different interrogation region *r*). The results are shown in Fig. 6-7, where the number of requests is 5, the independent regions included in a request is 4, and the interference region is 1.3 times the interrogation region *r*.

As shown in Fig. 6, the execution time of ADRA is reduced when the number of readers increases. This is because in a fixed region, more readers denote more independent sub-regions. Therefore when each request issues a fixed number of independent regions, the collision can be reduced and hence the execution time is reduced. It holds in different interrogation regions as shown in the figure. In all cases, the execution time of ADRA is less than that of GA-1. When $r = 2$, the execution time is reduced by 48.1%-58.8% by ADRA compared with GA-1.

As shown in Fig. 7, the message overhead of both ADRA and PDRC decrease when the number of readers increases. In all cases, the message overhead of ADRA is much less than corresponding PDRC due to adaptiveness. When $r = 2$, the message overhead of ADRA is reduced by 24.9%-26.9% compared with PRDC.

D. Impact of Number of Requests and Number of Independent Regions in a Request

In this subsection, we compare the performance of ADRA and PRDC under different number of requests and different number of independent regions included in one request. Since it is not a concern for the centralized algorithm GA-1, we do not include it in the comparison.

According to Fig. 8, the execution time of ADRA and PRDC is proportional to the number of requests. The relation between the execution time and the number of requests is near linear. It shows that the distributed coordination does not incur exponential increased overhead and hence suitable for large-scale systems. The performance of ADRA is better than PRDC since its adaptive design reduces the number of messages to be sent. When $r = 2$, the message overhead of ADRA is reduced by 24.6%-25.2% compared with PRDC.

According to Fig. 9, it can be seen that the message overhead of both ADRA and PRDC increase when the number of independent regions in a request increses. Again,

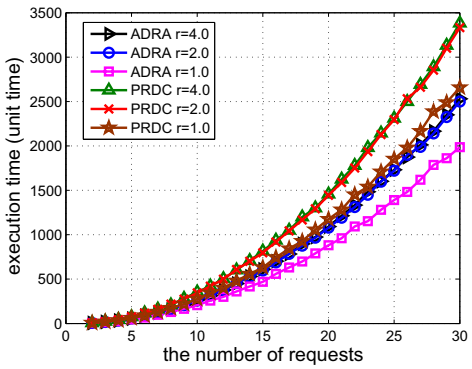

Fig. 8. The message overhead of ADRA and PRDC when varying the number of requests

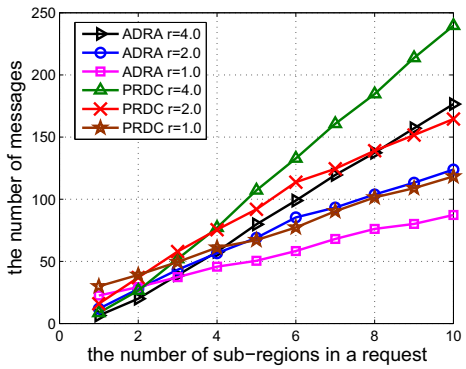

Fig. 9. The message overhead of ADRA and PRDC when varying the number of independent regions included in a request

the performance of ADRA is better than that of PRDC. When $r = 2$, the message overhead of ADRA is reduced by 24.7%-25.3% compared with PRDC.

VII. Conclusion

In this paper, we investigated the adaptive distributed reader activation problem in large-scale RFID systems. After analyzing the identification requirements in such systems, we build the conflict graphs of two kinds of RFID collisions: reader-tag collision and reader-reader collisions. A three-layer approach is proposed to achieve the collision-free identification. The share permission based mutual exclusion method is used as a basic building block in the approach, to coordinate the conflicting readers. The adaptiveness of the approach is achieved by assigning the permissions properly to the readers and make the idle readers no need to participate in the coordination control. We have taken extensive simulations to validate the effectiveness of the proposed approach. The results show that the proposed approach outperforms existing approaches in terms of execution time and message overhead.

Acknowledgment

This research is supported in part by the Fundamental Research Funds for the Central Universities of China No. 216-410500026, Nature Science Foundation of Hubei, China No. 2015CFB340, and Outstanding Young Academic Talents Start-up Funds of Wuhan University No. 216-410100004.

References

[1] K. Finkenzeller, *RFID handbook: Fundaments and Application in Contactless Smart card and Identification*. John Wiley & Sons, 2003.
[2] S. Lahiri, *RFID Sourcebook*. IBM Press, 2006.
[3] R. Want, "An introduction to RFID technology," *IEEE Pervasive Computing*, vol. 5, no. 1, pp. 25–33, 2006.
[4] Z. Zhou, H. Gupta, S. Das, and X. Zhu, "Slotted scheduled tag access in multi-reader RFID systems," in *Proc. of ICNP*, 2007, pp. 61–70.
[5] D.-H. Shih, P.-L. Sun, D. C. Yen, and S.-M. Huang, "Taxonomy and survey of RFID anti-collision protocols," *Computer Communications*, vol. 29, no. 11, pp. 2150 – 2166, 2006.
[6] V. Sarangan, M. R. Devarapalli, and S. Radhakrishnan, "A framework for fast RFID tag reading in static and mobile environments," *Computer Networks*, vol. 22, no. 5, pp. 1058–1073, 2008.
[7] W. Zhu, J. Cao, H. Chan, X. Liu, and V. Raychoudhury, "Mobile RFID with a high identification rate," *IEEE Trans. on Computers*, vol. 63, no. 7, pp. 1778–1792, July 2014.
[8] EPC Global, Inc., "EPC^{TM} Radio Frequency Identity Protocols Class 1 Generation 2 UHF RFID Protocol for Communications at 860MHz-960MHz Version 1.1.0," *Auto-ID Labs White Paper WP-HARDWARE-045*, Oct 2007.
[9] J. Waldrop, D. Engels, and S. E. Sarma, "Colorwave: a MAC for RFID reader networks," in *Proc. of WCNC*, 2003, pp. 1701–1704.
[10] J. Ho, D. Engels, and S. E. Sarma, "HiQ: a hierarchical Q-learning algorithm to solve the reader collision problem," in *International Symposium on Applications and the Internet Workshops*, 2006, pp. 23–27.
[11] J. Yu, W. Lee, and D.-Z. Du, "Reducing reader collision for mobile RFID," *IEEE Trans. on Consumer Electronics*, vol. 57, no. 2, pp. 574–582, 2011.
[12] W. Zhu, X. Cui, and C. Hu, "Complex data collection in large-scale RFID systems," in *Proc. of International Conference on Smart Computing*, 2014, pp. 25–32.
[13] D. Vinay, M. Malena, R. John, D. Devaraj, and P. Salil, "Perturbative time and frequency allocations for rfid reader networks," *HP Labs Technical Report HPL-2005-162*, Sep. 2005.
[14] L. Yang, J. Han, Y. Qi, C. Wang, T. Gu, and Y. Liu, "Season: Shelving interference and joint identification in large-scale RFID systems," in *Proc. of INFOCOM*, 2011, pp. 3092–3100.
[15] S. Tang, J. Yuan, M. Li, G. Chen, Y. Liu, and J. Zhao, "RASPberry: A stable reader activation scheduling protocol in multi-reader RFID systems," in *Proc. of ICNP*, 2009, pp. 304–313.
[16] J. Ahmed and M. H. T., "Decentralized RFID coverage algorithms with applications for the reader collision avoidance problem," *IEEE Transactions on Emerging Topics in Computing*, IEEE early access article 2015.
[17] Y. Bai, F. Wang, P. Liu, C. Zaniolo, and S. Liu, "RFID data processing with a data stream query language," in *Proc. of ICDE*, 2007, pp. 1184–1193.
[18] W. Yao, C.-H. Chu, and Z. Li, "Leveraging complex event processing for smart hospitals using RFID," *Journal of Network and Computer Applications*, vol. 34, no. 3, pp. 799–810, July 2011.
[19] J. Liu, B. Xiao, K. Bu, and L. Chen, "Efficient distributed query processing in large RFID-enabled supply chains," in *Proc. of INFOCOM*, 2014, pp. 163–171.
[20] M. Raynal, *Distributed Algorithms for Message-Passing Systems*. Springer, 2013.
[21] O. Carvalho and G. Roucairol, "On mutual exclusion in computer networks," *Comm. ACM*, vol. 26, no. 2, pp. 146–147, 1983.