Mobile Netw Appl (2014) 19:524–533
DOI 10.1007/s11036-014-0524-9

Check out the Rules: Towards Time-Efficient Rule Checking over RFID Tags

Yafeng Yin · Lei Xie · Sanglu Lu · Daoxu Chen

Published online: 20 July 2014

Abstract With the rapid proliferation of RFID technologies, RFID has been introduced to the applications like safety inspection and warehouse management. Conventionally a number of deployment rules are specified for these applications. This paper studies a practically important problem of rule checking over RFID tags, i.e., checking whether the specified rules are satisfied according to the RFID tags within the monitoring area. This rule checking function may need to be executed frequently over a large number of tags and therefore should be made efficient in terms of execution time. Aiming to achieve time efficiency, we respectively propose two protocols, CRCP and ECRCP. CRCP works based on collision detection, while ECRCP combines the collision detection and the logical features of the rules. Simulation results indicate that our protocols achieve much better performance than other solutions in terms of time efficiency.

Keywords RFID · Rule checking · Algorithm design · Time-efficient · Optimization

1 Introduction

With the development of RFID technologies, RFID tags have been widely deployed into a variety of applications. Conventionally, an RFID system typically consists of one or several readers and a large number of tags. Each tag is attached to a physical item and has a unique identification (ID) describing the item. The reader recognizes the object by identifying its attached tag.

Recently, RFID has been introduced to a number of rule checking-based applications, e.g., safety inspection and warehouse management. In these applications, a set of rules are specified over the deployment of the items (tags), which vary from application to application. For example, in the chemical laboratory, as shown in Fig. 1a, when some chemicals (eg. metal material and corrosive solution) come together, the chemical reaction occurs, which may cause an accident. Therefore, these objects should not be placed together. In the warehouse management, the lighter and the alcohol should not be close to each other in consideration of safety, while the pillow core and the pillowcase should be placed together, since they are matching products, as shown in Fig. 1b. In order to check the rules over a specified area, the reader can reasonably adjust its power to a certain level. The objective is to check whether the rules are satisfied according to the detected information from tags in the scanning area. The rule checking function may need to be executed frequently over a large number of tags and therefore should be made efficient in terms of execution time. For example, the security checking in the airport, as shown in Fig. 1c. A straightforward solution is to collect all the tag IDs, and then check the rules one by one based on the collected IDs. However, this approach is rather time-consuming due to the large number of tags deployed in the applications.

Based on the above understanding, it is essential to provide a time-efficient solution for these rule checking-based applications. We note that conventionally the rules are only related to the tags' categories instead of the detail IDs, and it is possible to quickly check the rules by exploring their logical features. For example, if the alcohol is not detected in the warehouse management, then the rule over the lighter and the alcohol can be verified as satisfied immediately, no

Y. Yin · L. Xie (✉) · S. Lu · D. Chen
State Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing, China
e-mail: lxie@nju.edu.cn

Y. Yin
e-mail: yyf@dislab.nju.edu.cn

S. Lu
e-mail: sanglu@nju.edu.cn

D. Chen
e-mail: cdx@nju.edu.cn

Fig. 1 Rule-checking based applications

matter whether the lighter exists. In regard to the tag identification protocol, traditional slotted ALOHA-based solutions always try to reduce the collision slots, without sufficiently exploring the information from the collision slots. In this paper, by effectively resolving the collision slots and leveraging the rules' logical features for verification, we propose efficient rule checking protocols based on the categories of tags, without the need of identifying tags one by one. While verifying all related rules in the applications, our protocols can dramatically reduce the overall execution time for rule checking.

We make the following contributions in this paper.

- We study a practically important problem of rule checking over a large set of RFID tags, which is essential for a number of RFID applications, such as safety inspection, warehouse management, and so on.
- We propose a time-efficient protocol for rule checking based on collision detection, which aims at sufficiently exploring the information from the collision slots. Furthermore, we propose an enhanced protocol which combines the collision detection and the logical features of the rules. By leveraging the rule's logical property, the enhanced protocol can effectively simplify the checking process.
- To the best of our knowledge, this is the first theoretical work to investigate the rule checking problem in RFID systems. While leveraging the information of the physical layer and the application layer, our solution conducts a cross-layer optimization to effectively achieve time efficiency.

2 Related works

Previous research on RFID has focused on anti-collision protocols, which can be categorized into tree-based protocols [1] and ALOHA-based ones [2]. In ALOHA-based protocols, most of the existing work considers the collision slots unuseful and wastes the collision slots. Recent research shows that, analog network coding can be used to extract useful information from collision slots to improve the RFID reading throughput [3]. In the collision slot, the the signals from multiple tags exhibit small offsets, which are sufficiently small for decoding the collision slot [4]. Besides, Manchester coding technology can be used in RFID communications to detect the bit collision [5, 6]. In [7], Manchester coding is used to decode the tag identifier from the collision bits with the known mask. In [8], Manchester coding is used to extract the information from collision slots to enhance the efficiency of identifying the tags.

Instead of identifying all the tags, missing tag identification only aims to find the missing tags [9, 10]. Opposite to the purpose of finding missing tags, Zheng et al. propose a two-phase approximation protocol for fast tag searching in large-scale RFID systems [11]. Rather than identifying the tags, the RFID cardinality estimation protocols count the number of distinct tags [12, 13]. However, all the literature does not research the problem of rule checking over RFID tags.

Being different from the related work, our research focuses on efficiently checking the rules based on the tags' categories. We resolve the collision slots to verify the tags' categories specified by the rules and leverage the rules' logical features for rule checking. In order to resolve the collision slots, we will use the bit collision detection technology based on Manchester coding.

3 Preliminary

3.1 Frame slotted ALOHA protocol

Frame slotted ALOHA protocol (FSA) is a popular anti-collision protocol for tag identification. In FSA, the reader first broadcasts the request message and specifies the following frame size f. After receiving f, each tag selects $h(ID)$ mod f as its slot number. Here, h is a hash function, ID is tag ID. If no tag responds in a slot (empty slot), the reader closes the slot immediately. If only one tag replies in a slot (singleton slot), the reader successfully receives the tag ID and sends an acknowledgement to the tag. The tag will keep silent in the rest of the session. If multiple tags respond in the same slot (collision slot), a collision occurs, and the involved tags will be acknowledged to restart in the next frame. The reader repeats the above process until no tags respond.

3.2 Synchronization

As a popular ID-collection protocol conforming to EPC-C1G2 standard, FSA discards the collision slot. However, if the tags' responses are synchronized in a slot, we can resolve the collision slot to get the tags' responses. Furthermore, if the *bit-level* synchronization is achieved, then each bit of the responses can be utilized to resolve the collision slot.

In RFID systems, each tag communicates with the reader in a single hop way. The transmissions can be synchronized by the reader's signal [14]. According to EPC-C1G2 standard [15], the tags responding in the same slot can be synchronized through *QueryRep* or *QueryAdjust*. Besides, the effective scanning range in the passive RFID system is limited (eg. 5-6 meters). The time delay difference between tags caused by the difference of their distances to the antenna is less than $\frac{6m}{3\times10^8 m/s} = 0.02\mu s$. When a tag receives the reader's command, it will respond in a very short time. When the rate from a tag to the reader is 53Kb/s, it takes $18.88\mu s$ for the tag to transmit one bit. The maximum time delay difference is less than $0.02\mu s$, which can be neglected compared to $18.88\mu s$. In regard to the physical-layer signals, the signals from multiple tags exhibit small offsets, which are sufficiently small for decoding [4]. Therefore, we consider that the tags responding in the same slot are synchronized. Moreover, the tags has the probability to achieve *bit-level* synchronization, which is essential to recover the useful bits of the tags' responses from the mixed signal. As the tag's manufacturing technology is improved, we believe that a feasible *bit-level* synchronization can be achieved with more precise clock synchronization.

3.3 Manchester coding

In RFID systems, each tag encodes the backscattered data before sending it. In this paper, we use Manchester coding to achieve bit collision detection. Manchester coding has been used in Type B of ISO 18000-6 [16].

In Manchester code, a falling edge transition represents 1, a rising edge transition represents 0. The transitions can be used to accurately detect the bit collision [5, 6, 8], as shown in Fig. 2. When tag1 and tag2 transmit IDs to the reader simultaneously in a slot, the falling edge transition and the rising edge transition cancel each other out, leading

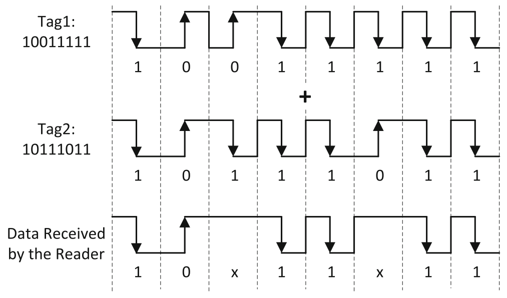

Fig. 2 Bit collision detection in Manchester code

to no transition in a received bit, such as bit3 and bit6. The corresponding bit is a collision bit. If there are more than two tags responding simultaneously in a slot, only all the tags transmit '0' (or '1'), the reader can recover the bit '0' (or '1'). Otherwise, the reader detects a collision bit x. We represent the conclusion as follows, $0 + ... + 0 = 0$ (all the tags send '0'), $1 + ... + 1 = 1$ (all the tags send '1'), $0 + ... + 1 = x$ (some tags send '0' and some tags send '1'). In order to decode the bits in a slot, current experimental platforms like USRP can be used [17], which can achieve the *bit-level* identification (0, 1, x).

4 Problem formulation

We assume that each tag has a category ID to denote the tag's category and the category ID is the prefix of tag ID. In order to efficiently get the category IDs, we will utilize the collision slots based on Manchester coding. Without loss of generality, we assume that a feasible *bit-level* synchronization can be achieved, which can be used to recover the category IDs from the mixed signal in a slot. In our problem, we check the rules based on the tag's category ID. We use $R = \{R_1, ..., R_i, ..., R_m\}$ to represent the rules. The category IDs in the rules are formulated as $C = \{C_1, ..., C_j, ..., C_n\}$. In real application, there often exist some constraint rules in the categories. For example, orange juice and apple juice both belong to fruit juice, it may be enough to have any one of them in the warehouse. The type of the relationship is called *OR*. While pillow core and pillowcase should be put together, their relationship's type is called *AND*. The lighter and the alcohol should not be close to each other for the sake of safety, their relationship's type is called *EXCLUSIVE*. Taking the actual situation into consideration, we classify the rules into three basic types.

- *OR*: At least one category must exist. We represent the rule as $R_i = C_j \vee C_k$. We list two categories here. Actually, there can be more categories in a rule. The rule's Boolean value $B(R_i)$ is false ($B(R_i) = 0$) only when both C_j and C_k are outside the scanning area, which are expressed as $B(C_j) = 0$ and $B(C_k) = 0$. Otherwise, $B(R_i)$ is true ($B(R_i) = 1$).

- *AND*: All the categories must exist together. We represent the rule as $R_i = C_j \wedge C_k$. The rule's Boolean value $B(R_i) = 1$ only when both C_j and C_k are in the scanning area, $B(C_j) = 1$ and $B(C_k) = 1$. Otherwise, $B(R_i) = 0$.
- *EXCLUSIVE*: The categories should not be put together. We represent it as $R_i = \neg(C_j \wedge C_k)$. It is essentially the same as the *AND* rule in consideration of logical relation. However, considering the actual situation in applications, we do not merge them. The rule's Boolean value $B(R_i) = 0$ only when $B(C_j) = 1$ and $B(C_k) = 1$. Otherwise, $B(R_i) = 1$.

However, if a rule is a hybrid rule, we can split it into subrules until each subrule belongs to one of the basic types. For example, a hybrid rule $R_i = (C_j \wedge C_k) \vee (C_u \vee C_v)$. We get the subrules, $C_j \wedge C_k$ and $C_u \vee C_v$. Then, $B(R_i)$ can be obtained from $B(C_j \wedge C_k)$ and $B(C_u \vee C_v)$.

Due to the large number of tags and rules, it is essential to propose a time-efficient solution for the rule checking-based applications, which aims at minimizing the execution time T for checking all the rules.

5 Baseline protocols

5.1 Frame slotted ALOHA based rule checking protocol (ARCP)

ARCP collects all the tag IDs in the scanning area , as described in Section 3.1. Then the reader extracts the category IDs in the scanning area and checks the rules.

5.2 Polling based rule checking protocol (PRCP)

We call the categories in the rules as *related categories*. PRCP checks the existence of *related category IDs* in the scanning area by broadcasting them one by one. When a tag's category ID is equal to the reader's request one, it will give a short response. The reader gets a nonempty slot. Otherwise, it gets an empty slot. When the polling process terminates, the reader checks the rules based on the tags' responses.

5.3 Bloom filter based rule checking protocol (BRCP)

BRCP uses the Bloom filter to inform the *related categories* to respond. For the related tags in the scanning area, each one uses k hash functions to select k slots to give a short response. The reader gets the responses as a virtual Bloom filter and obtains the wanted category IDs from the virtual Bloom filter to check the rules. In [11], a similar protocol CATS is proposed for fast tag searching in large-scale RFID systems.

6 Collision detection based rule checking protocol

6.1 Motivation

Based on the above analysis, we propose a Collision detection based Rule Checking Protocol (CRCP). It focuses on resolving the collision slots based on Manchester code, which can be used for bit collision detection according to Section 3.

In CRCP, if α categories $\{C_1, C_2, \ldots, C_\alpha\}$ should select the same slot to respond, the reader gets an α – collision slot. We use $M = \sum_{j=1}^{\alpha} M_j$ to represent the mixed signal of M_j. M_j and M are Manchester code of d bits. M_j is the short term of C_j to reduce transmission time. If $\alpha = 2$, $M_1 = 100101$, $M_2 = 010111$, then $M = \sum_{j=1}^{2} M_j = 100101 + 010111 = \text{xx01x1}$. M is the *expected code* to be received by the reader. The reader decodes M_j from M by comparing the received code with the expected one. If the *received code* M_r is 010111, the reader can confirm that only C_2 gives the response M_2. Because the first bit in M is x, while the first bit in M_r is 0. It indicates that C_1 is outside the scanning area, while C_2 is in the area. After that, the reader uses the decoded category IDs to check the rules.

6.2 Protocol overview

In CRCP, the reader sends the request by Bloom filter. It selects k hash functions $h_1, h_2, \ldots, h_k$ and c category IDs $C_1, C_2, \ldots, C_c$ to construct the Bloom filter with length l, as shown in Fig. 3. It maps each C_j into k bits at positions $h_1(C_j) \bmod l$, $h_2(C_j) \bmod l$, ..., $h_k(C_j) \bmod l$ and sets these bits to 1. When a tag receives the request, it checks the corresponding k bits. Only all the bits are 1, it gives response. See Fig. 3 for an example of $k = 3$, $l = 16$,

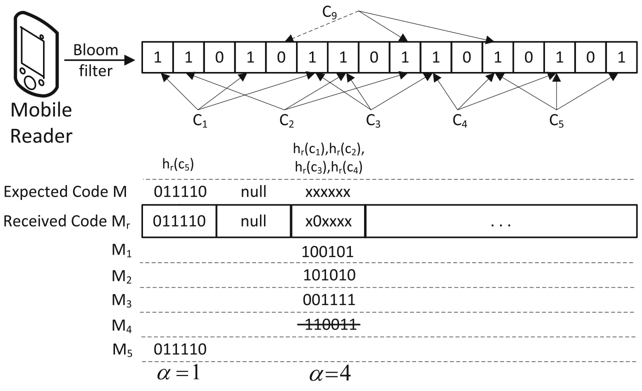

Fig. 3 Collision detection based rule checking protocol

the reader uses $h_i(C_j)$ mod 16 ($i \in [1,3]$, $j \in [1,5]$) to construct the Bloom filter. The tags in C_1, C_2, C_3, C_4, C_5 pass the test and give responses, while the tags in C_9 keep silent. If a tag in C_j should give response in the following frame with size f, it uses a hash function h_r to select a slot $h_r(C_j)$ mod f. Then, it sends d bits Manchester code M_j. Here, $M_j = h(C_j)$ mod (2^d), h is a hash function.

Because the reader knows the related categories and the hash functions, it can get the mapping relation of C_j and M_j. When getting the responses, the reader checks whether each bit in the slot can be recovered from the mixed signal. If the bits cannot be recovered, it means the number of tags responding in the slot is too large. Then the reader puts the corresponding category IDs into the new set C', which will be verified later. If the bits can be recovered successfully, then the reader resolves the collision slots, as shown in Algorithm 1. In the frame received from tags, suppose that the reader is able to get q_i new category IDs $C_i, C_j, \dots, C_k$ by decoding an α – collision slot. The reader decodes the collision slots one by one until it gets all the c *related category IDs* in the frame or the frame terminates. Then, it repeats the similar process. At last, the reader verify the category IDs in C' using probability p_r. The reader broadcasts an integer $y = \lceil p_r \times Y \rceil$, Y is a large constant. Then the tag calculates the hash result $h_r(ID)$ mod Y, only if $h_r(ID)$ mod $Y \leq y$, the tag gives response. In this way, there will be only a few number of tags (eg. 1) in one category giving responses. Then, the reader can decode the category IDs. When all the *related category IDs* are verified, the reader will check the rules based on Section 4.

6.3 Resolving the collision slot

According to Algorithm 1, the critical problem of CRCP is how to resolve the collision slots efficiently. In fact, decoding M_j from M is like to solve the system of linear equations. In an α – collision slot, α represents the number of variables in an equation, while the length of Manchester code d represents the number of equations. In order to simplify the procedure of resolving the collision slot, we compare each bit in the expected code M with the corresponding bit in received code M_r to decode M_j, as shown in Algorithm 2.

In Algorithm 2, *null* means an empty slot. $M(i)$ means the ith bit in the Manchester code M. $M_r(i)$ and $M_1(i), M_2(i), \dots, M_\alpha(i)$ are defined in the same way. Based on the description in Section 3, $M(i)$ equals 0 ,1 or x. It is produced as follows, $M(i) = \sum_{j=0}^{n_0} 0 + \sum_{j=1}^{n_1} 1$. n_0 and n_1 respectively represent the number of $M_j(i)$ ($j \in [1,\alpha]$) equivalent to 0 and 1 in the slot. Before the reader decodes the category IDs in the ith bit, it will use the known set $\{B(C_j)\}$ to update $M(i)$. If $B(C_j) = 0$, the reader removes M_j from M and updates $M(i)$, n_0, n_1, α. If the expected bit $M(i)$ = x, while the received bit $M_r(i) = 0$, then the category C_j with the corresponding bit equivalent to 1 is outside the scanning area, $B(C_j) = 0$. Besides, if $n_0 = 1$, the category C_j with corresponding bit equivalent to 0 must in the scanning area, $B(C_j) = 1$. If $M(i)$ = x and $M_r(i) = 1$, we can get the conclusion in the same way. However, if the expected bit $M(i) = \text{x} = 0 + \sum_{j=1}^{\alpha-1} 1$ and the received bit $M_r(i)$ = x, then the category C_j with corresponding bit equivalent to 0 gives response, $B(C_j) = 1$. If $M(i) = \text{x} = 1 + \sum_{j=1}^{\alpha-1} 0$ and $M_r(i)$ = x, we can get

Algorithm 1 CRCP: Reader Side

```
Input: $m$ rules $\{R_1, R_2, \dots, R_m\}$, $n$ related category
       IDs $\{C_1, C_2, \dots, C_n\}$, frame size $f$
$z = 0, q = 0$.
while any related category is not verified do
    Set $q = 0$, and set $c$ equal to the number of
    unverified related category IDs.
    Send the request Bloom filter.
    In a new frame, wait for the tags' responses.
    for each slot $i \in [1, f]$ and $q < c$ do
        if Manchester code cannot be recovered then
            Put these category IDs into set $C'$.
        else
            Decode the $\alpha$ – collision slot.
            Get $q_i$ new verified categories, $q = q + q_i$.
    $z = z + q$.
    if Cardinality of $C' = n - z$ then
        Verify the tags in $C'$ using probability $p_r$.
Check the rules according to the verified categories
Output: Checking result $B(R_1), B(R_2), \dots, B(R_m)$.
```

Algorithm 2 Resolving an α – collision slot

```
Input: $d$ bits expected Manchester code $M$ composed
       of $\{M_1, M_2, \dots, M_\alpha\}$, corresponding category
       IDs $\{C_1, C_2, \dots, C_\alpha\}$, received Manchester
       code $M_r$, Boolean values of $\{C_j\}$ are stored in
       $\{B(C_j)\}$.
Initialization: $B(C_j) = -1, j \in [1, \alpha]$.
if $M_r = null$ then
    $B(C_1) = 0, B(C_2) = 0, \dots, B(C_\alpha) = 0$.
    Return.
if $\alpha = 1$ then
    $B(C_1) = 1$.
    Return.
while $i \in [1, d]$ do
    $M(i) = \sum_{j=1}^{n_0} 0 + \sum_{j=1}^{n_1} 1$, $n_0 + n_1 = \alpha$.
    Use the known set $\{B(C_j) = 0\}$ to update
    $M(i), n_0, n_1, \alpha$.
    if $M(i) = \text{x}$ and $M_r(i) = 0$ then
        for each $M_j(i) = 1$ do $B(C_j) = 0$.
        if $n_0 = 1$ then
            for each $M_j(i) = 0$ do $B(C_j) = 1$.
    if $M(i) = \text{x}$ and $M_r(i) = 1$ then
        for each $M_j(i) = 0$ do $B(C_j) = 0$.
        if $n_1 = 1$ then
            for each $M_j(i) = 1$ do $B(C_j) = 1$.
    if $M(i) = \text{x}$ and $M_r(i) = \text{x}$ then
        if $n_0 = 1$ then
            for each $M_j(i) = 0$ do $B(C_j) = 1$.
        if $n_1 = 1$ then
            for each $M_j(i) = 1$ do $B(C_j) = 1$.
Output: Verified category IDs. ($\{C_j | B(C_j) \neq -1\}$.)
```

Table 1 An example for resolving a collision slot with $\alpha = 4$

Step	1	2	3	4
Expected M_j set	M_1, M_2, M_3, M_4	M_1, M_2, M_3, M_4	M_1, M_2, M_3	M_1, M_2, M_3
Expected code	$\underline{\text{x}}\text{xxxxx}$	$\text{x}\underline{\text{x}}\text{xxxx}$	$\text{xx}\underline{\text{x}}\text{xxx}$	$\text{xxx}\underline{\text{x}}\text{xx}$
Received code	$\underline{\text{x}}\text{0xxxx}$	$\text{x}\underline{0}\text{xxxx}$	$\text{x0}\underline{\text{x}}\text{xxx}$	$\text{x0x}\underline{\text{x}}\text{xx}$
M_1	$\underline{1}00101$	$1\underline{0}0101$	$10\underline{0}101$ ✓	$100\underline{1}01$
M_2	$\underline{1}01010$	$1\underline{0}1010$	$10\underline{1}010$	$101\underline{0}10$ ✓
M_3	$\underline{0}01111$ ✓	$0\underline{0}1111$	$00\underline{1}111$	$001\underline{1}11$
M_4	$\underline{1}10011$	$1\underline{1}0011$ $\times$		
$B(C_j)$	$B(C_3) = 1$	$B(C_4) = 0$	$B(C_1) = 1$	$B(C_2) = 1$

the conclusion in the same way. The reader repeats the similar process until it decodes all the α category IDs or it has repeated d times.

We give an example in Table 1, the expected Manchester code M is composed of $\{M_1, M_2, M_3, M_4\}$, the received code $M_r = \text{x0xxxx}$. $M = \text{xxxxxx}$, $M(1) = \text{x} = 0 + \sum_{j=1}^{3} 1$, only $M_3(1) = 0$. Besides, $M(1) = M_r(1) = \text{x}$. It indicates M_3 must give the response. Therefore, $B(C_3) = 1$. We can get $B(C_4) = 0$ in the same way. Then, the reader removes C_4 and updates the third bit in M. $M(3) = 0+1+1 = \text{x}$, $n_0 = 1$, $n_1 = 2$, M is composed of $\{M_1, M_2, M_3\}$, $\alpha = 3$. It repeats the similar process and gets $B(C_1) = 1$, $B(C_2) = 1$.

6.4 Performance analysis

In CRCP, the reader uses k hash functions and c category IDs to construct the Bloom filter with length l. If we require that the false positive probability is less than ϵ, we get the optimal value of l as $l = \left\lceil \frac{-\ln(\epsilon)\cdot c}{\ln(2)\cdot\ln(2)} \right\rceil$.

In each $\alpha -$ collision slot, the tag gives responses with d bits. We use $\bar{\alpha}$ and $\bar{p}_d$ represent the average value of α and p_d, respectively. τ_0 denotes the waiting time between any two consecutive transmissions, and τ_{bit} denotes the time for a tag to transmit one bit. Without loss of generality, we set $\tau_0 = 302\mu s$, $\tau_{bit} = 18.88\mu s$, $p_c = \frac{1}{2}$. Besides, $\bar{\alpha}$ should be a small number(such as 2, 3, or 4) [3], considering the actual situation. Therefore, $\bar{\alpha}$ is limited to $[1, 4]$. We get the optimal value of $\bar{\alpha}$ is $\bar{\alpha}^* = 4$, the corresponding optimal value of d is $d^* = 7$. We set $\bar{\alpha} = \bar{\alpha}^*$, $d = d^*$, the frame size $f = \frac{c}{\bar{\alpha}} = \frac{c}{\bar{\alpha}^*}$. More detail information of performance analysis can be found in the conference version of this paper [18].

7 Enhanced collision detection based rule checking protocol

7.1 Motivation

CRCP mainly focuses on verifying all the *related category IDs* in the scanning area while ignoring the rule's logical feature. In this section, we propose an Enhanced Collision detection based Rule Checking Protocol (ECRCP) in consideration of the rule's logical property. In fact, in the *OR* rule, if any category exists, the rule's Boolean value is true. In the *AND* rule, if any category is out of the scanning area, the rule's Boolean value is false. In the *EXCLUSIVE* rule, if any category is out of the scanning area, the rule's Boolean value is true. In regard to a hybrid rule, it can be determined by its subrule's Boolean value in a similar way. Therefore, when the reader gets a part of the category IDs, it can check the correlated rules. Then it only needs to check the remaining categories in the unverified rules.

7.2 Protocol description

7.2.1 Estimating the tag size

In CRCP, the reader sends the request message to tags based on Bloom filter, while ignoring the effect of tag size. When the tag size in the scanning area is much smaller than the number of *related categories*, CRCP is rather time-consuming. Thus ECRCP uses the average run based tag estimation (ART) scheme [13] to estimate the tag size. If the ratio ρ of tag size to the number of *related categories* is less than the threshold ρ^*, the reader will use APCR to check the rules. Otherwise, the reader uses the Bloom filter to inform the *related categories* to give responses and verifies the categories, as described in the following subsections.

7.2.2 Finding the popular category

Based on Section 4, a *related category* may exist in serval rules. We use frequency f_j to describe the number of times that C_j appears in different rules. If C_j appears in a different rule, f_j only adds one to its original value, no matter how many times C_j appears in this rule. For example, in the rule $R_i = (C_j \wedge C_k) \vee (C_j \vee C_u)$, f_j, f_k, f_u respectively add one. Obviously, $f_j \in [1, m]$, m is the number of rules. If a category C_j has a larger value of f_j, it means that the category affects more rules' Boolean values. ECRCP first finds the unverified popular category ID with the largest value of f_j from the unverified rules. If several categories have the same values of f_j, the reader randomly chooses one of them as the popular category. Then, it updates the frequency of each unverified *related category* in the remaining rules by eliminating the selected rules, which contain the popular category, as shown in Algorithm 3.

Algorithm 3 ECRCP: Reader Side

```
Input: $m$ rules $\{R_1, R_2, \ldots, R_m\}$, Boolean values of
       $\{R_i\}$ is $\{B(R_i)\}$, frequency of $\{C_j\}$ is $\{f_j\}$,
       popular category set is $\{C^*\}$.
Estimate the tag size $N$ in the interrogation region.
if $N < \rho^* \times m$ then
    Use ARCP to check the rules.
    Return.
Initialize $\{f_j\}$ by setting $f_j = 0$.
while any rule is not yet checked do
    Checked rules are defined as original selected
    rules, $C^* = \emptyset$.
    while any unchecked rule is not yet selected do
        for each unselected rule $R_i$ do
            if $C_j$ is unverified and $C_j$ exists in $R_i$
            then $f_j = f_j + 1$.
        Select the popular category $C_j^*$ with the
        largest value of $f_j$.
        if $C^* = \emptyset$ then $f_j^* = f_j$.
        if $C^* \neq \emptyset$ and $f_j < f_j^*$ then Break.
        Add $C_j^*$ into $\{C^*\}$, the rules containing $C_j^*$
        are selected.
    Call Algorithm 1 to verify the categories in $\{C^*\}$.
    Check the undetermined rules based on the
    verified category IDs.
Output: $B(R_1), B(R_2), \ldots, B(R_m)$.
```

7.2.3 Checking the rules

When the reader first selects the popular category ID C_j^*, it adds C_j^* into the set $\{C^*\}$. Then, it updates f_j of each unverified category C_j in the unchecked rules. The reader selects the new popular category, which has the same frequency with the the former popular category, adding it into $\{C^*\}$. It repeats the above process until no category can be added into $\{C^*\}$. The reader uses $\{C^*\}$ to produce the Bloom filter, as shown in Algorithm 3. When the reader gets the responses from the tags, it will resolve the category IDs like CRCP. Then, it checks the correlated rules. After that, the reader will only check the remaining *related categories* in the unverified rules. It repeats the similar process until all the rules are checked.

In order to describe the process well, we provide an example. We assume that $R_1 = C_1 \vee C_2$, $R_2 = C_1 \wedge C_3$, $R_3 = (C_4 \wedge C_5) \vee (C_4 \wedge C_6)$, $R_4 = \neg(C_6 \wedge C_7)$. Firstly, the reader gets $f_1 = 2$, $f_2 = 1$, $f_3 = 1$, $f_4 = 1$, $f_5 = 1$, $f_6 = 2$, $f_7 = 1$. C_1 and C_6 have the largest frequency value. It randomly selects C_1 as the popular category, $\{C^*\} = \{C_1\}$. The value of frequency in $\{C^*\}$ is $f_j^* = f_1 = 2$. Then, it updates f_j of each unverified category in the remaining unselected rules $\{R_3, R_4\}$, $f_4 = 1$, $f_5 = 1$, $f_6 = 2$, $f_7 = 1$. Obviously, C_6 is the popular category and $f_6 = f_j^*$, $\{C^*\} = \{C_1, C_6\}$. At this time, all the unverified rules are selected. The reader verifies C_1 and C_6 like CRCP. If the reader gets $B(C_1) = 1$, $B(C_6) = 0$, it can immediately conclude that $B(R_1) = 1$, $B(R_4) = 1$. After that, the reader only needs to verify the undetermined categories C_3, C_4, C_5 in the unverified rules R_2, R_3, while ignoring other categories. Obviously, ECRCP only needs to verify a part of the related categories instead of all of them.

8 Performance evaluation

We evaluate the performance of our proposed protocols by comparing them with the baseline protocols. We use the overall execution time as the performance metric.

8.1 Experiment setting

We use the following parameters [14] to configure the simulation: tag ID is 96 bits long. It takes $37.76\mu s$ for the reader to transmit one bit. Any two consecutive transmissions are separated by a waiting time $\tau_0 = 302\mu s$. It takes $18.88\mu s$ for a tag to transmit one bit. If the tag transmits d bits to the reader, the transmission time of the slot is $(302 + 18.88 \times d)\mu s$. It needs $321\mu s$ for the reader to detect an empty slot. In PRCP and BRCP, the tag can transmit one-bit information to announce its presence. The slot is about $321\mu s$. The number of tags in the interrogation region (three-dimensional physical space) is set to 5000 at most. Unless otherwise specified, we set the length of category ID to 32bits, the number of tags to 3000 and the number of rules to 300 by default. The false positive probability is set to 1×10^{-4}. Under the same simulation setting, we average the results over 100 trials.

8.2 Optimal values of $\bar{\alpha}$ and d

Based on Fig. 4, when $\bar{\alpha}$ approaches to 4 and d approaches to 7, the execution time decreases. When $\bar{\alpha}$ is too small ($\bar{\alpha} =$

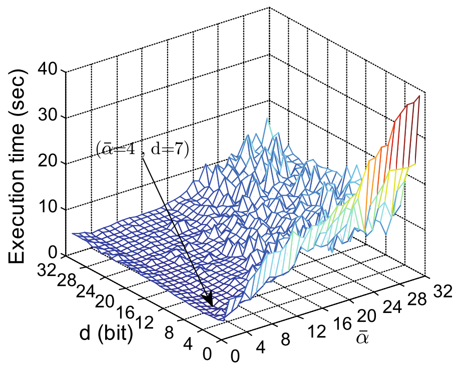

Fig. 4 Optimal values of $\bar{\alpha}$ and d

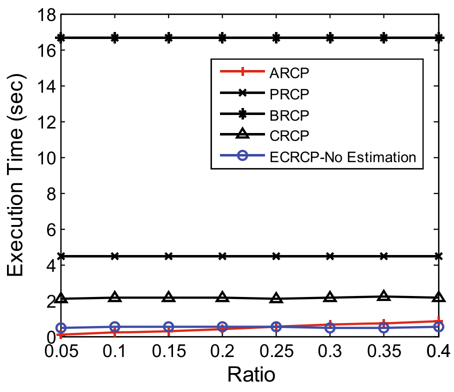

Fig. 5 Execution time under different value of ratios

2, 3), the kinds of category IDs in the slot are small and collision slots cannot be fully utilized. If $\bar{\alpha}$ is too large, the collision slots are difficult to be decoded, which wastes a lot of time. Therefore, the optimal values of $\bar{\alpha}$ and d are $\bar{\alpha} = 4$ and $d = 7$, which are not relevant to the length of category ID, the number of rules or tags, and are used in the following experiments.

8.3 Threshold value ρ^*

Figure 5 shows the execution time of each protocol while varying the ratio of ρ. *ECRCP-No estimation* means that ECRCP does not estimate the tag size and works in the way described in Section 7.2.2. In Fig. 5, when $\rho \leq 0.25$, ARCP achieves the best performance. This is because the tag size in the scanning area is much smaller than the number of *related categories*. In regard to ρ, it is not relevant to the length of category ID, the number of rules or tags. It only concentrates on the ratio of tag size to the number of *related categories*. Therefore, we set the threshold of the ratio to $\rho^* = 0.25$. In our proposed ECRCP, if $\rho \leq \rho^*$, ECRCP works as ARCP to check the rules. Otherwise, it works in the way described in Section 7.2.2.

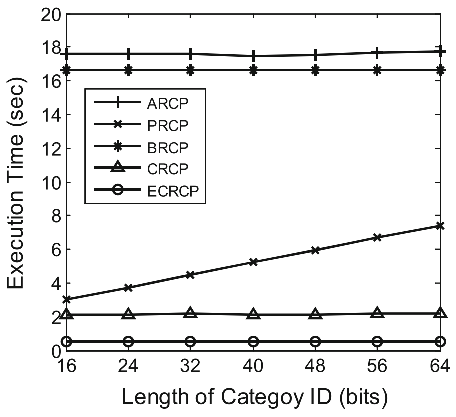

Fig. 6 Execution time under different length of C_j

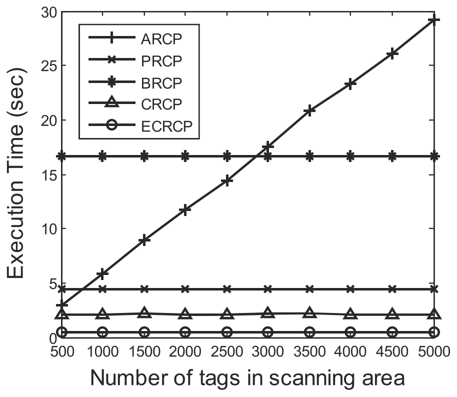

Fig. 7 Execution time under different number of tags in scanning area

8.4 Execution time comparison

Figures 6, 7 and 8 shows the execution time, CRCP and ECRCP have better performances, and ECRCP achieves the highest time efficiency. In Fig. 6, when the length of C_j is 64 bits, the execution time of ECRCP is about 0.5 seconds, which is 3 % of the time required by ARCP. The performance of ECRCP is unrelated to the length of category ID. In Fig. 7, when the number of tags is 5000, the execution time of ECRCP is about 0.5 seconds, which is 1.8 % of the time required by ARCP. The number of tags has no effect on ECRCP, because ECRCP focuses on the categories in the rules instead of those in the scanning area. In Fig. 8, when the number of rules is 500, the execution time of ECRCP is about 0.8 seconds, which is 3.1 % of the time required by BRCP. This is because ECRCP not only resolves the collision slot but also leverages the rule's logical feature. In Fig. 9, when the number of rules is 50, ECRCP only verifies 21 % of the *related category IDs*.

Figures 10 and 11 provide some fine-grained analysis about the efficiency of the protocols. Figure 10 illustrates the utilization ratio of the responses slots. CRCP and ECRCP have higher utilization ratio of responses slots than the baseline ones, because CRCP and ECRCP resolve the

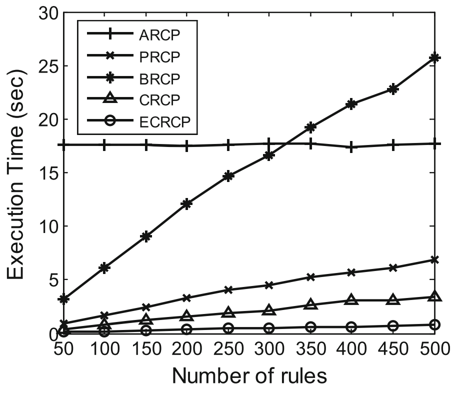

Fig. 8 Execution time under different number of rules

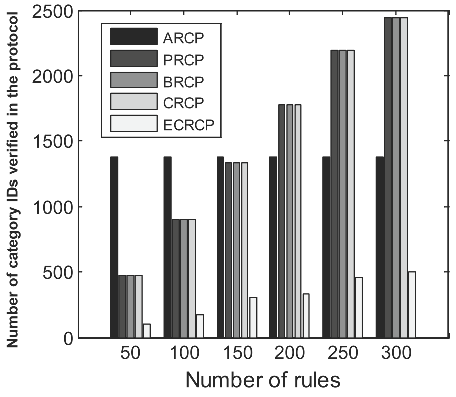

Fig. 9 Number of verified category IDs under different number of rules

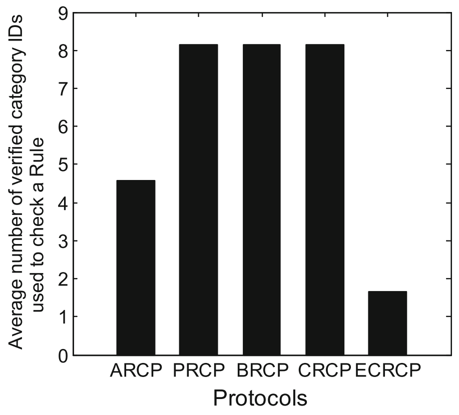

Fig. 11 Average Number of verified category IDs used to check a rule

collision slots to get more category IDs. Besides, Fig. 11 shows that ECRCP only needs about 1.7 verified *related categories* to check a rule on average, which is only 21 % of that needed in RRCP, BRCP, CRCP. This is because RCRCP verifies the most popular categories, which affect more rules' Boolean values. ECRP only verifies part of the *related categories*, which is consistent with Fig. 9. Therefore, it achieves the best performance.

8.5 Accuracy of checking all the rules

Figure 12 illustrates the accuracies of checking all the rules. The accuracies of ARCP and PRCP are higher than those of BRCP, CRCP and ECRCP. Because BRCP, CRCP and ECRCP use the Bloom filter, which has the probability of false positive. It can result in decoding the category IDs wrongly, leading to wrong result of the correlated rule. However, the accuracy of 96 % can be achieved by CRCP and ECRCP, which is high enough in many applications. Furthermore, the accuracy of 98 % can be achieved by ECRCP. When the number of rules is 300, the accuracy of ECRCP is 99.5 %. If a higher accuracy is needed, we can properly adjust the length of the Bloom filter and the number of hash functions used in the Bloom filter to meet the requirement.

9 Discussion

Bit-level collision detection is important to CRCP and ECRCP. While considering the realistic environments, detecting a collision bit can be affected by the capture effect, channel error, etc.

– Capture Effect: When tag1 sends bit '0' and tag2 sends bit '1' simultaneously, the expected mixed result is a collision bit x. However, if the signal strength of tag1 is much more strong than that of tag2, the reader is likely to get bit '0', capture effect occurs. At this time, CRCP and ECRCP may consider that tag2 is missing, leading to an error.

  Aiming to relieve capture effect, we can check the rules in a mobile way to change the distance between the tags and the antenna, which affects the tag's signal strength. If a tag can be detected at least one time, then the reader considers that it exists. Moreover, in our proposed protocols CRCP and ECRCP, there may be more

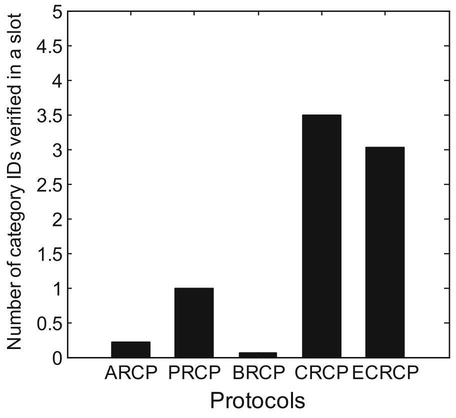

Fig. 10 Average Number of category IDs verified in a response slot

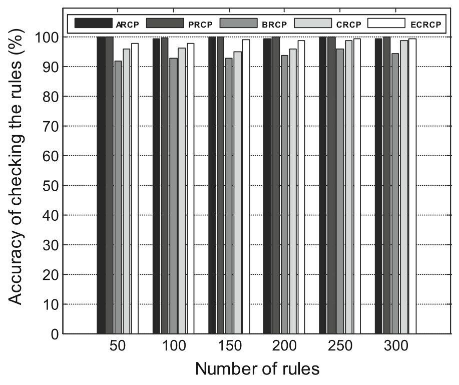

Fig. 12 Accuracy under different number of rules

than one tag in category i giving response in a slot. It will further reduce the capture effect, because the existence of a category is more easily to be detected than the existence of a tag.

– Channel Error: Channel error may corrupt the signal transmitted by a tag. The problem is common to all RFID reading protocols. Therefore, the RFID tag usually keeps transmitting its ID until it receives the acknowledgement from the reader. In our proposed protocols CRCP and ECRCP, only the bits of a slot can be recovered from the signal, the reader will resolve the collision slot, as shown in Algorithm 1. Otherwise, the tags mapping to this slot will reply in the next frame, in order to reduce the effect of channel error.

While considering the issues like capture effect, channel error, etc, there may be some potential inaccuracy of our protocols CRCP and ECRCP. However, by adopting the techniques described above, our protocols can work efficiently and reduce the effect caused by realistic environments in a degree. In future, we will provide more improvements to CRCP and ECRCP, in order to solve the problems better.

10 Conclusion

In this paper, we investigate the rule checking problem in RFID systems. We present two efficient protocols, CRCP and ECRCP. CRCP resolves the collision slots, while ECRCP further combines the collision detection and the rules' logical features to achieve time efficiency. Simulation results show that CRCP and ECRCP have better performance than the baseline protocols. Besides, ECRCP outperforms all the other solutions.

Acknowledgments This work is supported in part by National Natural Science Foundation of China under Grant No. 61100196, 61321491, 91218302; JiangSu Natural Science Foundation under Grant No. BK2011559; Key Project of JiangSu Research Program under Grant No. BE2013116; EU FP7 IRSES MobileCloud Project under Grant No. 612212.

References

1. Pan L, Wu H (2009) Smart trend-traversal: a low delay and energy tag arbitration protocol for large RFID systems. In: Proceedings of INFOCOM, mini-conference. Rio de Janeiro, pp 2571–2575
2. Xie L, Sheng B, Tan CC, Han H, Li Q, Chen D (2010) Efficient tag identification in mobile RFID systems. In: Proceedings of INFOCOM. San Diego, pp 1001–1009
3. Zhang M, Li T, Chen S, Li B (2010) Using analog network coding to improve the RFID reading throughput. In: Proceedings of ICDCS. Genova, pp 547–556
4. Zheng Y, Li M (2013) P-MTI: physical-layer missing tag identification via compressive sensing. In: Proceedings of INFOCOM. Turin, pp 917–925
5. Alotaibi M, Bialkowski KS, Postula A (2010) A signal strength based tag estimation technique for RFID systems. In: 2010 IEEE international conference on RFID-technology and applications (RFID-TA). Guangzhou, pp 251–256
6. Ning H, Cong Y, Xu ZQ, Hong T, Zhao JC, Zhang Y (2007) Performance evaluation of RFID anti-collision algorithm with FPGA implementation. In: 21st international conference on advanced information networking and applications workshops (AINAW). Niagara Falls, pp 153–158
7. Lim T, Li T, Yeo S (2008) A cross-layer framework for privacy enhancement in RFID systems. Pervasive Mob Comput 4(6):889–905
8. Liu L, Xie Z, Xi J, Lai S (2005) An improved anti-collision algorithm in RFID system. In: Proceedings of 2nd international conference on mobile technology, applications and systems. Guangzhou
9. Tan CC, Sheng B, Li Q (2008) How to monitor for missing RFID tags. In: Proceedings of ICDCS. Beijing, pp 295–302
10. Luo W, Chen S, Li T, Qiao Y (2012) Probabilistic missing-tag detection and energy-time tradeoff in large-scale RFID systems. In: Proceedings of MobiHoc. South Carolina, pp 95–104
11. Zheng Y, Li M (2011) Fast tag searching protocol for large-scale RFID systems. In: 19th IEEE international conference on network protocols (ICNP). Vancouver, pp 363–372
12. Kodialam M, Nandagopal T (2006) Fast and reliable estimation schemes in RFID systems. In: Proceedings of the 12th annual international conference on mobile computing and networking (MobiCom). Los Angeles, pp 322–333
13. Shahzad M, Liu AX (2012) Every bit counts - fast and scalable RFID estimation. In: Proceedings of the 18th annual international conference on mobile computing and networking (MobiCom). Istanbul, pp 365–376
14. Yue H, Zhang C, Pan M, Fang Y, Chen S (2012) A time-efficient information collection protocol for large-scale RFID Systems. In: Proceedings of IEEE INFOCOM. Orlando, pp 2158–2166
15. EPC radio frequency identify protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz - 916 MHz Version 1.2.0
16. Information technology automatic identification and data capture techniques-radio frequency identification for item management air interface. Part 6. parameters for air interface communications at 860- 960 MHZ, ed: Standard ISO 18000–6, (2003)
17. Wang J, Hassanieh H, Katabi D, Indyk P (2012) Efficient and reliable low-power backscatter networks. ACM SIGCOMM Comput Commun Rev 42(4):61–72
18. Yin Y, Xie L, Lu S, Chen D (2013) Efficient Protocols for Rule Checking in RFID Systems. In: Proceedings of IEEE ICCCN. Nassau, pp 1–7