Efficient Protocols for Rule Checking in RFID Systems

Yafeng Yin, Lei Xie, Sanglu Lu, Daoxu Chen
State Key Laboratory for Novel Software Technology, Nanjing University, China
Email: yyf@dislab.nju.edu.cn, {lxie, sanglu, cdx}@nju.edu.cn

***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 a large set of 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 propose two efficient protocols based on the collision detection and the logical features of rules, respectively. Simulation results indicate that our protocols achieve much better performance than other solutions in terms of time efficiency.**

I. Introduction

With the rapid proliferation of RFID technologies, RFID tags have been widely deployed into a variety of applications. In conventional applications, 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 is used to recognize the object by identifying its attached tag.

In recent years, 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 warehouse management, the lighter and the alcohol should not be close to each other in consideration of safety, while the pillow core and pillowcase should be placed together, since they are matching products. In order to verify the rules in a specified area, we can reasonably adjust the reader's 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 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, in the airport, the security checking is frequently executed and each passenger cannot stand too much time. A straightforward solution is to collect all the tag IDs from the scanning area, 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 matter whether the lighter exists. Besides, 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 according to 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. (1) 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. (2) We propose an 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 leverages the rule's logical property to effectively simplify the checking process. (3) To the best of our knowledge, this is the first theoretical work to investigate the rule checking problem in RFID systems. While leveraging the physical and application layer's information, our solution conducts a cross-layer optimization to effectively achieve time efficiency.

II. Related Work

Previous research on RFID has focused on anti-collision protocols, which can be categorized into tree-based [1][2] and ALOHA-based ones [3][4]. Tree-based protocols resolve collisions by muting subsets of tags involved in a collision. ALOHA-based protocols assign distinct transmission time slot to each tag, and sequentially identify the tags. Most of the existing work considers the collision slots unuseful and wastes the collision slots. However, the collision slot can be used to improve the RFID reading performance. In [5], the analog network coding is used to extract useful information from collision slots to improve the reading throughput. Besides, Manchester coding technology can be used in RFID communications to detect the bit collision [6][7]. In [8], Manchester coding is used to decode the tag identifier from the collision bits with the known mask. In [9][10], Manchester coding is

978-1-4673-5775-3/13/$31.00 ©2013 IEEE

used to extract the information from collision slots to enhance the identification efficiency. Instead of identifying all RFID tags, Zheng et al. [11] propose a two-phase approximation protocol for fast tag searching in large-scale RFID systems, which significantly improves the searching efficiency.

Being different from the related work, our research focuses on verifying the categories of the rules. We aim to resolve the collision slots and leverage the rules' logical features for verification in the scanning area. In order to resolve the collision slots, we will use the bit collision detection technology based on Manchester coding.

III. Preliminary

A. Synchronization

Before checking the rules, we should get the responses from tags. As a popular ID-collection protocol conforming to EPC-C1G2, frame slotted ALOHA protocol wastes the collision slot, which is considered to be useless. However, if the tags' responses are synchronized in a slot, we can resolve the collision slot to get the tags' categories. 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 [12]. According to EPC-C1G2 standard [13], the reader sends *Query* command to start an inventory round. Then each tag selects a slot to respond, the beginning of each slot is indicated by *QueryRep* or *QueryAdjust* messages. Therefore, the tags responding in the same slot can be synchronized through *QueryRep* or *QueryAdjust*.

In RFID systems, due to the limitation of the reader's power, the effective scanning range is limited. For example, the effective range of Alien 9611 antenna is limited to 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 $18.88\mu s$. 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.

B. Manchester Coding

In RFID systems, each tag encodes the backscattered data before sending it. In EPC-C1G2 standard, FM0 coding and Miller coding are used. However, the two codes are difficult to be used for detecting bit collision. In this paper, we use Manchester coding to achieve bit collision detection. Manchester coding has been used in Type B of ISO 18000-6 [14].

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 [6][7][9], as shown in Fig. 1. 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 to no transition in a received bit, such as bit3 and bit6. The corresponding bit is a collision bit. Therefore, when the two tags both send '0' (or '1'), the reader is able to recover the bit as '0' (or '1'). When a '0' and a '1' are transmitted simultaneously, the reader detects a collision bit x. We represent the above conclusion as follows, $0 + 0 = 0$, $1 + 1 = 1$, $0 + 1 =$ x ($1 + 0 =$ x). Furthermore, if there are more than two tags responding simultaneously in a slot, only when 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' to the reader). In order to decode the bits in a slot, current experimental platforms like USRP can be used [15], which is able to achieve the *bit-level* identification (0, 1, x).

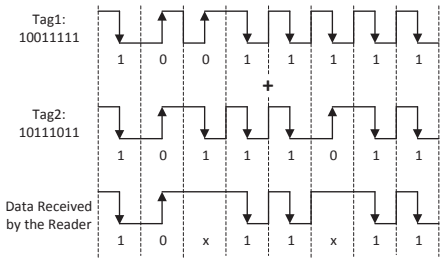

Fig. 1. Bit collision detection in Manchester code

IV. Problem Formulation

In our problem, 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 rule checking 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.

1) 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 ($B(R_i) = 1$).

2) 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, $B(R_i) = 0$.

3) 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 the large number of rules, traditional solutions are rather time consuming for collecting the tag IDs in the scanning area. Therefore, 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.

V. Baseline Protocols

A. Frame Slotted ALOHA Based Rule Checking Protocol

The frame slotted ALOHA based Rule Checking Protocol (ARCP) collects all the tag IDs in the scanning area and extracts the category IDs from them. In ARCP, the reader first broadcasts a request message, which specifies the size of the following frame. Then each tag individually and randomly selects one slot to transmit its ID. If only one tag replies in a slot, the reader can successfully receive the tag ID. If multiple tags respond simultaneously, a collision occurs, the involved tags will be acknowledged to restart in the next frame. The similar process repeats until no tags respond in the frame. Then the reader extracts the category IDs and checks the rules.

B. Polling Based Rule Checking Protocol

We call the categories in the rules as *related categories*. The Polling based Rule Checking Protocol (PRCP) checks the existence of *related categories* in the scanning area. In PRCP, the reader broadcasts a category ID and waits for the response. When a tag's category ID is equal to the reader's request one, the tag will give a short response. Then, the reader gets a nonempty slot. Otherwise, the reader gets an empty slot. The reader sends out all the *related category IDs* one by one to detect the existence of these category IDs. When the polling process terminates, the reader determines each rule's Boolean value based on the tags' responses.

C. Bloom Filter Based Rule Checking Protocol

Bloom filter based Rule Checking Protocol (BRCP) uses the Bloom filter to inform the *related categories* to give responses. Each one of the related tags in the scanning area uses k hash functions to select k slots to give a short response. Then the reader gets the tags' responses as a virtual Bloom filter and obtains the related category IDs from the responses to check the rules. In [11], a similar protocol CATS is proposed for fast tag searching in large-scale RFID systems with multiple readers rather than a mobile reader used in our system.

D. Analysis

In the baseline protocols, ARCP collects the categories in the reader's scanning area, which wastes time in collecting the unrelated tag IDs. PRCP collects the categories in the rules, which wastes time in polling some nonexistent category IDs specified by the rules. BRCP uses the Bloom filter to filter unrelated categories, and gets the responses from tags like a virtual Bloom filter. However, the unit of the virtual Bloom filter is a slot instead of a bit, which is rather time consuming, leading to the low efficiency of BRCP.

VI. Collision Detection Based Rule Checking Protocol

A. 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 III.

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 $M = \sum_{j=1}^{2} M_j = 100101 + 010111 =$ 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.

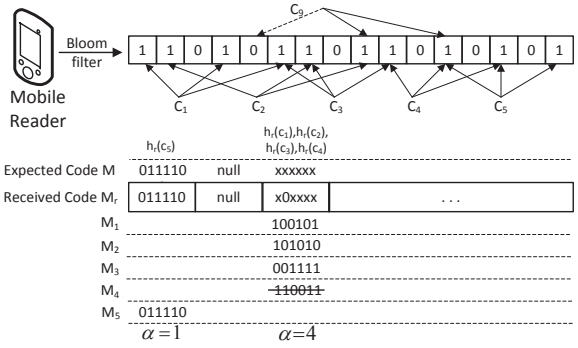

Fig. 2. Collision detection based rule checking protocol

B. Protocol Overview

In CRCP, the reader sends the request in the form of Bloom filter, as shown in Fig. 2. 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. It maps each C_j into k bits at positions $h_1(C_j)$ mod l, $h_2(C_j)$ mod l, …, $h_k(C_j)$ mod 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. 2 for an example of $k = 3$, $l = 16$, 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 do not pass the test and 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, the tag 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 $C_i, C_j, \ldots, C_k$ 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 IV.

Algorithm 1: CRCP: Reader Side

Input: m rules $\{R_1, R_2, \ldots, R_m\}$, n related category IDs $\{C_1, C_2, \ldots, C_n\}$, 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 α – 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 $B(R_1), B(R_2), \ldots, B(R_m)$.

C. 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.

Algorithm 2: Resolving an α – collision slot

Input: d bits expected Manchester code M composed of $\{M_1, M_2, \ldots, M_\alpha\}$, corresponding category IDs $\{C_1, C_2, \ldots, C_\alpha\}$, 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**

  $B(C_1) = 0, B(C_2) = 0, \ldots, B(C_\alpha) = 0$.

  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 $M(i) =$ 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 $M(i) =$ 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 $M(i) =$ x *and* $M_r(i) =$ 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. ($\{C_j | B(C_j) \neq -1\}$.)

In Algorithm 2, $null$ means an empty slot. $M(i)$ means the ith bit in the Manchester code M. $M_r(i)$ and $M_1(i), M_2(i), \ldots, M_\alpha(i)$ are defined in the same way. Based on the description in section III, $M(i)$ 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 $M(i) =$ 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 $M(i) =$ x $= 1 + \sum_{j=1}^{\alpha-1} 0$ and $M_r(i) =$ x, we can get 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 I, the expected Manchester code M is composed of $\{M_1, M_2, M_3, M_4\}$, the received code $M_r =$ x0xxxx. $M =$ xxxxxx, $M(1) =$ x $= 0 + \sum_{j=1}^{3} 1$, only $M_3(1) = 0$. Besides, $M(1) = M_r(1) =$ 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. $M(3) = 0 + 1 + 1 =$ 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 $B(C_1) = 1$, $B(C_2) = 1$.

TABLE I
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	x̲xxxxx	xx̲xxxx	xxx̲xxx	xxxx̲xx
Received code	x̲0xxxx	x0̲xxxx	x0x̲xxx	x0xx̲x
M_1	1̲00101	10̲0101	100̲101 √	1001̲01
M_2	1̲01010	10̲1010	101̲010	1010̲10 √
M_3	0̲01111 √	00̲1111	001̲111	0011̲11
M_4	1̲10011	11̲0011 ×		
$B(C_j)$	$B(C_3) = 1$	$B(C_4) = 0$	$B(C_1) = 1$	$B(C_2) = 1$

D. Performance Analysis

In CRCP, the reader uses k hash functions and c category IDs to construct the Bloom filter with length l. We use p_0 to represent the probability of any bit equivalent to 0, $p_0 = (1 - \frac{1}{l})^{kc} \approx e^{-kc/l}$. Therefore, the false positive probability of the Bloom filter is $p_f(l, k, c) = (1 - p_0)^k \approx (1 - e^{-kc/l})^k$. In order to minimize $p_f(l, k, c)$, the optimal value of k is set as $k^* = (\ln 2)\left(\frac{l}{c}\right)$. If we require $p_f(l, k, c) \leq \epsilon$, the value of l should satisfy $l \geq \frac{-\ln(\epsilon)\cdot c}{\ln(2)\cdot\ln(2)}$. We set $l = \left\lceil \frac{-\ln(\epsilon)\cdot c}{\ln(2)\cdot\ln(2)} \right\rceil$.

In the frame received from tags, we respectively use p_{b0} and p_{b1} to represent the probability that a bit in a category ID equivalent to 0 and 1, $p_{b0} = p_{b1} = \frac{1}{2}$. Each category responds with probability p_c. We use p_{bit} to represent the probability that a Manchester code M_j can be decoded by a bit $M_j(i)$ of M_j. If the bit used for decoding M_j is 0, the probability that M_j can be decoded is p_{bit0}. p_{bit1} is defined in the same way, $p_{bit} = p_{bit0} + p_{bit1}$. When $\alpha = 1$, $p_{bit} = 1$. If $\alpha > 1$, we can calculate p_{bit} by p_{bit0} and p_{bit1} as follows.

In regard to p_{bit0}, the bit used for decoding M_j is 0. Based on Algorithm 2, the ith expected bit $M(i) = \sum_{j=1}^{n_0} 0 + \sum_{j=1}^{n_1} 1$. If $n_0 = 1$, M_j can always be decoded. The probability of this situation is $p_{(n_0=1)} = p_{b0} \cdot p_{b1}^{(\alpha-1)}$. If $n_0 \in [2, \alpha - 1]$, M_j can be decoded when the received bit is $null$ or 1. It indicates that the received bit is only produced by s bits which equal to 1, $s \in [0, n_1]$. The process is like a Bernoulli process, the probability is $\binom{n_1}{s} \cdot p_c^s \cdot (1 - p_c)^{(\alpha-s)}$, $s \in [0, n_1]$. Therefore, the probability that the received bit is $null$ or 1 is $p_{(null/1)} = \sum_{s=0}^{n_1} \binom{n_1}{s} \cdot p_c^s \cdot (1 - p_c)^{(\alpha-s)}$. Meanwhile, the conditional probability that the current bit equals 0, while n_1 bits equal to 1 in the other $\alpha - 1$ bits is $p_{(con)} = p_{b0} \cdot \binom{\alpha-1}{n_1} \cdot p_{b1}^{n_1} \cdot p_{b0}^{(\alpha-1-n_1)}$. Therefore, the probability of getting M_j is $p_{(n_0\in[2,\alpha-1])} = p_{(con)} \cdot p_{(null/1)}$. If $n_0 = \alpha$, M_j can be decoded only when the received bit is $null$, the probability is $p_{(n_0=\alpha)} = p_{b0}^{\alpha} \cdot (1 - p_c)^{\alpha}$. Then, we can obtain that $p_{bit0} = p_{(n_0=1)} + \sum_{n_0=2}^{\alpha-1} p_{(n_0\in[2,\alpha-1])} + p_{(n_0=\alpha)}$. Moreover, we can get p_{bit1} in the same way. Obviously, $p_{bit1} = p_{bit0}$. Therefore, $p_{bit} = 2 \times p_{bit0} = P(\alpha)$.

The probability that a category ID C_j can be decoded in a slot is $p_d = 1 - (1 - p_{bit})^d$, d is the length of M_j. The time duration of a frame is $t_f = (\tau_{bit} \cdot d + \tau_0) \cdot f$, τ_0 denotes the waiting time between any two consecutive transmissions, and τ_{bit} denotes the time for a tag to transmit one bit. We estimate the response time t_{res} for all the category IDs as $t_{res} \approx \frac{n}{f \cdot \bar{\alpha} \cdot \bar{p}_d} \cdot t_f = \frac{(\tau_{bit} \cdot d + \tau_0) n}{\bar{\alpha} \cdot \bar{p}_d}$, where $\bar{\alpha}$ and $\bar{p}_d$ represent the average value of α and p_d, respectively. We should minimize t_{res} and it is achieved by maximizing the following function

$$T(\bar{\alpha}, d) \triangleq \frac{\bar{\alpha} \cdot \bar{p}_d}{\tau_{bit} \cdot d + \tau_0} = \frac{\bar{\alpha} \cdot \left[1 - \left(1 - P(\bar{\alpha})\right)^d\right]}{\tau_{bit} \cdot d + \tau_0}. \quad (1)$$

In order to decode the category IDs, $\bar{\alpha} \leq d$. Then we can get the optimal values of $\bar{\alpha}$ and d in the limited search space through enumeration. 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) [5], 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 $f = \frac{c}{\bar{\alpha}} = \frac{c}{\bar{\alpha}^*}$.

If $\bar{\alpha} \in [1, 4]$, the reader has a high probability for decoding C_j. However, if the number of tags is large, the mixed signal may be irresolvable. In this case, the reader verifies the categories using p_r, which is a random number in $(0, 1)$. It repeats the process until all the *related categories* are verified.

VII. ENHANCED COLLISION DETECTION BASED RULE CHECKING PROTOCOL

A. 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 rules which have not been verified.

Algorithm 3: ECRCP: Reader Side

Input: m rules $\{R_1, R_2, \ldots, R_m\}$, $\{B(R_i)\}$ is used for storing the Boolean values of $\{R_i\}$.

while *any rule is not yet checked* **do**

  for *each Rule* R_i **do**

    if *Rule* R_i *is not yet checked* **then**

      Select an undetermined category ID in R_i.

  Start a new *Query* based on the selected category IDs.

  Call Algorithm 1 to verify these category IDs.

  Check the undetermined rules based on the verified category IDs.

Output: $B(R_1), B(R_2), \ldots, B(R_m)$.

B. Protocol Description

In ECRCP, the reader successively selects the *related category IDs* from different rules to produce the Bloom filter, which aims at determining more rules' Boolean values with the verified category IDs, as shown in Algorithm 3. It selects one undetermined category ID from each rule not yet verified and resolves the category IDs like CRCP. Then, the reader checks the rules based on the verified category IDs instead of entering the next *Query Cycle*. After that, it will only check the categories in the rules with unknown Boolean values. It repeats the similar process until all the rules are checked.

In order to describe the process more clearly, we provide an example here. We assume that $R_1 = C_1 \vee C_2 \vee C_3$, $R_2 = C_4 \wedge C_5$, $R_3 = \neg(C_6 \wedge C_7)$. The reader checks C_1, C_4, C_6 first and gets $B(C_1) = 1$, $B(C_4) = 1$ and $B(C_6) = 0$. It can conclude that $B(R_1) = 1$, $B(R_3) = 1$. Then, it only needs to check the unknown category ID C_5 in R_2 while ignoring C_2, C_3, C_7. Therefore, ECRCP is more efficient than CRCP.

VIII. Performance Evaluation

In this section, we evaluate the performance of our two efficient protocols by comparing them with the baseline protocols. We use the overall execution time as the performance metric.

A. Experiment Setting

We use the following parameters [12] to configure the simulation: each tag ID is 96 bits. The rate from the reader to tags is 26.5Kb/s, it takes $37.76\mu s$ for the reader to transmit one bit. Any two consecutive transmissions are separated by a waiting time $\tau_0 = 302\mu s$. If the category ID is 32 bits long, it takes $1510\mu s$ for the reader to transmit it with a time interval included. The rate from a tag to the reader is 53Kb/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 $(302 + 18.88 \times d)\mu s$. In regard to an empty slot, it needs $321\mu s$ for the reader to detect it. In PRCP and BRCP, the tag can transmit one-bit information to the reader to announce its presence. The slot is about $321\mu s$. In regard to the number of tags in the scanning area, considering the limited effective scanning range (eg. 5-6m), the number of tags in the interrogation region (three-dimensional physical space) is set to 5000 at most. Based on the above limitation, the tag IDs and the rules are generated randomly. 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.

B. Optimal Values of $\bar{\alpha}$ and d

We first investigate the optimal values of $\bar{\alpha}$ and d in CRCP. The execution time under different values of $\bar{\alpha}$ and d is illustrated in Fig. 3. When $\bar{\alpha}$ approaches to 4 and d approaches to 7, the execution time decreases. When $\bar{\alpha} = 4, 5, 6, 7$, the protocol has the good performance. However, only when $\bar{\alpha} = 4$ and $d = 7$, the execution time gets the minimum value. When $\bar{\alpha}$ is too small ($\bar{\alpha} = 2, 3$), the kinds of category IDs in the slot are small and collision slots cannot be fully utilized. On the contrary, if $\bar{\alpha}$ is too large($\bar{\alpha} \geq 8$), the collision slots are difficult to be decoded, which wastes a lot of time. When $\bar{\alpha}$ approximates to 4 ($\bar{\alpha} = 5, 6, 7$), the kinds of category IDs in a slot are appropriate. However, there may be more than one tag in a category, the number of tags responding in the same slot can be large, causing the collision slot to be irresolvable. The problem becomes more serious with the increase of $\bar{\alpha}$. 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.

C. Execution Time Comparison

Fig. 4, 5, 6 shows the execution time, CRCP and ECRCP have better performances, and ECRCP achieves the highest time efficiency. In Fig. 4, when the length of C_j is 64 bits, the execution time of ECRCP is about 0.6 seconds, which is 3.4% of the time required by ARCP. The performance of ECRCP is unrelated to the length of category ID. In Fig. 5, when the number of tags is 5000, the execution time of ECRCP is about 0.6 seconds, which is 2.1% 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. 6, when the number of rules is 500, the execution time of ECRCP is about 0.9 seconds, which is 3.5% 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. 7, when the number of rules is 50, ECRCP only verifies 24% of the *related category IDs*.

D. Accuracy of checking all the rules

Fig. 8 illustrates the accuracies of checking all the rules. The accuracies of ARCP and PRCP are higher than those of BRCP, CRCP and ECRCP. This is because that BRCP, CRCP and ECRCP use the Bloom filter, which has the probability of false positive, which can result in decoding the category IDs wrongly, leading to wrong result of the corresponding 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

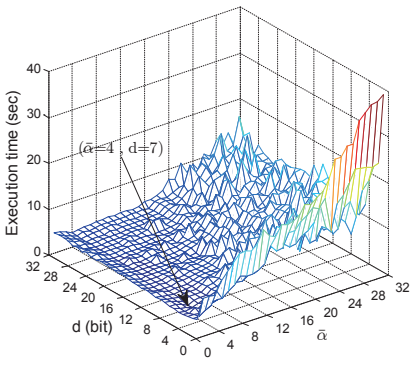

Fig. 3. Optimal values of $\bar{\alpha}$ and d.

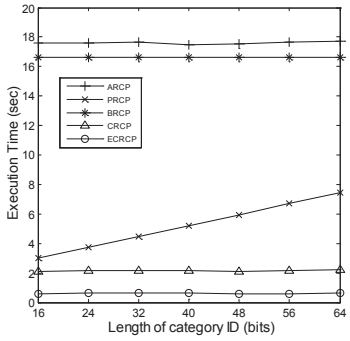

Fig. 4. Execution time under different length of category ID C_j.

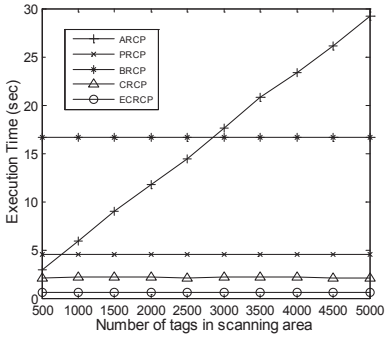

Fig. 5. Execution time under different number of tags in scanning area.

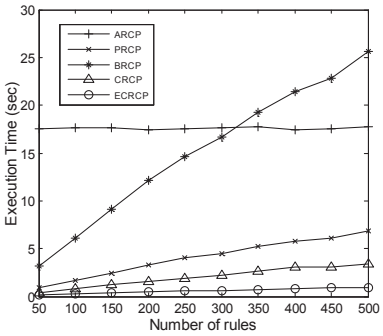

Fig. 6. Execution time under different number of rules.

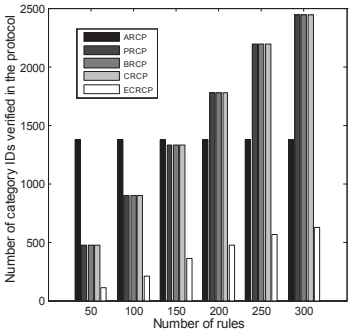

Fig. 7. Number of verified category IDs under different number of rules.

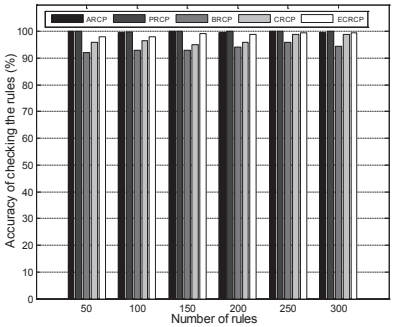

Fig. 8. Accuracy under different number of rules.

the length of the Bloom filter and the number of hash functions used in the Bloom filter to meet the requirement.

IX. Conclusion

In this paper, we investigate the rule checking problem in RFID systems. We present two efficient protocols, one is CRCP, it improves the time efficiency by resolving the collision slots. The other one is ECRCP, it combines the collision detection and the rules' logical features to achieve the time efficiency. Based on the performance evaluation, CRCP and ECRCP have better performance. Besides, the simulation results show that ECRCP outperforms all the other solutions.

Acknowledgement

This work is partially supported by the National Basic Research Program of China (973) under Grant No. 2009CB320705; the National Natural Science Foundation of China under Grant No. 61100196, 61073028, 61021062; the JiangSu Natural Science Foundation under Grant No. BK2011559.

References

[1] J. Myung, W. Lee, and S. Jaideep, "Adaptive binary splitting for efficient rfid tag anti-collision," *IEEE Communications Letters*, vol. 10, no. 3, pp. 144-146, 2006.

[2] L. Pan and H. Wu, "Smart Trend-Traversal: A Low Delay and Energy Tag Arbitration Protocol for Large RFID Systems," in *Proc. of INFOCOM, mini-conference*, 2009.

[3] B. Sheng, C. C. Tan, Q. Li, and W. Mao, "Finding popular categoried for RFID tags," in *Proc. of ACM Mobihoc*, 2008.

[4] L. Xie, B. Sheng, C. C. Tan, H. Han, Q. Li and D. Chen, "Efficient tag identification in mobile RFID systems," in *Proc. of INFOCOM*, 2010.

[5] M. Zhang, T. Li, S. Chen, and B. Li, "Using analog network coding to improve the RFID reading throughput," in *IEEE ICDCS*, 2010, pp. 547C556.

[6] H. Ning, Y. CongZ.-Q. Xu, T. Hong, J.-C. Zhao, and Y. Zhang, "Performance Evaluation of RFID Anti-Collision Algorithm with FPGA Implementation," in *21st International Conference on Advanced Information Networking and Applications Workshops*, May 2007.

[7] M. Alotaibi, KS. Bialkowski, and A. Postula, "A Signal Strength Based Tag Estimation Technique for RFID Systems," in *2010 IEEE International Conference on RFID-Technology and Applications (RFID-TA)*, June 2010.

[8] T. Lim, T. Li, and S. Yeo, "A cross-layer framework for privacy enhancement in RFID systems," *Pervasive and Mobile Computing*, Vol. 4, no. 6, pp. 889-905, December 2008.

[9] L. Liu, Z. Xie, J. Xi, and S. Lai, "An Improved Anti-collision Algorithm in RFID System," in *Proc. of 2nd International Conference on Mobile Technology, Applications and Systems*, pp. 15-17, 2005.

[10] SS. Kim, YH. Kim, SJ. Lee, and KS. Ahn, "An Improved Anti Collision Algorithm using Parity Bit in RFID System," in *Proc. of 7th IEEE Internation Symposium*, pp.224-227, 2008.

[11] Y. Zheng, and M. Li, "Fast Tag Searching Protocol for Large-Scale RFID Systems," in *19th IEEE International Conference on Network Protocols (ICNP)*, October 2011.

[12] H. Yue, C. Zhang, M. Pan, Y. Fang, and S. Chen, "A Time-efficient Information Collection Protocol for Large-scale RFID Systems," in *Proceedings IEEE INFOCOM*, 2012.

[13] EPC Radio Frequency Identify Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz - 916 MHz Version 1.2.0.

[14] "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.

[15] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk, "Efficient and Reliable Low-Power Backscatter Networks," *ACM SIGCOMM Computer Communication Review*, Vol. 42, no. 4, pp.61-72, October 2012.