Anti-collision Algorithms in Mobile Tag Scenarios: A Postprint

Authors: Li Liao, CUI Xiaohui, Li Mingzhe

Date: 2019-05-10T00:00:00+00:00

Abstract

Resolving collision problems and reducing tag identification time are of significant importance to RFID applications. To address the issue that many existing tag anti-collision algorithms are primarily designed for static tag scenarios and exhibit poor performance in mobile tag scenarios, this paper proposes an anti-collision algorithm for tag moving scenes (TMS). The algorithm first distinguishes between incoming tags and resident tags, then estimates the number of tags, and finally adopts a hybrid identification strategy for tag recognition based on the estimated value. Simulation results demonstrate that, compared with other algorithms, the TMS algorithm can effectively reduce tag identification time in tag moving scenarios, which holds certain significance for research on RFID tag anti-collision algorithms.

Full Text

Preamble

Vol. 37 No. 6
Application Research of Computers (ChinaXiv Partner Journal)

Research on Anti-Collision Algorithm in Tag Moving Scene

Li Liao[1] , Cui Xiaohui[1] , Li Mingzhe[1]
[1](a. School of Cyber Science & Engineering; b. School of Computer Science, Wuhan University, Wuhan 430072, China)

Abstract: Tag collision is one of the most common and difficult problems in RFID systems. Solving the collision problem and reducing tag recognition time are of great significance for RFID applications. To address the issue that existing anti-collision algorithms are mostly designed for static tag scenarios and perform poorly in tag moving scenarios, this paper proposes an anti-collision algorithm for tag moving scenes (TMS). The algorithm first distinguishes between move-in

tags and resident tags, then estimates the number of tags, and finally employs a hybrid recognition strategy based on the estimated tag count. Simulation results demonstrate that compared with other algorithms, the TMS algorithm can effectively reduce tag recognition time in tag moving scenarios, which holds certain significance for RFID tag anti-collision algorithm research.

Keywords: radio frequency identification (RFID); tag anti-collision; tag estimation; tag recognition time

0 Introduction

RFID (radio frequency identification) is a convenient and efficient non-contact automatic identification technology that enables communication between tags and readers through radio frequency signals to identify specific targets. With advantages such as fast identification speed, high accuracy, and low cost, RFID has been widely applied in industry [**?**]. However, when multiple tags simultaneously enter a reader' s identification range and respond, the reader cannot quickly and correctly identify the tags—a phenomenon known as tag collision [**?**]. Therefore, research on anti-collision algorithms to solve this problem is particularly important.

Constrained by existing technology, RFID tag anti-collision methods are primarily divided into tree-based algorithms \cite{3~9} and Aloha algorithms based on slotted random selection \cite{10~16}. However, these algorithms are mostly applied in scenarios with fixed tags and seldom consider situations where tags move into or out of the reader' s identification range. Literature [**?**] considered the possibility of tag movement and proposed an Adaptive Binary Splitting (ABS) algorithm, but this algorithm performs poorly when the proportion of resident tags is high. Building upon this, literature [**?**] proposed a Pair Resolution Blocking ABS (PRBABS) algorithm to further reduce the tag identification time of ABS. However, PRBABS generates numerous empty slots when the resident tag proportion is low, leading to longer identification times. Moreover, both algorithms assume that the number of tags residing within the reader' s identification range far exceeds the number of moving-in and moving-out tags, which is not applicable in many cases. Furthermore, when tags frequently move in and out of the reader' s identification range, the identification times of both algorithms are unsatisfactory, and they exhibit certain errors in tag quantity estimation.

To address these issues and reduce tag identification time, this paper proposes a Tag Moving Scene (TMS) anti-collision algorithm. By recording the reader ID and current frame ID in tags, the algorithm distinguishes between tags that have moved in and those that have remained in the reader' s identification range. It estimates the quantities of move-in and resident tags to determine whether to use ABS or PRBABS for tag identification. Through theoretical analysis and simulation experiments, the proposed algorithm requires the least identification

time in tag moving scenarios, particularly when the resident tag proportion is small, thereby solving the aforementioned problems to a certain extent.

1 Algorithm Description

The TMS algorithm consists of three main steps: first, distinguishing move-in tags from resident tags by recording reader ID and current frame ID in the tags; second, estimating the quantities of move-in and resident tags; and third, determining whether to use ABS or PRBABS for identification based on the estimated tag quantities, with the goal of minimizing tag identification time.

1.1 Basic Definitions

Assume that tags continuously move in and out of a reader' s identification range. The process of the reader identifying all tags within its range is called a frame. Tags select a slot to respond to the reader' s command and return information packets. A slot where only one tag returns a packet is called a read slot; a slot with no tag response is an empty slot; and a slot where two or more tags return packets is a collision slot. Because the reader continuously identifies tags, numerous frames are generated during the identification process.

For algorithmic clarity, we define: tags existing in frame i but not in frame *i-1* as **move-in tags**; the ratio of move-in tag quantity to the tag quantity in frame *i-1* as the **move-in tag ratio**; tags existing in both frame i and frame *i-1* as **resident tags**; the ratio of resident tag quantity to the tag quantity in frame *i-1* as the **resident tag ratio**; and tags existing in frame *i-1* but not in frame i as **move-out tags**.

1.2 Distinguishing Mechanism for Move-in and Resident Tags

Traditional tag anti-collision algorithms typically record only the reader ID, making it difficult to effectively distinguish move-in tags from resident tags. The TMS algorithm employs a method similar to that proposed in literature [**?**]: each tag has two registers, A and B, while each reader has two registers, A and C. Register A records the tag identification sequence, B records the number of identified tags in the current frame, and C records the maximum register B value among all tags. Each reader has a unique ID, and each tag contains a variable to store the reader ID. In every frame, the reader stores its own ID and current frame ID, while tags store both the reader ID they communicate with and the current frame ID. Additionally, each tag uses a variable `is_in` to indicate whether it is a move-in tag.

At the beginning of each frame, the reader broadcasts its ID and frame ID. Each tag compares its stored reader ID and frame ID with the broadcast values. If both match, the tag is identified as a resident tag and its `is_in` variable is set to 0; otherwise, it is identified as a move-in tag and `is_in` is set to 1. Subsequently, both the reader' s and tags' frame IDs are updated to the next frame' s ID,

allowing tags with unchanged reader IDs to be identified as resident tags in the next frame. This mechanism effectively distinguishes move-in tags from resident tags.

1.3 Tag Quantity Estimation Principle

The TMS algorithm estimates tag quantities using a method similar to the LOF (Lottery of Frame) approach proposed in literature [**?**]. Assuming there are 2^m tags, each tag possesses an m-bit binary ID, and a frame contains z slots ($z = 1, 2, 3, ...$) with fixed slot duration (e.g., 0.1 seconds). Taking the estimation of move-in tag quantity as an example: the reader first broadcasts number z in slot z and listens for tag responses. A tag responds if its `is_in` variable equals 1 and the position of the rightmost 1 in its ID (counted from the least significant bit) equals the broadcast number z. For instance, if a tag' s ID is 101010, since the rightmost 1 is at the 2nd least significant bit, it responds when receiving broadcast number 2. This process continues until no tags respond in slot z.

Through this method, 2^{m-1} tags respond in slot 1, 2^{m-2} tags respond in slot 2, and so on. The move-in tag quantity can be estimated using the formula $S = 1.2897 \times 2^{m-1}$. The same method is applied to estimate resident tag quantity.

1.4 Tag Identification Strategy

In the ABS algorithm, tags respond only when register A equals B. For a read slot, all tags increment A by 1; for an empty slot, all tags decrement A by 1 and the reader decrements C; for a collision slot, colliding tags randomly add a binary number to A while other tags increment A by 1, and the reader increments C. Tags with A greater than B are considered already identified. When the reader' s C becomes less than A, all tags are deemed identified. This algorithm suits scenarios with low resident tag ratios. PRBABS employs a mechanism similar to TMS to distinguish move-in and resident tags, achieving shorter identification times than ABS when resident tag ratios are high, but generating numerous empty slots when resident tag ratios are low, resulting in longer identification times.

Therefore, TMS adopts the following identification strategy: determine a threshold ratio r—use ABS when the resident tag ratio is below r, otherwise use PRBABS. According to literature [**?**], ABS reserves $0.88m$ slots for m tags and requires $2.32m$ slots to identify them, while PRBABS requires only $0.5m$ slots. The slots needed to identify resident tags in frame i using ABS are $2.32 \times$ (tag quantity in frame $i-1$) $\times$ resident tag ratio, while PRBABS requires $0.5 \times$ (tag quantity in frame $i-1$) slots. Solving the equation yields $r = 0.215517241$. Based on Section 1.3' s analysis, if $1.2897 \times 2^{z-1}$/(previous frame tag quantity) is less than the resident tag ratio, the estimated resident tag quantity is $1.2897 \times 2^{z-1}$ and ABS is used. When the resident tag ratio exceeds r, precise resident tag quantity estimation is unnecessary, reducing reader estimation time, and

PRBABS is employed.

2 Algorithm Performance Analysis

We analyze the identification time required by the TMS algorithm to identify all tags through theoretical derivation. The TMS identification process can be represented as traversing a binary tree, where each node represents tags sorted by identification order, leaf nodes represent read or empty slots, and internal nodes represent collision slots. For a binary tree of depth h, the number of nodes is 2^h. For m tags, the probability that only one tag selects the current slot (i.e., a leaf node is a read slot) is:

$$P_{\text{read}} = \binom{m}{1} \times \left(\frac{1}{2^h}\right) \times \left(1 - \frac{1}{2^h}\right)^{m-1}$$

The probability that no tags select the current slot (i.e., a leaf node is an empty slot) is:

$$P_{\text{empty}} = \left(1 - \frac{1}{2^h}\right)^m$$

After a tag collision occurs, the probability that a tag' s register B randomly increases by 0 or 1 is 1/2 each. Therefore, the probability of reaching a leaf node after h collisions is:

$$P_{\text{coll}} = \frac{1}{2^h}$$

Since slots are divided into only three types, the number of collision slots in a binary tree of depth h can be calculated through iterative computation:

$$N_{\text{coll}} = \sum_{h=0}^{\infty} \left[1 - P_{\text{coll}} - P_{\text{empty}}\right] = \sum_{h=0}^{\infty} \left\{1 - \frac{1}{2^h} - \left(1 - \frac{1}{2^h}\right)^m\right\}$$

The total number of tree nodes is:

$$N_{\text{node}} = 2 \times N_{\text{coll}} + 1$$

Assume that in a tag moving scenario, the actual quantities of move-in and resident tags are x_1 and y_1, respectively, while TMS estimates them as x_2 and y_2, consuming z slots for estimation. When the resident tag ratio is small, TMS uses ABS, requiring $0.88 \times 2 \times \sqrt{x_2 + y_2}$ slots to identify $x_1 + y_1$ tags. For any given slot, the probability that a tags respond is:

$$P_{\text{reply}} = \binom{x_1}{a} \times \left(\frac{0.88}{2 \times \sqrt{x_2 + y_2}}\right)^a \times \left(1 - \frac{0.88}{2 \times \sqrt{x_2 + y_2}}\right)^{x_1 - a}$$

Calculating from $a = 0$ to x_1, the average error in slot consumption reaches 51.84%, indicating that ABS and PRBABS cannot accurately estimate tag quantities as they always base estimates on the previous frame' s tag count. The above analysis demonstrates that TMS's tag estimation accuracy is significantly higher than that of ABS and PRBABS.

When the resident tag ratio is large, TMS employs PRBABS, consuming $0.5 \times y_1$ slots to identify resident tags and $0.88 \times 2 \times \sqrt{x_2}$ slots for move-in tags. The total slot consumption can be calculated as:

$$N_{\text{total}} = z + 0.5 \times y_1 + 0.88 \times 2 \times \sqrt{x_2}$$

[Figure 1: see original paper] Comparison of theoretical and actual values of number of slots in TMS algorithm

3 Simulation Results and Analysis

We verify the effectiveness of the TMS algorithm through computer simulation experiments, with results averaged over 100 runs under identical conditions. The experimental setup assumes m tags need identification in frame *i-1*, all being move-in tags.

Experimental parameters: Tag quantity $m = 100$, tag ID length $l = 256$ bits, per-bit transmission response time $t = 5\mu s$. The simulation first compares the theoretical and actual slot numbers required by TMS, then compares tag quantity estimation accuracy among ABS, PRBABS, and TMS, and finally calculates and compares the identification time for recognizing 100 tags in frame i by the three algorithms.

3.1 Comparison of Theoretical and Actual Slot Numbers in TMS Algorithm

Section 2 analyzed the theoretical slot numbers required by TMS. Here, we compare these theoretical values with actual slot numbers from simulation experiments to verify the correctness of the theoretical performance analysis. The move-in tag ratio is fixed at 0.5, allowing observation of slot number variations by changing the resident tag ratio. The results are shown in Figure 1.

The figure reveals that as the resident tag ratio increases from 0 to 1, both theoretical and actual slot numbers follow the same trend, remaining very close throughout—in some cases identical (e.g., at resident tag ratios of 0.3 and 0.5). Since the theoretical analysis of the binary tree in TMS involves estimations

rather than exact values, some deviation from actual values is inevitable. However, the average error is only 6.73% (2.64 slots), indicating small discrepancy and confirming the validity of the theoretical performance analysis.

3.2 Tag Quantity Estimation

The resident tag ratio is fixed at 1 to compare estimation accuracy among ABS, PRBABS, and TMS by varying the move-in tag ratio. The results are shown in Figure 2. As the move-in tag ratio increases from 0 to 1, the actual tag quantity grows linearly, while TMS' s estimated values also increase nearly linearly—matching exactly at some points (e.g., move-in tag ratios of 0.1 and 0.4). The average error between TMS estimates and actual values is only 10.48%, demonstrating high accuracy due to TMS' s LOF-based approach that estimates tag quantities based on current tag feedback. In contrast, ABS and PRBABS estimates remain constant at the configured 100 tags, deviating significantly from actual values.

[Figure 2: see original paper] Comparison of estimated number of labels and actual values by different algorithms

3.3 Comparison of Tag Identification Time Among Different Algorithms

We first analyze the performance of the three algorithms under fixed resident tag ratios while varying the move-in tag ratio. Resident tag ratios are set to 0, 0.5, and 1, with results shown in Figures 3–5.

The three figures show that as the move-in tag ratio changes from 0 to 1: when the resident tag ratio is 0, ABS consistently requires less time than PRBABS; when the resident tag ratio is 0.5 or 1, ABS outperforms PRBABS only at low move-in tag ratios (0–0.2). This occurs because PRBABS always prepares slots for move-in tags assuming their quantity matches the previous frame, generating numerous empty slots, whereas ABS identifies all tags equally without specifically reserving slots for move-in tags.

When the resident tag ratio is 0, TMS performs best for move-in tag ratios below 0.6, with identification time far shorter than the other two algorithms. When the resident tag ratio is 0.5 or 1, TMS performs best for move-in tag ratios below 0.4, after which its performance falls between ABS and PRBABS. This is because TMS provides the most accurate estimation of move-in tag quantities compared to ABS and PRBABS.

In Figure 3, the average identification times are 74.39 slots for ABS, 95.84 slots for PRBABS, and 66.34 slots for TMS. In Figure 4, the averages are 107.31 slots for ABS, 95.89 slots for PRBABS, and 94.09 slots for TMS. In Figure 5, the averages are 139.70 slots for ABS, 95.63 slots for PRBABS, and 93.73 slots for TMS. Overall, TMS achieves the shortest average identification time and delivers the best performance.

[Figure 3: see original paper] Identification time of three algorithms when the resident tag ratio=0
[Figure 4: see original paper] Identification time of three algorithms when the resident tag ratio=0.5
[Figure 5: see original paper] Identification time of three algorithms when the resident tag ratio=1

We then analyze the performance under fixed move-in tag ratios while varying the resident tag ratio. Move-in tag ratios are set to 0, 0.5, and 1, with results shown in Figures 6–8.

The three figures show that as the resident tag ratio changes from 0 to 1: when the move-in tag ratio is 0, ABS consistently outperforms PRBABS; when the move-in tag ratio is 0.5 or 1, ABS only performs better at low resident tag ratios (0–0.2) for the same reason mentioned above. Additionally, PRBABS' s identification time remains nearly unchanged because it always uses a fixed number of slots when identifying resident tags, making it insensitive to resident tag ratio variations.

When the move-in tag ratio is 0, TMS delivers the best performance with consistently far shorter identification times. When the move-in tag ratio is 0.5 or 1, TMS performs best at small resident tag ratios because it generates fewer empty slots than the other algorithms during resident tag identification, and provides more accurate move-in tag quantity estimation when move-in tag ratios are small.

In Figure 6, the average identification times are 49.50 slots for ABS, 68.31 slots for PRBABS, and 26.09 slots for TMS. In Figure 7, the averages are 105.88 slots for ABS, 90.52 slots for PRBABS, and 89.27 slots for TMS. In Figure 8, the averages are 169.64 slots for ABS, 140.32 slots for PRBABS, and 150.18 slots for TMS. TMS' s slightly worse performance than PRBABS in Figure 8 is due to its estimation error increasing with larger move-in tag ratios. Nevertheless, TMS overall achieves the shortest average identification time and the best performance.

[Figure 6: see original paper] Recognition time of three algorithms when moving-in tag ratio=0
[Figure 7: see original paper] Recognition time of three algorithms when moving-in tag ratio=0.5
[Figure 8: see original paper] Recognition time of three algorithms when moving-in tag ratio=1

4 Conclusion

This paper addresses the limitation that traditional tag anti-collision algorithms seldom consider tag moving scenarios. Through analysis of existing algorithms, we propose the TMS algorithm for tag moving scenes. The algorithm distinguishes move-in tags from resident tags by storing reader ID and frame ID on

tags, estimates the quantities of both tag types using an LOF-based approach, and dynamically selects between ABS and PRBABS based on the resident tag ratio. Theoretical analysis and simulation results demonstrate that the TMS algorithm can effectively reduce identification time in tag moving scenarios, particularly when the resident tag ratio is small.

References

[1] Lehto A, Nummela J, Ukkonen L, et al. Passive UHF RFID in paper industry: Challenges, benefits and the application environment [J]. IEEE Trans on Automation Science and Engineering, 2009, 6(1): 66-79.
[2] Fu Haolei, Lu Wangyan. Research of tag anti-collision algorithm based on multiple collision bits detection [J]. Application Research of Computers, 2018, 35(12): 3757-3765.
[3] Hai Linpeng, Wang Rui, Xiao Liying. A novel RFID anti-collision algorithm based on binary tree [J]. Journal of Networks, 2013, 8(12): 2817-2823.
[4] Lai Yuancheng, Hsiao Linyen, Chen HongJie, et al. A novel query tree protocol with bit tracking in RFID tag identification [J]. IEEE Trans on Mobile Computing, 2013, 12(10): 2063-2075.
[5] Lai Yuancheng, Hsiao Linyen, Lin Borshen. Optimal slot assignment for binary tracking tree protocol in RFID tag identification [J]. IEEE/ACM Trans on Networking, 2015, 23(1): 255-268.
[6] Liu Zilong, Ji Jinshui, Liu Caihong, et al. An anti-collision algorithm based on continuous collision bit detection [J]. Acta Electronica Sinica, 2013, 41(11): 2156-2160.
[7] Nan Jingchang, Jia Xiaomeng, Gao Mingming. Lock-bit RFID adjustive multi-tree anti-collision algorithm [J]. Application Research of Computers, 2018, 35(9): 2741-2743.
[8] Shahzad M, Alex X L. Probabilistic optimal tree hopping for RFID identification [J]. IEEE/ACM Trans on Networking, 2015, 23(3): 762-775.
[9] Zheng Jiali, Qin Tuanfa, Ni Guangnan. Tree-based backoff protocol for fast RFID tag identification [J]. Journal of China Universities of Posts and Telecommunications, 2013, 20(2): 37-41.
[10] Wu Haifeng, Zeng Yu. Bayesian tag estimate and optimal frame length for anti-collision aloha RFID system [J]. IEEE Trans on Automation Science & Engineering, 2010, 7(4): 963-969.
[11] Li Meng, Qian Zhihong, Zhang Xu, et al. Slot-predicting based ALOHA algorithm for RFID anti-collision [J]. Journal on Communications, 2011, 32(12): 43-50.
[12] Qiao Juyi, Wang Weidong, Zhang Yinghai. Matching grouping dynamic Frame Slotted ALOHA for anti-collision in RFID systems [C]//Proc of International Conference on Communication Technology Application. 2011: 796-800.
[13] Solic P, Radic J, Rozic N. Early frame break policy for ALOHA-based RFID systems [J]. IEEE Trans on Automation Science & Engineering, 2016, 13(2): 876-881.
[14] Ding Zhiguo, Lei Yingkei. Research on anti-collision algorithm based on

priority aversion [J]. Application Research of Computers, 2016, 33(3): 836-839.
[15] Liu Xiulong, Li Keqiu, Wu Jie, et al. Top-k queries for multi-category RFID systems [C]//Proc of the 35th Annual IEEE International Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2016.
[16] Chen Wentzu. Optimal frame length analysis and an efficient anti-collision algorithm with early adjustment of frame length for RFID systems [J]. IEEE Trans on Vehicular Technology, 2016, 65(5): 3343-3349.
[17] Myung J, Lee W, Srivastava J. Adaptive binary splitting for efficient RFID tag anti-collision [J]. IEEE Communications Letters, 2006, 10(3): 144-146.
[18] Lai Yuancheng, Lin Chihchung. Two blocking algorithms on adaptive binary splitting: single and pair resolutions for RFID tag identification [J]. IEEE/ACM Trans on Networking, 2009, 17(3): 962-975.
[19] Lai Yuancheng, Lin Chihchung. Two couple-resolution blocking protocols on adaptive query splitting for RFID tag identification [J]. IEEE Trans on Mobile Computing, 2012, 11(10): 1450-1463.
[20] Qian Chen, Ngan Hoilun, Liu Yunhao. Cardinality estimation for large-scale RFID systems [C]//Proc of IEEE International Conference on Pervasive Computing & Communications. Washington DC: IEEE Computer Society, 2008.
[21] Eom J B, Lee T J, Rietman R, et al. An efficient framed-slotted ALOHA algorithm with pilot frame and binary selection for anti-collision of RFID tags [J]. IEEE Communications Letters, 2008, 12(11): 861-863.

Note: Figure translations are in progress. See original paper for figures.