SCIENCE CHINA
Information Sciences

• LETTER •

June 2021, Vol. 64 169305:1–169305:3
https://doi.org/10.1007/s11432-019-2891-3

On efficient key tag writing in RFID-enabled IoT

Pengfei ZHANG, Hao LIU & Jihong YU*

School of Information and Electronics, Beijing Institute of Technology, Beijing 100081, *China*

Received 6 November 2019/Revised 12 February 2020/Accepted 22 April 2020/Published online 26 April 2021

Citation Zhang P F, Liu H, Yu J H. On efficient key tag writing in RFID-enabled IoT. Sci China Inf Sci, 2021, 64(6): 169305, https://doi.org/10.1007/s11432-019-2891-3

Dear editor,

Radio-frequency identification (RFID) can provide data in an energy-efficient and low-cost way for data-driven intelligent decisions in emerging Industrial IoT [1]. In RFID systems, tags store information of physical objects and can be attached to these objects [2–5]. Tags can capture energy in the RF signal of a nearby RFID reader for data transmission [6]. This article studies a variant of classical group writing problems. In contrast to the classical problem that writes group data to all tags in the system, the new problem deals only on how to write group data to partial tags effectively, named key tags. The existence of the key tags and the normal tags in the RFID systems is well known, and it is preferable to manage the key tags separately from the normal tags. The writing problem associated with the key tag is more challenging than the classical one since the mappings of the normal tags that are in majority in the system lead to considerable interference to the key tags and reduce time efficiency.

The tag writing is a prerequisite for effective RFID-enabled applications, such as Group-ID as a service and group password update. Yet, little effort has been devoted to this cornerstone service. The protocol named concurrent grouping (CCG) [7, 8] exploits the effective slots that are selected by multiple tags of the same group to build an indicator vector. This vector can inform each tag when it should receive its group data. The CCG has to transmit the whole indicator vector and thus wastes much time on the transmission of non-effective slots that are in majority. Multiple seeds are used in [9] to increase the ratio of the effective slots at the cost of high computational complexity. None of the prior studies can address the key tag writing problem efficiently for two reasons: they cannot eliminate the interference of the normal tags and a large body of communication overhead is wasted on the transmission of noneffective slots in the indicator vector.

To solve the key tag writing problem, we present a protocol named K-Write. K-Write can reduce the activities of the normal tags while compressing the reader-to-tag indicator vector. The superiority of K-Write is in two-folds: the active normal tags in the existing studies can be deactivated in our work so that the interference is reduced and only the slots useful to inform key tags of when they should receive their group data are transmitted instead of all slots in the existing studies. This significantly reduces transmission cost.

Proposed protocol. Consider the RFID system of k key tags and n normal tags. The k key tags are divided into G disjoint groups referred to as key groups. The size of key group i ($1 \leqslant i \leqslant G$) is g_i. Our interest is to address the key tag writing problem of devising a protocol to send data of each key group only to all its members (key tags) accurately within the minimum time.

(1) Protocol description. To address the key tag writing problem, we use multiple hashing operations to build a Bloom filter to deactivate normal tags for interference reduction and construct a compressed filter to reduce transmission of noneffective slots. The proposed protocol named K-Write consists of two phases referred to as filtering phase and writing phase, respectively.

Phase 1. To begin, the reader maps k key tags into an l-bit null array using the h hash functions with seed s_1 and set the value of the mapped positions to "1"s. After hashing all the key tags, the reader constructs a Bloom filter defined as BF. The parameter configuration will be analyzed shortly.

The reader broadcasts the BF, its size (i.e., frame length) l, h, and s_1 to all the tags. Each tag then maps its ID to h slots pseudo-randomly at positions $\mathrm{hash}_1(l, \mathrm{ID}, s_1)$, $\mathrm{hash}_2(l, \mathrm{ID}, s_1), \ldots, \mathrm{hash}_h(l, \mathrm{ID}, s_1)$, and checks the corresponding positions in the BF. If all of the h positions are "1"s, then the tag regards itself as a key tag. Otherwise, it is normal and then remains silent in the rest of the protocol. After all the tags conduct a membership test, all key tags and partial normal tags will be active since Bloom filter has no false-negative and has only false-positives that tags not used to build the Bloom filter may also pass the test.

Phase 2. It operates in multiple rounds each with two steps, namely, the preparatory step and the writing step. The first step is to inform the key tags of their corresponding pure slots and the group data is then sent to the ready key tags in the second step.

Consider an arbitrary round j in the execution of Phase 2.

* Corresponding author (email: jihong.yu@bit.edu.cn)

Let K_j and $g_{i,j}$ denote the number of the remaining unwritten key tags in the system and unwritten key tags of the group g_i at the beginning of this round, respectively. Denote by G' the number of the groups with unwritten key tags.

Preparatory step. First, the reader offline builds a compressed filter. Given the frame size f_j, the reader maps each active tag with one hashing operation to an f_j-bit null vector and sets the value of pure slots that are selected by the key tags of the same groups to "1"s while keeping empty and non-effective slots "0"s. After constructing the f_j-bit original vector, the reader starts compressing it instead of directly broadcasting the whole original vector in the previous studies, which is more time efficient. For compression, we use the distance between two "1"s to indicate the positions of "1"s, i.e., pure slots. To do this, the reader finds the longest segment of the consecutive zeros, the length of which is defined as z_0, and replaces each segment of the consecutive zeros with a decimal value of its length in $\lceil \log(z_0+1) \rceil$ bits. Consequently, only the pure slots are recorded in the compressed filter and are transmitted, thereby reducing communication overhead.

The reader sends the original vector size f_j, segment length $\lceil \log(z_0+1) \rceil$ and the compressed filter to all the active tags. On receiving these, a tag calculates the decimal value of each disjoint $\lceil \log(z_0+1) \rceil$-bit segment assumed to be D and knows there are D consecutive zeros with one "1" followed. Repeat this for all segments, a tag can learn all the positions of the value "1" among $[1, f_j]$. It then checks from the compressed filter whether its hashed slot is a pure slot. Specifically, the tag hashes itself to a slot among $[1, f_j]$. If the corresponding position is "1", it is hashed to a pure slot and will receive its group data in this round. Otherwise, it does not participate in the second step and waits for the start of the next round.

Writing step. After the preparatory step, only the key tags that are hashed to pure slots can participate in this step and are ready to receive their group data. Since each key tag that selects a pure slot in the original vector can learn the number of "1"s before its chosen position, assumed to be q, then it will wait for its group data at $(q+1)$-th slot in the writing frame. Let S_j be the number of "1"s (pure slots) in the original vector, the reader starts a writing frame of S_j slots. For any slot in this frame, the reader knows which group of key tags is waiting at this slot and sends their associated group data. Meanwhile, each key tag passing the compressed filter knows at which slot the reader will transmit its group data, and can thus, receive the data at that slot.

After the current round, the reader moves to the next round, which is identical except that the written key tags will remain silent. The above process repeats round after round until all key tags receive their corresponding group data.

(2) Performance optimization and parameter configuration. We examine how to configure the used parameters in the protocol to minimize the communication overhead spent in completing the key tag writing task.

Parameter configuration in Phase 1. Phase 1 aims to sift out the normal tags; the communication overhead is thus equal to the length of the broadcast Bloom filter. Owing to the fact that Bloom filter has no false-negatives, we only need to consider its false-positive rate, denoted as P_{fp}; we have $P_{\text{fp}} = (1-e^{-kh/l})^h$.

The total time spent in this round can thus be calculated as $l \times t_1$, where t_1 denotes the time needed by the reader to transmit one bit to tags. We denote C_1 as the average time cost consumed to reduce a normal tag; it can be expressed as $C_1 = -t_1 kh/(n(1-P_{\text{fp}})\ln(1-P_{\text{fp}}^{\frac{1}{h}}))$. The C_1 is minimized when $h = -\log P_{\text{fp}}$. Therefore, the execution time T_1 of Phase 1 and the expected number n' of normal tags still active after Phase 1 can be calculated as $T_1 = kht_1/\ln 2$ and $n' = n0.5^h$.

Parameter configuration in Phase 2. Since Phase 2 would operate round by round, we optimize the time efficiency of each round for feasibility analysis. Consider an arbitrary round j, the number of active key tags is K_j, and there are still $g_{i,j}$ key tags in the group i. The execution time of round j, denoted as $T_{2,j}$, comprises the time to transmit the compressed filter and the group data. Let f'_j be the compressed filter size in the round j, and denote by S_j the number of the pure slots in the original vector, which is also the frame size in the writing step. We define w_j as the number of key tags receiving their group data in this round, then the time efficiency η_j of round j can be expressed as $\eta_j = w_j/(f'_j t_1 + S_j d t_1)$, and we can derive the optimum f_j maximizing η_j.

Theorem 1. When the time efficiency η_j of round j reaches its maximum value, the corresponding f_j must fall into the interval $\left[1, \frac{(K_j+n')^2}{0.5K_j+n'}\right]$ where $K_j \leqslant k$ is the number of active key tags at the beginning of round j, and $n' \leqslant n$ is the number of active normal tags after Phase 1.

Proof. Refer to Appendix A.

Next, we introduce how to set the number h of hash functions by analyzing its impact on the overall time cost; K_j and n' may not be available for offline computation, then we can set them with known system parameters to $K_j = k$ and $n' = n$ because the upper bound f_j^* increases with both K_j and n' and the value of n' has no impact on the monotonicity of η_j with respect to f_j. Therefore, we can derive the approximate optimum f_j offline for each round of Phase 2 following Theorem 1. Consequently, η_j, w_j, f'_j, and $T_{2,j}$ are the function of only n', and the execution time of Phase 2, defined as T_2, is determined by n'. Recall in Phase 1, the overall execution time of the K-Write also depends on h. Intuitively, a higher h leads to a longer Phase 1, but it reduces interferences of normal tags to key tag writing so that Phase 2 is reduced. Therefore, there exists an optimum h minimizing the overall execution time of K-Write. We can obtain such an optimum h through the method explained in Appendix B.

Performance evaluation. We evaluated the performance

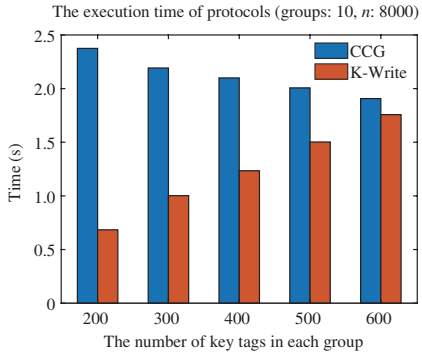

Figure 1 (Color online) The number of key tags v.s. the execution time.

of the proposed protocol in terms of the execution time in comparison with the CCG. The simulation parameters are specified in Appendix C. We displayed how the number of key tags influences the execution time of protocols. To this end, we simulated scenarios with k varied from 2000 to 6000 where the tags are evenly partitioned into $G = 10$ groups. The simulation results are shown in Figure 1 where we observe that the K-Write performs better than the CCG even when the number of the key tags exceeds that of the normal tags. Specifically, the K-Write can perform 4× better when compared to the CCG.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61901035), Beijing Institute of Technology Research Fund Program for Young Scholars, Young Elite Scientist Sponsorship Program by CAST, and Chongqing Key Laboratory of Mobile Communications Technology.

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Ning H S, Liu H. Cyber-physical-social-thinking space based science and technology framework for the Internet of Things. Sci China Inf Sci, 2015, 58: 031102
2 Liu X L, Chen S, Liu J, et al. Fast and accurate detection of unknown tags for RFID systems-hash collisions are desirable. IEEE/ACM Trans Netw, 2020, 28: 126–139
3 Yu J H, Gong W, Liu J C, et al. Missing tag identification in COTS RFID systems: bridging the gap between theory and practice. IEEE Trans Mobile Comput, 2020, 19: 130–141
4 Liu X L, Zhang J W, Jiang S, et al. Accurate localization of tagged objects using mobile RFID-augmented robots. IEEE Trans Mobile Comput, 2019. doi: 10.1109/TMC.2019.2962129
5 Yu J H, Gong W, Liu J C, et al. On efficient tree-based tag search in large-scale RFID systems. IEEE/ACM Trans Netw, 2019, 27: 42–55
6 Gao X Z, Niyato D, Wang P, et al. Contract design for time resource assignment and pricing in backscatter-assisted RF-powered networks. IEEE Wirel Commun Lett, 2020, 9: 42–46
7 Liu J, Chen M, Xiao B, et al. Efficient RFID grouping protocols. IEEE/ACM Trans Netw, 2016, 24: 3177–3190
8 Liu J, Xiao B, et al. Fast RFID grouping protocols. In: Proceedings of the 34th Conference on Computer Communications, Hong Kong, 2015. 1948–1956
9 Yu J H, Liu J C, Zhang R R, et al. Multi-seed group labeling in RFID systems. IEEE Trans Mobile Comput, 2019. doi: 10.1109/TMC.2019.2934445