Practical Key Tag Monitoring in RFID Systems

Jihong Yu*, Wei Gong*, Jiangchuan Liu†, Lin Chen†, Fangxin Wang*, Haitian Pang‡
*School of Computing Science, Simon Fraser University, Canada. Email: {jihongy, gongweig}@sfu.ca
†College of Natural Resources and Environment, South China Agricultural University, China. Email: csljc@ieee.org
†LRI-CNRS UMR 8623, University of Paris-Sud, France. Email: chen@lri.fr
‡Department of Computer Science and Technology, Tsinghua University, China. Email: pht14@mails.tsinghua.edu.cn

***Abstract*—With rapid development of radio frequency identification (RFID) technology, ever-increasing research effort has been dedicated to devising various RFID-enabled services. The key tag monitoring, which is to detect anomaly of key tags, is one of the most important services in such important Internet-of-Things applications as inventory management. Yet prior work assumes that all tags are armed with hashing functionality and a reader would report channel states in every slot, which is not supported by commercial off-the-shelf (COTS) RFID tags and readers. To bridge this gap, this paper is devoted to enabling key tag monitoring service with COTS devices. In particular, we introduce two anomaly monitoring protocols to detect whether there is any key tag absent from the system. The first protocol employs Q-query that works in an analog frame slotted Aloha paradigm to interrogate tags and collect tag IDs. An anomaly event will be found if at least one key tag ID is not present in the collected ones. To reduce time cost of the first protocol resulted from tag collisions, we present a collision-free method that uses select-query to specify a key tag to reply in each slot. Once there is no response in a slot, the specified key tag is regarded as a missing tag. We conduct experiments to evaluate two protocols.**

I. Introduction

Recent years have witnessed an unprecedented development of the radio frequency identification (RFID) technology. An RFID system typically consists of readers and tags. A reader is a device equipped with a dedicated power source and an antenna, and can collect and process information sent by tags in its interrogation region. A tag, on the other hand, is a low-cost microchip labeled with a unique serial number (ID) to identify an object. The core characteristic of a tag is that it is battery-free and can capture energy from the radio wave of its nearby reader to backscatter messages. As a promising technology, RFID is widely used in various applications ranging from inventory control, supply chain management and logistics [3] to object tracking and location [2].

In an RFID system, one may be more interested in partial tags instead of the whole tag population. We define these tags as key tags. Here, we present two examples to motivate the presence of key tags in practice.

- In a retail store with expensive and inexpensive goods, an RFID system is deployed to monitor them. Staffs more care about expensive ones due to higher value, thus the tags on these objects are expected to be detected more frequently.
- Consider a warehouse rented to multiple companies, when a company requests tag anomaly service, warehouse management only needs to detect tags attached on products from this company instead of all tags in the region.

The work of L. Chen is supported by the project CNRS PEPS (Projet Exploratoire Premier Soutien) on "Objets communicants: algorithmes, architectures et application" (OCAAA) under the project MIRFID: Management of Infrastructural RFID systems: algorithm design and implementation.

In this paper, we study key tag anomaly monitoring problem which is to detect whether some key tags are no more present in the coverage of RFID system. A large body of work has been proposed to address this problem. The nature of these work is that the reader predicts response slot of each tag and could find an absent one if a slot expected to be single does not see any reply. Although technical tools, such as multiple-seed method and sampling method, are very promising in accelerating the detection process, the reality is that the tag response slot cannot be predicted a prior due to the lack of hashing functionality in COTS tags. Therefore, monitoring key tag anomaly while being compatible with COTS RFID devices is still an open problem.

In this paper, we introduce two protocols that are able to find key tag anomaly while being compatible with existing COTS RFID devices. Specifically, in the first protocol named Q2, we employ Q-query to interrogate the tags. Q2 works in an analog frame slotted Aloha paradigm, which is the de facto random access protocol in the Gen2 standard [1]. To avoid tag collisions as well as interferences of non-key tags in Q2, we then design a select-query based protocol (SQ) that can singularize the tags in each slot with a selective bitmask and ensures successful communication in all slots. We implement Q2 and SQ with COTS RFID devices.

II. Problem Definition and Proposed Solutions

Consider an RFID system composing of a reader and n tags containing k key tags. The reader knows all tag IDs [1]. *The key tag anomaly monitoring problem is to detect the existence of key tag absence.* Here, execution time that is measured as the time spent achieving the task is the most important metric.

A. Q2: Q-query based solution

The reader conducts Q-query to collect tag IDs. The query can stop when the reader reads all key tags, meaning there is no absent key tag; otherwise, the query process continues until the execution of the whole frame. At the end, the reader compares the collected key tag IDs with those recorded in the

[1] In this paper, each tag is assigned an electronic product code (EPC) that is stored in the tag memory and is used to distinguish different tags.

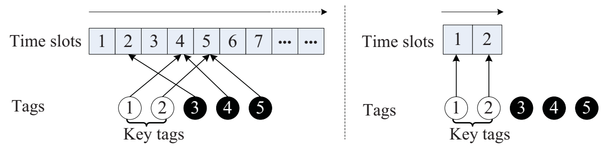


Fig. 1. Comparison of Q2 and SQ for multiple tags. Q2 would waste some slots that are collided (slots 4 and 5) or empty (slots 1, 3, 6, and 7). Moreover, the non-key tags interfere with key tags, such as slots 4 and 5. While SQ can selectively read key tags and only needs two slots.

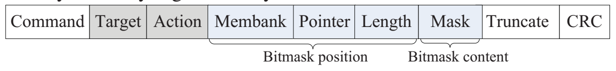


Fig. 2. Format of *Select* command: `MemBank`, `Pointer` and `Length` specify the target bit string that tags need to search for in its memory; `Mask` is the bitmask that tags will compare with their individual target bit strings.

database. If some recorded key tag IDs are not present in the collected set, then these key tags are absent.

Q-query operates as follows: The reader first broadcasts a *Query* command containing tag-to-reader link rate and frame size. Tags use these parameters to determine their encoding methods and their individual response slots by picking a random number smaller than frame size as their slot counters. If a counter is equal to 0, this tag responds to the reader immediately with a 16-bit random number (RN16); otherwise it will stay silent. On correctly decoding *RN16* from a tag, the reader sends *ACK* packaging the decoded *RN16* to acknowledge this tag. If the tag confirms the correctness of the reader-to-tag *RN16*, it will send its *ID* to the reader. After that, the reader transmits *QueryRep* to instruct tags to decrement their slot counters and the tags whose counters are equal to 0 reply with another *RN16*, indicating the start of a new slot. It is worth noting that If the reader does not receive an *RN16* correctly, it transmits *QueryRep* to initiate a new slot.

B. SQ: Select-query based solution

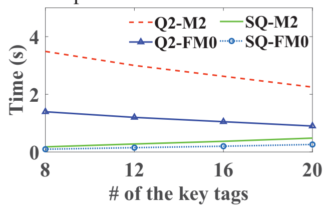
Since Q2 is a random protocol, multiple tags may reply with *RN16* simultaneously leading to the failure of decoding *RN16* at the reader side. To deal with tag collisions in Q2, we turn to SQ that is able to singularize tags in every slot. The difference of Q2 and SQ are shown in Figure 1.

SQ operates as follows: The reader first sends a *Select* command that specifies a bitmask. On receiving *Select*, each tag checks whether it matches the reader-to-tag bitmask. Only the matching tags will wait for the further query of the reader. We will explain shortly how to choose the bitmask such that only one tag can pass the bitmask comparison in a slot. Then the reader transmits a *Query* that specifies the matching tag to reply. Because only one tag meets the requirement in SQ, this tag is the only one responding in this slot with its *RN16*. After that, the reader issues an *ACK* with the decoded *RN16* and waits to receive the *ID* of this tag. The reader will repeat the above process until all key tags are probed.

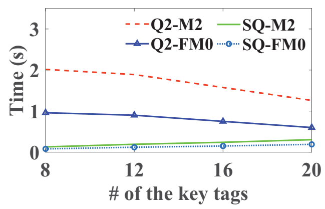
The core of SQ is *Select* function, which is used to inform tags of which bit string in their individual memory should be compared with the received mask. There are six mandatory fields in the *Select* command that is shown in Figure 2, we will mainly introduce the four fields. 1) Filed `MemBank`: This field indicates the memory model where a tag will search for

```
String targetEpc1 = System.getProperty(SampleProperties.targetTag1);
if (targetEpc1 != null) {
        TagFilter t2 = settings.getFilters().getTagFilter2();
        t2.setBitCount(16);
        t2.setBitPointer(0x20);
        t2.setMemoryBank(MemoryBank.Epc);
        t2.setFilterOp(TagFilterOp.Match);
        t2.setTagMask("06DD7F27437B193326BA3F35");
        settings.getFilters().setMode(TagFilterMode.OnlyFilter2);
        System.out.println("Matching 1st 16 bits of epc " + targetEpc1);}
```

Fig. 3. Implementation of *Select* command in Java: `MemBank` $= 0$, `Pointer` $= 0$, `Length` $= 16$, and `Mask` $=$'06DD'. In this example, tags will compare the first 16 bits of their individual IDs with the mask '06DD'.



(a) Missing key tag ratio $\alpha = 0.25$



(b) Missing key tag ratio $\alpha = 0.5$

Fig. 4. Performance comparison with different numbers of key tags under two tag-to-reader rates.

a target bit string to compare with the received mask. The EPC memory bank is used in this paper, i.e., `MemBank` $= 1$. 2) Field `Pointer`: This field points the starting address of target bit string in the used memory model. 3) Field `Length`: This field suggests the length of the target bit string. 4) Field `Mask`: The mask that tags will compare their target bit strings with is recorded here. In this paper, we use each key tag ID as mask. An illustrative implementation of *Select* function is shown in Figure 3.

III. Evaluation

We use one ImpinJ reader and 30 ImpinJ tags in our implementation. The transmission power of the reader is set to 20 dbm, and its reception sensitivity is -60 dbm. We examine two tag-to-reader encoding methods: FM0 and Miller 2, corresponding to the tag-to-read link rates of 640kbps and 320kbps, respectively. We assume there are $k = 8:4:20$ key tags and the number of missing key tags is equal to $\alpha \cdot k$ where α is set to 0.25 and 0.5. As shown in Figure 4, SQ significantly performs better than Q2 as it only probes the key tags instead of the whole in SQ while eliminating collisions.

This paper studies how to use COTS RFID devices to monitoring key tag anomaly. In the future, we will use compact bitmask instead of whole tag ID, and combine multiple *Select* for more functionalities. Moreover, we will also design COTS-device-friendly protocols for other RFID-enabled services.

References

[1] EPCglobal Inc. Radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 mhz - 960 mhz version 1.0.9. [Online], 2005. Available: http://www.gs1.org.

[2] J. Han, C. Qian, X. Wang, D. Ma, J. Zhao, P. Zhang, W. Xi, and Z. Jiang. Twins: Device-free object tracking using passive tags. In *IEEE INFOCOM*, pages 469–476, 2014.

[3] RFID Journal. The RFID application in logistics and supply chain management, 2009. Available: http://www.etsi.org/deliver/etsi_en/302200_302299/30220801/01.01.01_20/en_30220801v010101c.pdf.