Bringing the Functionality of Tag Sampling to Reality for COTS RFID Systems

Qiwen Hu, Jiuwu Zhang, Sheng Chen, Xin Xie, Xiulong Liu, Keqiu Li, and Jie Wu

***Abstract*—Tag sampling is required by many promising RFID applications to read only a part of tags instead of all for better time-efficiency. However, the functionality of tag sampling is not actually supported by the COTS RFID devices. This paper proposes the Multi-array Tag Sampling (MTS) scheme, in which the interactions between reader and tags are clearly specified, to bring tag sampling from theoretical assumption to reality. We use Impinj R420 reader and Monza 4QT tags to implement a prototype to validate the feasibility of MTS. Extensive experimental and simulation results reveal MTS can sample the tags with predefined probabilities, meanwhile performing better in terms of space- and time-efficiency than the state-of-the-art protocol.**

***Index Terms*—RFID, TDMA, C1G2 standard, tag sampling.**

I. Introduction

The Radio Frequency Identification (RFID) technology has been widely used in various applications such as smart warehouse and traceable logistics. An RFID system typically consists of a large number of tags, several readers, and a powerful backend server. Tags with unique IDs are normally attached to objects. The server controls an RFID reader to scan and read the IDs from nearby tags, thereby enabling automatic identification, tracking, and monitoring applications.

Tag collision is a serious issue in the RFID research field, because numerous tags within a reader's interrogation range share and contend for the same wireless channel. As a matter of fact, identifying all tags in the system really takes time. For better time-efficiency, many sampling-based approaches are proposed to only read a part of tags instead of all, then perform the probabilistic analytics on the collected tag information, *e.g.*, tag cardinality estimation [1] and missing tag detection [2], [3]. Unfortunately, these sampling-based schemes cannot be applied in reality, because they commonly made an assumption that RFID tags can be sampled with a given probability while such a functionality of tag sampling is still in the theory stage and not available on Commercial Off-The-Shelf (COTS) RFID tags. Moreover, in the existing sampling-based schemes, all tags are assumed to be read with the same sampling probability, which is not reasonable because tags are often attached to various types of items and their reading frequencies should coincide with the values of the attached items, *e.g.*, the tags on expensive items should be read more frequently when monitoring the theft events.

In this paper, we investigate how to enable the functionality of tag sampling for C1G2-compliant RFID devices, thereby bringing the desirable functionality of tag sampling from the theoretical assumption to reality. A simple solution [4] is to implement the tag sampling operation by writing a long binary array on tag memory to represent the sampling probability. The ratio of bits '1' in the array indicates the sampling probability. Although simple, such a Single-array Tag Sampling (STS) scheme incurs too much memory consumption on tags, *e.g.*, 1000 bits are required if the granularity of sampling probability is 0.001. To reduce the memory cost on tags, we propose the Multi-array Tag Sampling (MTS) scheme, in which a fine-grained sampling probability is transformed into the product of multiple coarse-grained probabilities, each of them is represented by a short binary array (the ratio fo '1's in an array equals the corresponding probability). Thus, unlike STS that uses a very long binary array, the MTS scheme uses multiple short binary arrays to represent a fine-grained sampling probability. We validate the correctness and efficiency of the proposed MTS scheme though both prototype experiments and large-scale simulations. Extensive results show that MTS can not only guarantee the predefined sampling probabilities, but also significantly outperforms the basic STS scheme in terms of memory cost and time-efficiency.

II. Multi-array Tag Sampling (MTS) Scheme

A. Preliminary Knowledge of C1G2 Standard

The EPC C1G2 standard [5] is an industry standard that specifies the physical and logical requirements and available commands for the RFID systems that operate within the frequency range of 860 MHz∼960 MHz. To be deployable on C1G2-compliant RFID readers and tags, the application protocols must follow the following specifications in C1G2 standard. Some preliminaries about C1G2 are as follows. Each tag holds four non-volatile memory banks for storing inventory management information. Among them, EPC memory and User memory are two major banks used in this paper. EPC memory stores the 96-bit electronic product code, which is a global unique identifier for labeling the tagged item. User memory is used for storing user-defined data, such as production date, product price and supply chain information. In MTS, the binary arrays are stored in the User memory.

B. Motivation of MTS and Scheme Overview

A straightforward solution to enable the functionality of tag sampling is to inject a binary array to each tag, in which the

Q. Hu, J. Zhang, Sheng Chen, and K. Li are with the College of Intelligence and Computing, Tianjin University, Tianjin, China.

X. Liu is with the School of Computing Science, Simon Fraser University, Canada.

X. Xie is with the School of Computer Science and Technology, Dalian University of Technology, Dalian, China.

J. Wu is with the Department of Computer and Information Sciences, Temple University, USA.

Corresponding Authors: Keqiu Li and Xiulong Liu

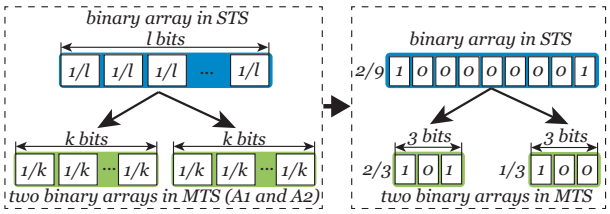

Fig. 1. An illustration of the Multi-array Tag Sampling (MTS) scheme.

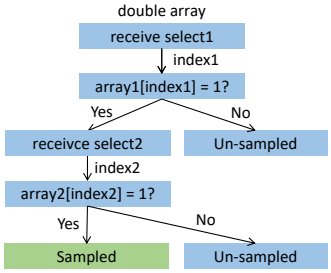

Fig. 2. The flowchart of the MTS scheme

fractions of bit '1's represents the sampling probability. Then, the reader broadcasts a `Select` command, which specifies an address of tag memory and a binary mask containing a single bit '1'. Upon receiving the `Select` command, each tag compares the mask with the binary array from the specified address in its memory. If the x-th bit in the mask is '1' and the x-th bit of the binary array from that address in its memory is also '1', this tag will be activated and respond to the reader's interrogation. Although simple, the above Single-array Tag Sampling (STS) scheme costs too much bits on the tag side to represent a fine-grained sampling probability. Specifically, if the required granularity of the sampling probability is α, the length of the binary string should be at least $\frac{1}{\alpha}$ bits, *e.g.*, it requires 1000 bits to represent a sampling probability with the granularity of 0.001. Writing a long binary array to tags has two deficiencies. First, an RFID tag has limited User memory space for storing other user-defined information, *e.g.*, a Monza 4QT tag only has a 512-byte User memory. Thus, we desire to reduce the length of binary array, that needs to be written to tag memory, to reduce memory consumption. Second, writing a long array to tag memory via a low rate wireless channel takes much time, especially when the number of tags is large. As specified in the C1G2 standard, a `Write` command can write only 16-bit data to tag memory, thus a long binary array requires several times writing operations in sequence. Hence, a more efficient representation of the sampling probability is desirable to save the tag memory and reduce the writing time.

To this end, we propose a Multi-array Tag Sampling (MTS) scheme to reduce the length of binary arrays for representing a fine-grained sampling probability. Different from STS, the MTS scheme represents the sampling probability using multiple binary arrays. Although some advanced schemes (*e.g.*, IEEE 754 [6]) can represent the floating number in a very space-efficient way, it is hard to use the `Select` command to resolve the probability in IEEE 754 formatting. For easy implementation using the `Select` commands, the MTS scheme still uses the fraction of '1's in a binary array to represent the probability, *e.g.*, if there are six '1's in a 10-bit binary array, the corresponding probability is 0.6. In MTS, each binary array only needs represent a coarse-grained probability and thus can have a much shorter length. Compared with the straightforward STS scheme, the MTS scheme significantly reduces the memory consumption from $\frac{1}{\alpha}$ to $\frac{n}{\sqrt[n]{\alpha}}$. Since commercial reader SDK only supports two-filter operations, our MTS scheme uses two binary arrays in each tag memory to represent the sampling probability. As exemplified in Fig. 1, the basic STS scheme uses a 9-bit array to represent the sampling probability of $\frac{2}{9}$, while the MTS scheme just uses two 3-bit binary arrays to represent $\frac{2}{3}$ and $\frac{1}{3}$, respectively. The memory cost on tag side is reduced from 9 bits to 6 bits. As illustrated in Fig. 2, the reader uses two `Select` commands to filter the tag by comparing the embedded masks with the binary arrays in tag memory. Suppose the two binary arrays are both k bits. The index specified in the `Select` command should be randomly chosen within $[0, k-1]$. In this example, the tag will pass the filtering operation with a probability of $\frac{2}{3} \times \frac{1}{3}$, which still equals the required sampling probability $\frac{2}{9}$. To enable the tag-sampling with a granularity of $\alpha \in (0, 1)$, the memory consumption is expected to be just $\frac{2}{\sqrt{\alpha}}$, *e.g.*, when $\alpha = 0.01$, MTS only uses $\frac{2}{\sqrt{0.01}} = 20$ bits; when $\alpha = 0.0001$, MTS only uses $\frac{2}{\sqrt{0.0001}} = 200$ bits.

C. Details of MTS

The proposed MTS scheme consists of two stages: probability assignment and sampling query. In the RFID-enabled warehousing management, the probability assignment stage (i.e., writing arrays to tags) usually happens when moving the tagged items into the warehouse for the first time. The sampling probability should be determined by the users according to the category of the attached products. A specific field of tag ID can be used to indicate the category of the attached product. Such kind of category information can help the users determine the sampling probabilities of tags before moving the tagged products into the warehouse. Before the sampling query stage (i.e., routine tag-sampling identification), the binary arrays are already injected into the tags. Next, we will present the details about these two stages.

1) Probability Assignment: In the stage of probability assignment, we need to assign a specific sampling probability to each tag. For a certain tag id, we first transform the required fine-grained sampling probability into the product of two coarse-grained probabilities denoted as p_1 and p_2, *e.g.*, $0.15 \Rightarrow 0.3 \times 0.5$. Then, the two probabilities p_1 and p_2 will be represented by two k-bit binary arrays, respectively. Here, the array length k depends on the required granularity of the sampling probability α, *e.g.*, when $\alpha = 0.01$, $k = \frac{1}{\sqrt{\alpha}} = 10$. In the first k-bit binary array $\mathtt{A}_1[..]$, the ratio of '1's should be equal to the probability p_1; and in the second k-bit binary array $\mathtt{A}_2[..]$, the ratio of '1's should be equal to the probability p_2. To guarantee the randomness of the tag-sampling process, we *randomly* choose the bits of arrays and set them to '1's. Then, the reader issues the `Write` commands to write these two binary arrays $\mathtt{A}_1[..]$ and $\mathtt{A}_2[..]$ to two addresses a_1 and a_2 of the user memory of the target tag. To save memory

space of tags, it is better to transform a sampling probability into two probabilities with the same granularity. For example, 0.0125 can be transformed into 0.125×0.1 or 0.25×0.05. To represent 0.125×0.1, we need to store a 1000-bit string and a 10-bit string in tag memory. In contrary, to represent 0.25×0.05, we only need to store two 100-bit strings in tag memory. Obviously, the latter one is more memory-efficient. By default, we assume a specific area of user memory in each tag is used for storing arrays $\mathtt{A}_1[..]$ and $\mathtt{A}_2[..]$. And a_1 and a_2 in tag user memory are the same across all tags.

When writing the two binary arrays to a tag, the reader needs to issue a sequence of C1G2 commands.The `Select` command is to activate the target tag and deactivate all the other tags, because the reader can only write one tag at a time. The `Query` command is to start an inventory round for tag identification because only the identified tags can be accessed by the reader. The tag replies a `RN16` for collision detection. If no collision is detected, the reader will issue an `ACK` command to collect the tag EPC. Then, the reader issues an `Req_RN` command to access the memory of the target tag. This tag will reply a handler as the access permission. This handler received by the reader will be embedded in the As specified in C1G2 standard, a `Write` command can only write 16-bit data to tag memory. Hence, if the binary array $\mathtt{A}_1[..]$ (or $\mathtt{A}_2[..]$) is longer than 16 bits, it will be divided into 16-bit segments and several `Write` commands are needed to write them all. For writing a segment of $\mathtt{A}_1[..]$ (or $\mathtt{A}_2[..]$), the reader will send a `Write` command embedded with the valid handler, the array segment, and the corresponding address in user memory to the target tag. Upon receiving such a `Write` command, the target tag will store the array segment to the specified address in its user memory. After writing all array segments as above, the binary array $\mathtt{A}_1[..]$ and $\mathtt{A}_2[..]$ representing p_1 and p_2 will be correctly stored at the address a_1 and a_2 in the user memory. Using the above methods, we assign the required sampling probabilities to the other tags in the system. Note that, the stage of probability assignment is only performed once for tags that are newly moved into the system.

2) Sampling Query: The sampling query stage is to sample tags with the pre-assigned sampling probabilities. The reader broadcasts two `Select` commands to filter the tags by comparing the embedded masks and the two binary arrays $\mathtt{A}_1[..]$ and $\mathtt{A}_2[..]$ in user memory. Specifically, the first `Select` command contains a single-bit mask '1', the address of the i-th bit of the array $\mathtt{A}_1[..]$ in user memory, where i is a random number within $[1, k]$. Upon receiving the first `Select` command, each tag checks if the bit with specified address in its first array $\mathtt{A}_1[..]$ is '1'. If so, the tag will pass the filtering of the first `Select` command. For a tag, the probability that it passes the filtering of the first `Select` command should be equal to the ratio of '1' in $\mathtt{A}_1[..]$, *i.e.*, p_1. The details about the second `Select` command are similar with the above description. Finally, for each tag, the probability that it passes the filtering of two `Select` command should be equal to $p_1 \times p_2$, which is exactly the expected sampling probability. The tags passing through the filtering process will respond to the reader's interrogation.

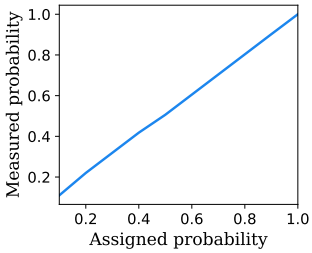

Fig. 3. Probability validation

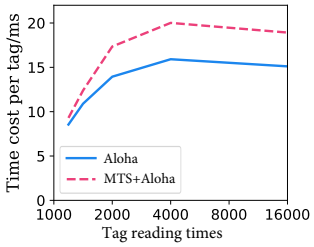

Fig. 4. Time-efficiency validation

Our MTS scheme aims at sampling tags with given probabilities. Hence, we cannot guarantee that all tags are identified in a single round of tag inventory process. Fortunately, we normally need to repeatedly perform the tag inventory process for multiple times in many practical scenarios, e.g., continuous theft-detection. Thus, each tag has a chance to be identified during a long monitoring period, just with different frequency.

III. Implementation

We validate the feasibility of our MTS scheme by implementing a prototype system, which consists of a Lenovo PC, an Impinj R420 reader [7], and several Monza 4QT tags [8]. The RFID reader is connected to the PC via an Ethernet cable. The PC uses the LLRP protocol [9] to manipulate a reader to broadcast the C1G2 commands. The LLRP messages, represented in XML format, are transmitted to the reader to control the interactions between reader and tags. The reader communicates with the tags through the Ultra High Frequency (UHF) wireless channel (900Mhz) following the C1G2 standard [5]. We leverage the Octane SDK.NET [10] to develop our MTS. The Octane SDK.NET provides APIs for sending C1G2 commands and frees us from the tedious work for defining LLRP messages.

IV. Experiments and Simulations

In this section, we will evaluate the sampling accuracy and time-efficiency of the proposed MTS scheme. Since we are the first to implement the tag sampling operations for C1G2-complaint RFID systems, we mainly compare the performance of MTS with that of the basic STS scheme.

A. Sampling Accuracy

We first validate the accuracy of sampling probabilities through prototype experiments. We place 10 tags in the front of the reader antenna. Each tag is assigned a distinct sampling probability ranges from 0.1 to 1.0. We run 1000 rounds of tag sampling operations, and the results in Fig. 3 demonstrate that the measured sampling probabilities match well with the assigned sampling probabilities.

B. Time-efficiency

After running MTS to sample tags, we need to invoke the tag identification protocol (e.g., Aloha in EPC) to perform the tag monitoring query. In the real experiments, we use the reader to repeatedly identify 10 tags until $16,000$ readings are obtained. In Fig. 5, we report the time cost of $1000 \sim 16000$ tag readings. We compare the time cost of MTS+Aloha with that of using Aloha alone, and find that MTS brings additional overhead of about $5ms$ per tag reading. Fortunately, this kind

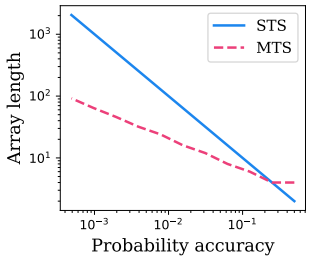

Fig. 5. Array length vs. accuracy

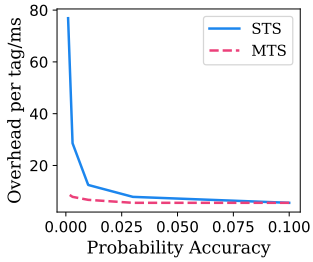

Fig. 6. Assignment cost vs. accuracy

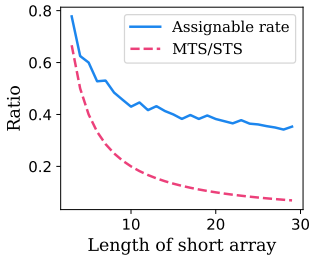

Fig. 7. Assignable rate vs. length

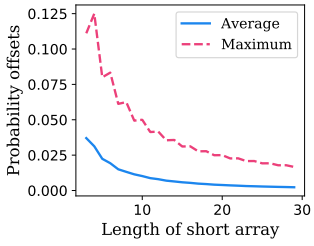

Fig. 8. Coverage rate vs. length

of overhead can be reduced in a large-scale RFID system containing thousands of tags. The reasons are explained as follows. The time complexity of the sampling query stage is $\mathcal{O}(1)$, as the reader only broadcasts two `Select` commands. The time complexity of the monitoring query stage is $\mathcal{O}(n)$, as the reader needs to identify n tags. Hence, the relative overhead caused by MTS is $\frac{\mathcal{O}(1)}{\mathcal{O}(1)+\mathcal{O}(n)}$. When the number of tags is small (e.g., $n = 10$ in the experiments of Fig. 5), the overhead caused by MTS is notable. In a large-scale RFID system containing thousands of tags, the overhead caused by MTS will decrease. Although MTS inevitably begets some overhead, it can help achieve real fairness among tags. Using the current Aloha protocol in EPC C1G2 standard, no matter what kind of products (expensive or cheap) the tags are attached to, they has the same probability to be identified. In practical theft-monitoring scenarios, we obviously want to read the tags of expensive products more frequently. With our MTS scheme, tags can be read with predefined sampling probabilities. In other words, limited wireless channel resources can be allocated to tags according to their values or priorities.

Next, we compare the performance of our MTS scheme with the basic STS scheme through simulations. As shown in Fig. 5, much less memory usage is a major advantage of the MTS scheme. For example, when the granularity of the sampling probability is set higher than 0.001, the memory cost of STS is nearly 50 times that of the MTS. Less memory cost is also equivalent to higher time-efficiency for writing binary arrays to tags. As shown in Fig. 6, the STS scheme causes much more time than MTS when writing sampling probabilities to a large number of tags. For example, when the granularity of sampling probability is 0.001, the STS scheme needs nearly 6 minutes to assign the sampling probabilities to 5000 tags.
of memory consumption by MTS and that by STS. We find

As aforementioned, a major deficiency of the MTS scheme is that two binary arrays in the user memory cannot always represent a sampling probability. We refer to *assignable rate* as the ratio of sampling probabilities that can be represented by two k-bit arrays. In Fig. 7, the curve for MTS/STS means the improvement ratio in memory utilization, i.e., the ratio that the proposed MTS significantly outperforms STS in terms of memory utilization. Fig. 7 also shows that the assignable rate decreases with the increase of the array length. This is because the production of two arrays produces a large number of duplicate probabilities when k is large. In the MTS scheme, the un-assignable sampling probabilities should be replaced by the closest assignable values. To evaluate the offsets of the assigned probabilities, we randomly and uniformly assign a sampling probability to 10000 tags. The simulation results in Fig. 8 show that, both the maximum offset and average offset decreases as the increase of array length k. This is because the assignable rate gets larger with the increase of array length.

V. Conclusion

In this paper, we proposed the Multi-array Tag Sampling (MTS) scheme for the C1G2-complaint COTS RFID systems. Our MTS scheme is expected to benefit various promising RFID application protocols that need to perform statistical analytics on sampled tags. The experimental results obtained by running the prototype demonstrate the feasibility and correctness of the MTS scheme. Extensive simulation results reveal that the proposed MTS scheme can significantly outperform the state-of-the-art Single-array Tag Sampling (STS) scheme in terms of both space- and time-efficiency.

Acknowledgment

This work is supported in part by the National Key Research and Development Program of China No. 2018YFB1004703, in part by the State Key Program of National Natural Science of China under Grant No. 61432002, in part by the NSFC-Guangdong Joint Funds under Grant No. U1701263, in part by the the NSFC-General Technology Basic Research Joint Funds under Grant No. U1836214, in part by the National Natural Science Foundation of China under Grant No. 61772251, 61672379, 61772112 and 61702365, in part by the Natural Science Foundation of Tianjin under Grant No. 17JCQNJC00700, 17ZXRGGX00150 and 18ZXZNGX00040, and in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1629746, CNS-1651947, and CNS 1564128.

References

[1] X. Liu, X. Xie, K. Li, B. Xiao, J. Wu, H. Qi, and D. Lu, "Fast Tracking the Population of Key Tags in Large-scale Anonymous RFID Systems," *IEEE/ACM Trans. on Networking*, vol. 25, no. 1, pp. 278–291, 2017.

[2] X. Liu, K. Li, G. Min, Y. Shen, A. X. Liu, and W. Qu, "A Multiple Hashing Approach to Complete Identification of Missing RFID Tags," *IEEE Trans. on Communications*, vol. 62, no. 3, pp. 1046–1057, 2014.

[3] C. C. Tan, B. Sheng, and Q. Li, "Efficient Techniques for Monitoring Missing RFID Tags," *IEEE Trans. on Wireless Communications*, vol. 9, no. 6, pp. 1882–1889, 2010.

[4] X. Xie, X. Liu, and et. al., "Implementation of differential tag sampling for cots rfid systems," *IEEE Transactions on Mobile Computing*, 2019.

[5] *EPC UHF Gen2 Air Interface Protocol for Communications at 860MHz-960MHz, EPCglobal*, July 2018.

[6] *IEEE 754*. [Online]. Available: https://en.wikipedia.org/wiki/IEEE_754

[7] *R420 RFID reader*. [Online]. Available: https://support.impinj.com/

[8] *Impinj Monza 4 UHF RFID Tag Chips*. [Online]. Available: https://support.impinj.com/hc/en-us/articles/

[9] *LLRP Protocol*. [Online]. Available: https://www.gs1.org/standards/epc-rfid/llrp/1-1-0

[10] *Octane SDK.NET for Impinj RAIN RFID readers*. [Online]. Available: https://www.nuget.org/packages/OctaneSDK/2.28.1