Retwork: Exploring Reader Network with COTS RFID Systems

Jia Liu and Xingyu Chen, *Nanjing University;* Shigang Chen, *University of Florida;* Wei Wang, Dong Jiang, and Lijun Chen, *Nanjing University*

https://www.usenix.org/conference/atc20/presentation/liu-jia

This paper is included in the Proceedings of the 2020 USENIX Annual Technical Conference.

July 15–17, 2020

978-1-939133-14-4

Retwork: Exploring Reader Network with a COTS RFID System

Jia Liu[1] Xingyu Chen[1] Shigang Chen[2] Wei Wang[1] Dong Jiang[1] Lijun Chen[1]
[1]*State Key Laboratory for Novel Software Technology, Nanjing University, China*
[2]*Department of Computer & Information Science & Engineering, University of Florida, USA*

Abstract

Radio frequency identification has been gaining popularity in a variety of applications from shipping and transportation to retail industry and logistics management. With a limited reader-tag communication range, multiple readers (or reader antennas) must be used to provide full coverage to any deployment area beyond a few meters across. However, reader contention can seriously degrade the performance of the system or even block out some tags from being read. Most prior work on this problem requires hardware and protocol support that is incompatible with the EPC Gen2 standard. Moreover, they assume the knowledge of a reader network that precisely describes the contention relationship among all readers, but the efficient acquisition of the reader network in a practical system with commercial-off-the-shelf (COTS) tags is an open problem. This study fills the gap by proposing a novel protocol *Retwork*, which works under the limitations imposed by commercial Gen2-compatible tags and identifies all possible reader contentions efficiently through careful protocol design that exploits the flag-setting capability of these tags. We have implemented a prototype with 8,000 commercial tags. Extensive experiments demonstrate that *Retwork* can reduce communication overhead by an order of magnitude, in comparison to an alternative solution.

1 Introduction

Radio frequency identification (RFID) has been gaining popularity in a variety of pervasive applications, including library inventory [7,13,16,25,26], warehouse control [6,17,21,22,24,38–41], supply chain management [12,15,19,20,23], and object tracking [9,18,29,31–34,36]. Given that tags use backscattering to communicate with readers, the communication distance between a reader and a tag is limited to a few meters. In a deployment (e.g., a retail store or a warehouse) that goes beyond the communication range of a single reader antenna, multiple reader antennas (referred to simply as *readers* for convenience) must be used to cover the whole area. If the readers take turn to communicate with the tags in their respective interrogation zones, then this condition will not be time efficient for inventory operations. On the contrary, if they operate simultaneously, a complex situation of collisions, where tags in overlapped areas will be left unread, may be created. To solve this dilemma, research has been trying to find solutions to properly schedule readers so that only those that do not collide will be active at any time.

This reader scheduling is built upon the knowledge of *reader network*, which is a graph that depicts the contention relationship among the readers and underlies many multi-reader protocols [5,8,10,14,27,28,30,35,37,42]. However, in practice, a reader can hardly know the size of its own interrogation zone, which takes an irregular shape that is difficult to determine due to directivity of reader antenna and environment reflection. More difficulty is to determine whether any two readers contend, which happens when their interrogation zones overlap and at least one tag in the overlapped area exists. To make reader scheduling practical, in this paper, we work under the limitations of commercial Gen2-compatible RFID systems and propose a solution to determine their reader network, without any modification to tags or reader-tag communication protocols, and without any assumption of pre-knowledge about the shape and size of any reader's interrogation zone. One naive solution is to activate the readers in sequence, one at a time, to collect tag IDs in its zone, and then compare the tag sets of any two readers to see if the intersection is empty. If it is, then no contention exists between the two readers; if not, there is a contention link between them in the network. However, this serialized approach is inefficient and already collects all tag IDs, which makes deriving the reader network unnecessary.

This paper proposes a new protocol called *Retwork* that efficiently determines the reader network of a large RFID system, with two key advantages. First, it avoids the need to perform inventory over the entire tag set by the native solution that activates one reader at a time (which reads its tags, one at a time). Second, it is completely compatible with the worldwide standard of EPC Gen2 [4], allowing the new protocol to be deployed in commercial systems. The idea behind *Retwork* is not to read all tags and compare the readers' tag sets, but to make each reader broadcast certain information to tags in its zone. The Gen2 standard does not allow us to write a reader's ID to all tags in its zone at once. Had this been supported, after all readers did that, tags would know whether they are in the overlapped areas of multiple readers and would then report that to their readers. Fortunately, according to the Gen2 standard, a reader's transmission can flip certain flags in the memory of all tags that receive the transmission. With a careful design, we show that these flags can be exploited to fully support identification of all contention relationship among the readers. Our protocol only requires each reader to transmit a few Gen2 commands that trick its tags to flip their flags in a certain way. Thus, after all readers transmit, tags in overlapped areas will have their flags set differently from other tags. Tags will signal the readers for contention relationship without having to deliver their IDs to the readers. The execution time of *Retwork* is a function of the number of readers, instead of the number of tags, which is much larger in practical systems. The major contributions of this study are listed below.

• We propose an efficient solution *Retwork* to the practically important problem of identifying the contention relationship among multiple readers in a large RFID system, which underlies a majority of multi-reader protocols.

• Our protocol exploits the flag-setting capability in Gen2. With a carefully-designed series of flag-flipping operations, our protocol can classify tags into groups: those under reader contention and those free of contention.

• We implement a prototype of *Retwork* with 8,000 commodity RFID tags. Extensive experiments show that it boosts the read throughput of the system and thus cuts the inventory time by an order of magnitude.

2 Problem Formulation

An RFID system generally consists of a large number of tags and multiple readers. Each tag is attached to an object to exclusively indicate the associated object. The tag set is denoted by $\Gamma = \{t_1, t_2, ..., t_n\}$. A reader is surrounded by a finite space within which it can communicate with tags. This space is referred to as the *interrogation zone* (or *read zone*) of that reader. In a multi-reader RFID system, the layout of readers constitutes a *reader network*, which is represented by a graph $\mathcal{G} = (V, E)$, where $V = \{v_1, v_2, ..., v_m\}$ is the set of vertices (readers) and E is the set of edges (contention links). An edge $(v_i, v_j) \in E$ exists between the reader v_i and the reader v_j if and only if at least one tag $t_k \in \Gamma$ is located at the overlapped interrogation zone covered by both readers. At this point, these two readers are called *neighbors* or *adjacent nodes*, which may incur collision if they communicate concurrently. The tags within the overlapped zone are referred to as *contentious tags*. Notably, forming an edge requires two necessary conditions: overlapped read zone and contentious tags. Two readers with the same read zone are still treated as collision-free if no tags reside in such a zone. This is reasonable because no contention will happen even if the two readers run in parallel. Note that, various factors, such as the reader planning, multi-path effects, and material of tagged objects, greatly affect a reader's signals and thus make the shape of its interrogation zone irregular. Our protocol design is robust to any kinds of RFID systems, regardless of the shape of the interrogation zone.

3 Tag Inventory

Exploring the reader network is essentially to check whether a contention link exists between any pair of readers. An intuitive solution is to conduct tag inventory reader by reader over the whole tag set. In particular, each reader individually queries the tag subpopulation in the field of view. Upon receiving a query request, all tags report their tag IDs to the reader by running the Gen2 protocol. To avoid reader collision, all readers need to execute the tag inventory sequentially rather than concurrently. That is because these readers do not have any prior knowledge on the reader network; a collision is very likely to take place when concurrent inventory is conducted. This blocks out some tags in the overlapped zone from being read. After one-round inventory by all readers, each reader can learn its neighbors by comparing its own tag list with others'. If two readers share a common tag subset, an edge must exist between them; otherwise, they are conflict-free. By checking all pairs of readers, the reader network $\mathcal{G} = (V, E)$ is formed finally. This solution is foolproof but suffers from high latency. The reason is that all tags have to transmit their long tag IDs to readers, resulting in at least $n \times t_{id}$ time overhead, where n is the number of tags and t_{id} is the time delay for transmitting a tag ID. This process is extremely time consuming, especially in a large RFID system.

4 Retwork

4.1 Basic Idea

The basic idea of *Retwork* is piggybacking some payload in a reader's instruction via one-to-many broadcasting over the air and taking a few tag replies instead of all as the indicator to obtain the link information between readers. By this means, most tag inventories are avoided and the identification time of reader network is determined by only the small number m of readers rather than the number n of tags ($m \ll n$), greatly improving the protocol efficiency. Following this idea, we propose *Retwork* in embryo, which gives us a clue to the protocol design and reveals the key hurdle that limits the implementation of *Retwork* on Gen2. It consists of two phases: *over-the-air writing* and *selective reading*. The former is to tell each tag in which readers' interrogation zones the tag resides via one-to-many broadcast. The latter chooses a specific tag subset to reply. By checking the tag responses, the reader is able to learn whether two readers conflict.

Over-the-air Writing. This phase is composed of m time slots, where m is the number of readers. Each reader is assigned an exclusive slot and scheduled to transmit its reader index in that slot. Without loss of generality, we suppose that the reader v_i is assigned to the i-th slot. The reader v_i broadcasts its index i to all tags in its vicinity during the i-th slot. Clearly, only the tags within the reader's interrogation zone can hear this broadcast. By recording reader indices, a tag knows at which readers' interrogation zones it is located. More specifically, each tag holds an m-bit indicator vector in its memory, which is denoted by I and initializes to zeros at first. Once a tag receives an index of value i, it sets the i-th bit of the vector to '1', that is, $I[i] = 1$, which indicates that this tag is under v_i's coverage. A contention link between v_i and v_j is formed if two bits $I[i]$ and $I[j]$ meet $I[i] = 1$ and $I[j] = 1$, where $1 \leq i < j \leq m$. By checking all indicator vectors and converging these pieces of information together, we can obtain the reader network $\mathcal{G}(V,E)$. However, although tags know all of this, the readers do not. We next introduce the second phase that aims to extract the contention information from tags and shed light on the reader network.

Selective Reading. Directly collecting each tag's indicator vector can obtain the reader network but suffers from long time delay as tag inventory. To improve time efficiency, the reader chooses only a few tags to reply each time and removes most of dispensable memory accesses. In particular, each reader in turn checks whether it conflicts with others. Consider any one reader v_i, $1 \leq i \leq m$. To obtain the link status between v_i and v_j $(j > i)$, the reader v_i first selects a specific subset of tags with indicator vectors that satisfy $I[j] = 1$. If the reader v_i detects any tags in the field of view, then v_i and v_j are neighbors for sure. Otherwise, no response from tags means no tags hold $I[j] = 1$, which further indicates that v_i and v_j are conflict-free.

Given the use of over-the-air writing and selective reading, the execution time relates to only the number m of readers, regardless of the number n of tags. Since m is much smaller than n in practice, the proposed solution greatly improves the time efficiency compared with tag inventory (in §3) that needs more than n tag collections.

4.2 Challenge in Implementation

Consider the above two phases. The selective reading can be well supported by the *Select* command specified in Gen2 (see §4.3). Over-the-air writing, however, needs the reader to write a group of tags in its vicinity via one instruction. We refer to this one-to-many write operation as *BlastWrite*, which is however out of the scope of Gen2 that specifies the reader has to perform memory access on one tag at a time. This condition makes building indicator vectors roll back to one-to-one transmission again.

4.3 EPCglobal Gen2 Protocol

The EPCglobal Gen2 protocol [4] defines the physical interactions and logical operating procedures between readers and tags. We highlight two functions that we will use on *Retwork* shortly later.

Select Command. *Select* is a mandatory command that can assert or deassert a tag's selected (SL) flag, or set a tag's inventoried flag to either A or B. These flags are used to determine whether a tag may respond to a reader, which is the key to identify the reader network (details are given in §4.4). *Select* comprises six mandatory fields.

• *Target.* It indicates the object that *Select* will operate, which is either a tag's SL flag or an inventoried flag in any one of four sessions. Sessions are specified by Gen2 to fit the case of exclusive reading amongst multiple readers.

• *Action.* This field elicits the action to be taken by a tag. Eight actions are available, where matching and not-matching tags assert or dessert their SL flags, or set their inventoried flags to A or B. By combining *Target* and *Action*, the reader is able to modify a specific flag. For example, a matching tag's inventoried flag in session 2 will be set to A when *Target*$= 010_2$ and *Action*$= 000_2$.

• *MemBank*, *Pointer*, *Length*, *Mask*. The four fields jointly determine which tags are matched for *Action*. *MemBank* specifies the memory bank. *Pointer* indicates the starting position. *Length* determines the length of *Mask*, which is a customized bit string according to upper application requirements. If *Mask* is the same as the string that begins at *Pointer* and ends *length* bits later in the memory of *MemBank*, then the corresponding tag is matched.

Query Command. After *Select*, *Query* initiates and specifies a new inventory round over the tag subpopulation chosen by *Select*. In the inventory round, the reader will play out a frame that consists of a group of time slots. Each selected tag randomly picks one of these time slots and transmits its tag ID to the reader in that slot. A tag inventory may need to execute several inventory rounds and is finished after all selected tags successfully reply to the reader. *Query* command includes three fields that we concern.

• *Sel.* This field consists of two bits that determine which tags respond to *Query*: 00_2 and 01_2 indicate all matching tags in the previous *Select* command; 10_2 indicates tags with deasserted SL flag; 11_2 indicates tags with asserted SL flag.

• *Session.* It selects a session for the subsequent inventory round. Gen2 requires readers and tags to provide four sessions (denoted as S0, S1, S2, and S3). Tags in one of these sessions shall neither use nor modify an inventoried flag for a different session. This way allows two or more readers to use different sessions to independently inventory a common tag population (in different time slots).

• *Target.* This field chooses whether tags with inventoried flag of A or B participate in the upcoming inventory round, where 0 indicates A and 1 indicates B. Tags may invert their inventoried flags from A to B (or vice versa) after being successfully queried.

4.4 Design of Retwork

Although *BlastWrite* is not supported by Gen2, a reader's command (e.g., *Select* and *Query*) can be transmitted to all tags simultaneously through one-to-many broadcast. If we can piggyback some useful information in a reader's command such that all tags' memories are updated in the meanwhile, then an equivalent mimic of *BlastWrite* might be implemented on Gen2. An indicator flag (inventoried flag or SL flag) can make this condition possible because all tags' indicator flags can be set by a single reader command. Below, we detail the use of reader commands together with the Gen2-compatible indicator flag to obtain the reader network. For ease of presentation, we choose the inventoried flag in session 2 (S2) as the vehicle to show how *Retwork* works; other indicator flags can also be adopted similarly.

4.4.1 Detection of Contention Link

Consider any two readers v_i and v_j, $1 \leq i < j \leq m$. The contention link between v_i and v_j can be determined by the following three steps: (i) the reader v_i broadcasts a *Select* command to set all inventoried flags of the tag set in its vicinity to A; (ii) the reader v_j performs the similar operation that sets the inventoried flags to B; (iii) the reader v_i issues a *Query* command and executes the tag inventory on the subset of tags with inventoried flags of B. If v_i and v_j are neighbors, the tags in the overlapped zone must be set to B and v_i can get replies from these tags. Otherwise, none of tags in the field of view of v_i is set to B and nothing will be received. Accordingly, by checking tag replies, v_i can learn whether it conflicts with v_j. Next, we elaborate the way to achieve the above function with Gen2-compatible *Select* and *Query*. A *Select* command is denoted by:

$$\mathcal{S}(\underbrace{t}_{Target}, \overbrace{a}^{Action}, \underbrace{b}_{MemBank}, \overbrace{p}^{Pointer}, \underbrace{l}_{Length}, \overbrace{k}^{Mask}), \qquad (1)$$

with the fields of *Target* (t), *Action* (a), *MemBank* (b), *Pointer* (p), *Length* (l), and *Mask* (k). To set all tags' inventoried flags (in S2) to A, the reader needs to broadcast:

$$Flag = A : \mathcal{S}(2,0,1,0,0,0),$$

where $t = 2$ (010_2) means the operating object is set to the inventoried flag in session 2 (S2), $a = 0$ indicates that the inventoried flags of matching tags will be set to A while those of not-matching will be set to B, and $(b, p, l, k) = (1,0,0,0)$ means all tags within the coverage are selected (matching). Similarly, to set the inventoried flag to B, the reader only needs to carry out the same *Select* except that the value of *Action* field is altered to 100_2 ($a = 4$). Thus, we have:

$$Flag = B : \mathcal{S}(2,4,1,0,0,0).$$

After *Select*, a *Query* is needed for inventory:

$$Q(\underbrace{e}_{Sel}, \overbrace{s}^{Session}, \underbrace{g}_{Target}). \qquad (2)$$

To inventory all tags with inventoried flags in S2 of B, the query command shall be $Q(0,2,1)$, where *Sel* (e), *Session* (s), and *Target* (g) are 00_2, 10_2, and 1, respectively. By combining the *Select* and *Query* together, we obtain the instructions for detecting the contention link between v_i and v_j.

$$\begin{aligned} ① \; v_i &: \; \mathcal{S}(2,0,1,0,0,0) \\ ② \; v_j &: \; \mathcal{S}(2,4,1,0,0,0) \\ ③ \; v_i &: \; Q(0,2,1). \end{aligned} \qquad (3)$$

With these commands, the contentious tags (if appropriate) in the overlapped zone between v_i and v_j are isolated from others; the reader v_i can check the existence or absence of these tags by executing a short inventory round. Compared with the basic tag inventory over the entire tag set, this way avoids a great number of tag memory accesses, regardless of the number of tags.

4.4.2 Identification of Reader Network

The findings above indicate that an intuitive solution to identifying the reader network is to detect each pair of readers with the instructions of (3) and later draw the reader graph with the identified contention relationships. This method is far superior to the inventory-based solution as most tag inventories can be avoided. However, this one-pair-at-a-time scheme suffers from repetitive transmissions for setting inventoried flags, increasing the communication overhead. Take the reader v_1 for example. To get the contention relationships between v_1 and other readers, the reader v_1 needs to execute the operation of setting A (① in (3)) $m-1$ times. If this repetition can be avoided, then a great deal of communication overhead will be saved, which improves the protocol performance. To this end, we propose a scheduling policy that tries to reduce the number of broadcasts of instructions and improve the global time efficiency for obtaining the reader network.

The basic idea is that, instead of solely checking each pair of readers, we identify a group of contention relationships by simultaneously taking multiple readers into account. To identify the entire reader network, $m-1$ identification rounds are needed (m is the number of readers). In each round, we select a reader and identify the contention relationships between this reader and others. More specifically, all readers except for v_1 initially set the inventoried flags to A. Thereafter, the identification starts, round by round. In the i-th round ($1 \leq i < m$), we select the reader v_i and check whether it conflicts with other readers v_j ($j > i$) via three steps. (i) The reader v_i solely updates the flag to B. (ii) Each reader v_j in turn queries the tags with the flag equal to B. If any v_j conflicts with v_i, then the tags in the overlapped zone must be set to B in the first step and v_j can obtain the reply from these tags. Otherwise, no tags in the field of view of v_j reply. By this means, all the contention relationships between v_i and v_j are identified. Compared with the one-pair-at-a-time scheme, the reader v_i executes the flag setting only once rather than $m-1$ times, greatly saving the communication overhead. (iii) In the last step, we reset all flags under v_j's coverage to A again, which would be the input of the next round. For this purpose, one broadcast of $Flag = A$ by v_i is sufficient; no need for all readers v_j to do so. That is because the flags that transform from A to B are due to the flag setting by v_i, that is, the tags with flag B must be under v_i's coverage. After the three steps, the current round terminates and we move to the next one. This process repeats round by round until the global reader network is identified.

4.5 Time Efficiency & Improvement

Now, we discuss the execution time of *Retwork*. As aforementioned, $m-1$ identification rounds need to be run to obtain the reader network. Consider the i-th round, $1 \leq i < m$. It consists of two *Select* commands (sent by v_i) and $m-i$ *Query* commands (issued by each reader v_j, $j > i$). Therefore, the execution time of the i-th round is $2t_s + (m-i)(t_q + t_v)$, where t_s, t_q, and t_v are the time intervals of a *Select* command, a *Query* command, and an inventory round that checks whether requested tags exist, respectively. By considering $m-1$ rounds together with initial $m-1$ flag settings, we have the execution time T of *Retwork*:

$$T = \frac{m(m-1)}{2}(t_q + t_v) + 3(m-1)t_s. \tag{4}$$

This execution time is determined by only the number of readers once the transmission rate of reader-tag communication is fixed, regardless of the large number n of tags. This way is a great performance boost compared with tag inventory given that m is much smaller than n. For example, in a practical scenario (e.g., warehouse or library), a reader usually covers more than 1000 passive tags, i.e., $m \leq 0.001n$. Below, we propose two schemes that further improve the time efficiency of *Retwork*. First, Gen2 allows a tag to send a truncated reply (i.e., a portion of EPC) by enabling *Truncate* field of the *Select* command. This way will help a tag reduce the transmission of the 96-bit EPC to only a single bit, which reduces the communication delay of tag replies. Second, if the range limit of RFID readers is considered, we do not need to check two readers beyond the communication range, which sharply reduces the number of contention detection.

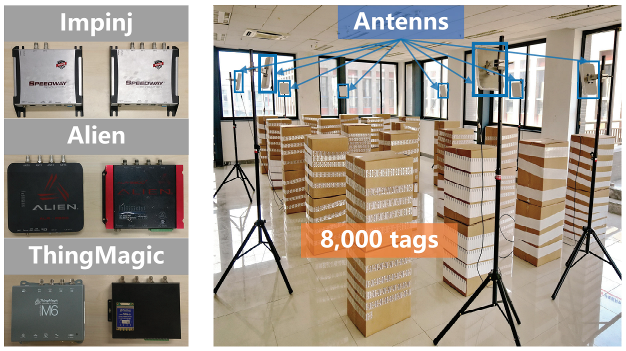

Figure 1: Experimental setup.

5 Evaluation

We evaluate *Retwork* using COTS RFID readers and tags. Six models of UHF RFID readers from three experienced suppliers of RFID products are used in our experiments: ALR-F800 and 9900+ from Alien Inc. [1], R220 and R420 from Impinj [2], Mercury6 and M6e from ThingMagic [3]. Each reader is connected to a directional antenna Larid S9028PCR [11] that is with 8.5 dBic gain and operates at around 900 MHz. To better mimic a real RFID system and extensively study the performance of *Retwork* in practice, we use a total of 8,000 commodity tags in our experiments. As shown in Fig. 1, these tags are densely attached to 40 cartons, each of which contains approximately 200 tags. The readers are deployed with three kinds of densities, namely, sparse (four readers), moderate (six readers), dense (eight readers), to cover a desired area of 12m×8m. The sparse fits for the scenarios of dock door or conveyor belt; the moderate is used for the case of laboratories or office; the dense is suitable for the case of libraries or shopping malls.

5.1 Identification Accuracy

Retwork examines each contention link between two readers by broadcasting *Select* and *Query* commands. Due to the manner of one-to-many transmission, *Retwork* might incur some false positives or false negatives. A false positive is a result that indicates a contention link exists, when it does not indeed. On the contrary, a false negative indicates that two readers do not conflict, while in fact they do. We next study the identification accuracy of *Retwork*, which is measured by false positive ratio (FPR) and false negative ratio (FNR). The ground truth is obtained by executing the inventory-based solution: each reader conducts the tag inventory in sequence and later compares its own tag list with others'; if two readers share a common tag subset, then they conflict. Fig. 2 shows FPR and FNR in three scenarios of different reader densities.

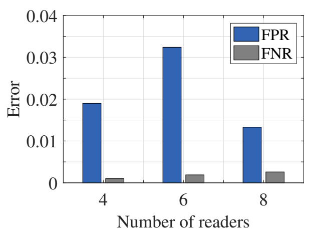

Figure 2: Accuracy.

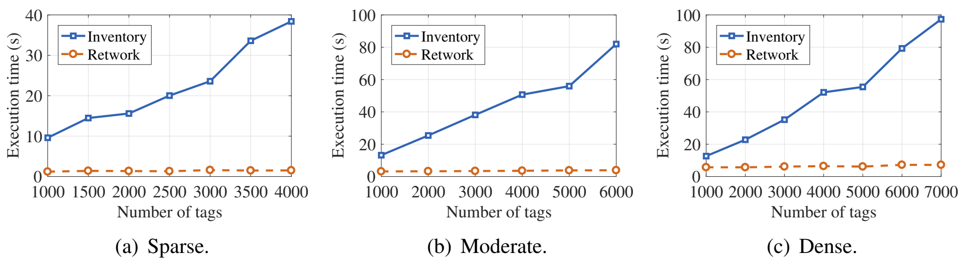

Figure 3: Time comparison between *Retwork* and tag inventory.

As observed, FPR and FNR are bounded within a low level of errors. Although FPR is relatively larger than FNR, false positives slightly negatively affect the functions of multi-reader protocols. Instead, false negatives might incur some errors: two neighbor readers work concurrently. However, the FNR is 0.3%, which is very small for most applications. We can further decrease FNR by running *Retwork* multiple times if desired.

5.2 Time Efficiency

The usage of inventoried flags and reader commands avoids most tag inventories and thus lowers the identification latency of the reader network. This is where *Retwork* shines. Now, we study the time efficiency of *Retwork* over the three reader scenarios, where tag inventory in §3 is taken as the baseline for comparison. Fig. 3 plots the execution time of *Retwork* and tag inventory with respect to the number of unique tags that have been read. As observed, *Retwork* is far superior to tag inventory under difference cases. For example, in the moderate case (six readers) with 5,000 tags (Fig. 3(b)), *Retwork* reduces the execution time from 55.9s to only 3.9s, producing a 14.7× performance gain. Given a reader deployment, the execution time nearly remains stable, regardless of the number of tags. This result is consistent with our previous belief in the protocol design. With the increase in the number of readers, the number of detection units required by *Retwork* increases, as well as the execution time. Notably, the execution time here is the worst case of *Retwork*. If the truncated reply and range limit are taken into account (see §4.5), then the performance of *Retwork* will be further improved. The execution time of tag inventory does not see a linear rise over the number of tags because the number of tags in the overlapped read zones increases correspondingly.

6 Related Work

In a multi-reader RFID system, reader collisions occur frequently and inevitably, which impairs the read throughout and leads to misreads. Research has been trying to find solutions to properly schedule readers to ensure that only those that do not collide will be active at any time. Colorwave [30] is one of the first work to address reader collision. It randomly colors readers such that each pair of interfering readers have different colors. In AcoRAS [8], readers are assigned colors by a central server following the build of a minimum independent set. Season [37] proposes a scheme of reader collaboration to improve the time efficiency of tag inventory by using two steps: shelving the collisions and identifying the tags that do not involve reader collisions; performing a joint identification, in which adjacent readers collaboratively identify the contentious tags. Liu et. al [14] propose a maximum-weight-independent-set-based approximation algorithm to address the problem of reader-coverage collision avoidance: activating readers and adjusting their interrogation ranges to cover maximum tags without collisions subject to the limited number of tags read by a reader. In spite of the advancement, the reader scheduling largely depends on the knowledge of reader network, which is a graph that depicts the contention relationship among the readers and underlies many prior multi-reader protocols. Obtaining the reader network, however, is no picnic in practice.

7 Conclusion

In this work, we investigate the fundamental problem of exploring reader network, which is vital to reader scheduling and underlies many anti-collision protocols in a multi-reader RFID system. A Gen2-compatible protocol *Retwork* is proposed to identify reader network on COTS devices. By exploiting flag-setting capability of commercial tags, *Retwork* avoids most tag inventories and improves time efficiency. Extensive experiments show that *Retwork* can reduce the execution time by an order of magnitude.

Acknowledgments

We would like to thank our shepherd, Prof. Lin Zhong, and the anonymous reviewers for their valuable feedback. This work is supported by National Natural Science Foundation of China (Nos. 61702257 and 61771236), Natural Science Foundation of Jiangsu Province (BK20170648), Jiangsu Key R&D Plan (Industry Foresight and Common Key Technology, BE2017154), US National Science Foundation CNS-1718708, Fundamental Research Funds for the Central Universities (14380066), and Collaborative Innovation Center of Novel Software Technology and Industrialization. Jia Liu and Lijun Chen are the corresponding authors.

References

[1] Alien Technology. http:// www. alientechnology. com.

[2] Impinj Inc. http:// www. impinj. com.

[3] Thingmagic. http:// www. thingmagic. com.

[4] *GS1 EPCglobal. EPC radio-frequency identity protocols generation-2 UHF RFID version 2.0.1*, 2015.

[5] Maurizio A. Bonuccelli and Francesca Martelli. A very fast tags polling protocol for single and multiple readers RFID systems, and its applications. *Ad Hoc Networks*, 71:14–30, 2018.

[6] Binbin Chen, Ziling Zhou, and Haifeng Yu. Understanding RFID counting protocols. In *Proc. of ACM MobiCom*, pages 291–302, 2013.

[7] Isaac Ehrenberg, Christian Floerkemeier, and Sanjay Sarma. Inventory management with an RFID-equipped mobile robot. In *Proc. of IEEE CASE*, pages 1020–1026, 2007.

[8] Essia Hamouda, Nathalie Mitton, and David Simplot-Ryl. Reader anti-collision in dense RFID networks with mobile tags. In *Proc. of IEEE RFID-TA*, pages 327–334, 2011.

[9] Jinsong Han, Chen Qian, Xing Wang, Dan Ma, Jizhong Zhao, Pengfeng Zhang, Wei Xi, and Zhiping Jiang. Twins: Device-free object tracking using passive tags. In *Proc. of IEEE INFOCOM*, pages 469–476, 2014.

[10] Nikolaos Konstantinou. Expowave: An RFID anti-collision algorithm for dense and lively environments. *IEEE Transactions on Communications*, 60(2):352–356, 2011.

[11] Larid. S9028PCL. https:// www. lairdtech. com/ products/ s9028pcl .

[12] Chun-Hee Lee and Chin-Wan Chung. RFID data processing in supply chain management using a path encoding scheme. *IEEE Transactions on Knowledge and Data Engineering*, 23(5):742–758, 2011.

[13] Renjun Li, Zhiyong Huang, Ernest Kurniawan, and Chin Keong Ho. AuRoSS: an autonomous robotic shelf scanning system. In *Proc. of IEEE/RSJ IROS*, pages 6100–6105, 2015.

[14] Bing-Hong Liu, Ngoc-Tu Nguyen, Van-Trung Pham, and Yu-Huan Yeh. A maximum-weight-independent-set-based algorithm for reader-coverage collision avoidance arrangement in RFID networks. *IEEE Sensors Journal*, 16(5):1342–1350, 2016.

[15] Jia Liu, Bin Xiao, Kai Bu, and Lijun Chen. Efficient distributed query processing in large RFID-enabled supply chains. In *Proc. of IEEE INFOCOM*, pages 163–171, 2014.

[16] Jia Liu, Feng Zhu, Yanyan Wang, Xia Wang, Qingfeng Pan, and Lijun Chen. RF-Scanner: Shelf scanning with robot-assisted RFID systems. In *Proc. of IEEE INFOCOM*, pages 1–9, 2017.

[17] Xuan Liu, Bin Xiao, Feng Zhu, and Shigeng Zhang. Let's work together: Fast tag identification by interference elimination for multiple RFID readers. In *Proc. of IEEE ICNP*, pages 1–10, 2016.

[18] Jiaqing Luo and Kang G Shin. Detecting misplaced RFID tags on static shelved items. In *Proc. of ACM MobiSys*, pages 378–390, 2019.

[19] Saiyu Qi, Yuanqing Zheng, Xiaofeng Chen, Jianfeng Ma, and Yong Qi. Double-edged sword: Incentivized verifiable product path query for RFID-enabled supply chain. In *Proc. of IEEE ICDCS*, pages 414–424, 2017.

[20] Saiyu Qi, Yuanqing Zheng, Mo Li, Yunhao Liu, and Jinli Qiu. Scalable data access control in RFID-enabled supply chain. In *Proc. of IEEE ICNP*, pages 71–82, 2014.

[21] Chen Qian, Yunhuai Liu, R.H. Ngan, and L.M. Ni. ASAP: Scalable collision arbitration for large RFID systems. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1277–1288, 2013.

[22] Chen Qian, Hoilun Ngan, Yunhao Liu, and L.M. Ni. Cardinality estimation for large-scale RFID systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(9):1441–1454, 2011.

[23] Aysegul Sarac, Nabil Absi, and Stephane Dauzere-Peres. A literature review on the impact of RFID technologies on supply chain management. *International Journal of Production Economics*, 128(1):77–95, 2010.

[24] Muhammad Shahzad and Alex X. Liu. Every bit counts: Fast and scalable RFID estimation. In *Proc. of ACM MobiCom*, pages 365–376, 2012.

[25] Longfei Shangguan and Kyle Jamieson. The design and implementation of a mobile RFID tag sorting robot. In *Proc. of ACM MobiSys*, pages 31–42, 2016.

[26] Longfei Shangguan, Zheng Yang, Alex X. Liu, Zimu Zhou, and Yunhao Liu. Relative localization of RFID tags using spatial-temporal phase profiling. In *Proc. of USENIX NSDI*, pages 251–263, 2015.

[27] Shaojie Tang, Cheng Wang, Xiangyang Li, and Changjun Jiang. Reader activation scheduling in multi-reader RFID systems: A study of general case. In *Proc. of IEEE IPDPS*, pages 1147–1155, 2011.

[28] ShaoJie Tang, Jing Yuan, Xiang-Yang Li, Guihai Chen, Yunhao Liu, and JiZhong Zhao. RASPberry: A stable reader activation scheduling protocol in multi-reader RFID systems. In *Proc. of IEEE ICNP*, pages 304–313, 2009.

[29] Deepak Vasisht, Guo Zhang, Omid Abari, Hsiao-Ming Lu, Jacob Flanz, and Dina Katabi. In-body backscatter communication and localization. In *Proc. of ACM SIGCOMM*, pages 132–146, 2018.

[30] James Waldrop, Daniel W. Engels, and Sanjay E. Sarma. Colorwave: a MAC for RFID reader networks. In *Proc. of IEEE WCNC*, volume 3, pages 1701–1704, 2003.

[31] Chuyu Wang, Jian Liu, Yingying Chen, Lei Xie, Hong Bo Liu, and Sanclu Lu. RF-Kinect: A wearable RFID-based approach towards 3D body movement tracking. *Proc. of ACM UbiComp*, 2(1), 2018.

[32] Ju Wang, Jie Xiong, Hongbo Jiang, Xiaojiang Chen, and Dingyi Fang. D-watch: Embracing bad multipaths for device-free localization with COTS RFID devices. In *Proc. of ACM CoNEXT*, pages 253–266, 2016.

[33] Lei Xie, Jianqiang Sun, Qingliang Cai, Chuyu Wang, Jie Wu, and Sanglu Lu. Tell me what I see: Recognize RFID tagged objects in augmented reality systems. In *Proc. of ACM UbiComp*, pages 916–927, 2016.

[34] Huatao Xu, Dong Wang, Run Zhao, and Qian Zhang. AdaRF: Adaptive RFID-based indoor localization using deep learning enhanced holography. *Proc. of ACM UbiComp*, 3(3):1–22, 2019.

[35] Peizhi Yan, Salimur Choudhury, and Ruizhong Wei. A distributed graph-based dense RFID readers arrangement algorithm. In *Proc. of IEEE ICC*, pages 1–6, 2019.

[36] Lei Yang, Yekui Chen, Xiang-Yang Li, Chaowei Xiao, Mo Li, and Yunhao Liu. Tagoram: Real-time tracking of mobile RFID tags to high precision using COTS devices. In *Proc. of ACM MobiCom*, pages 237–248, 2014.

[37] Lei Yang, Yong Qi, Jinsong Han, Cheng Wang, and Yunhao Liu. Shelving interference and joint identification in large-scale RFID systems. *IEEE Transactions on Parallel and Distributed Systems*, 26(11):3149–3159, 2015.

[38] Jihong Yu, Wei Gong, Jiangchuan Liu, and Lin Chen. Fast and reliable tag search in large-scale RFID systems: A probabilistic tree-based approach. In *Proc. of IEEE INFOCOM*, pages 1133–1141, 2018.

[39] Jihong Yu, Wei Gong, Jiangchuan Liu, Lin Chen, Fangxin Wang, and Haitian Pang. Practical key tag monitoring in RFID systems. In *Proc. of IEEE/ACM IWQoS*, pages 1–10, 2018.

[40] Shigeng Zhang, Xuan Liu, Jianxin Wang, and Jiannong Cao. Tag size profiling in multiple reader RFID systems. In *Proc. of IEEE INFOCOM*, pages 1–9, 2017.

[41] Yuanqing Zheng and Mo Li. ZOE: Fast cardinality estimation for large-scale RFID systems. In *Proc. of IEEE INFOCOM*, pages 908–916, 2013.

[42] Zongheng Zhou, Himanshu Gupta, Samir R. Das, and Xianjin Zhu. Slotted scheduled tag access in multi-reader rfid systems. In *Proc. of IEEE ICNP*, pages 61–70, 2007.