RASPberry: A Stable Reader Activation Scheduling Protocol in Multi-Reader RFID Systems

ShaoJie Tang*, Jing Yuan†, Xiang-Yang Li*,§, Guihai Chen†, Yunhao Liu‡, and JiZhong Zhao§
*Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA
†State Key Lab of Novel Software Technology, Nanjing University, Nanjing, China
‡Department of Computer Science and Engineering, HKUST, Hongkong, China
§Department of Computer Science, Xi'An JiaoTong University, Xi'An, China

***Abstract*—Recent technological advances have motivated large-scale deployment of RFID systems. RFID readers are often static and carefully deployed in a planned manner. However, the distribution and movements of tags are often dynamically changed and unpredictable. We study a challenging problem of scheduling the activation of the readers without collision such that the system can work in a stable way in the long term. Here a schedule is stable if at any time slot, the number of total unread tags is bounded from above with high probability under this scheduling. In this paper, we propose a stable reader activation scheduling protocol, *RASPberry*, in multi-reader RFID systems. We analytically prove that our scheduling protocol, RASPberry, is stable if the arrival rate of tags is less than the processing rate of all readers. In RASPberry, at any time slot, a reader can determine its status using only information of readers within a local neighborhood. To the best of our knowledge, this is the first work to address the stability problem of reader activation scheduling in RFID systems. Our extensive simulations show that our system performs very well.**

***Index Terms*—RFID, reader, scheduling, stability, graph.**

I. Introduction

In order to develop efficient and effective RFID systems, we need to deal with many challenging issues such as *anonymous*, *security* and *read throughput*. In this work, we focus on the read throughput, the number of tags read per time slot. RFID readers often have a limited *interrogation region* within which it can communicate with a tag. This interrogation region depends on many factors such as the antenna, barriers between the reader and tags, and the characteristics of tags.

In many locations essential to daily life such as supermarket or post office, multiple RFID readers are needed in a given region to ensure a certain level of coverage. We study a large-scale RFID system for estimating the number of tourists in a large park. In the system, we need multiple static readers to perform tag reading concurrently, which will improve the read throughput greatly. RFID readers are often static and carefully deployed in a planned fashion. When the tags are static and given a priori, Zhou *et al.* [4] proposed a novel RFID reader scheduling protocol to improve the read throughput of multiple readers RFID system. However, the distribution and movements of tags are often dynamically changed, instead of static and known a priori. For example, within one day, new tourists will arrive at different times and at different locations, existing tourists will tour the park and leave the park finally. Under this highly dynamic environment, we cannot expect the number of unread tags within the interrogation range of any reader to be fixed, but to follow some arrival distribution.

One challenging problem with multi reader RFID systems is to schedule the activation of readers without collision such that the system can work stably for a long term. We need to schedule the readers in such a way that at any time slot, the total number of unread tags at any timeslot is bounded from above by some constant. Note that by considering the limitation of the system's ability (or capacity), the system may not always work in a stable way no matter what kind of scheduling scheme is implemented, *e.g.*, when a huge number of tags arrive at the system suddenly. Thus, instead of guaranteeing the stability always, the objective of this work is to design a scheduling scheme under which the system can perform "best" in terms of stability within its ability (or capacity).

Specifically, we study the problem of slotted scheduled activation of RFID readers in a *dynamic* environment. We assume that at every time slot, some new tags will appear in the interrogation ranges of readers and some tags will be accessed by these readers and thus remain silent henceforth. Tags that will be accessed periodically can be treated in a similar manner. We develop low complexity centralized and distributed scheduling algorithms for the activation of readers, by assuming the knowledge of the readers' positions, their interrogation regions, and the arrival rate of new tags. In our distributed scheduling protocol, each reader only needs the information of readers within a distance $O(r)$ to finalize its activity for every time-slot, where r is its interrogation range. We prove that our scheduling schemes can achieve near optimal capacity, *i.e.*, the total number of tags that can be read without leaving many tags unread in some region. Our protocols are proven to be *stable* if the tag arrival rates are in the capacity region of the readers. To the best of our knowledge, this is the first work to address the stability problem of reader activation scheduling in RFID systems.

The rest of the paper is organized as follows: We present our system model in Section II, then formally define the problem in Section III and present an overview of our protocol, RASPberry. We present our centralized RFID activation scheduling protocol in Section IV and distributed read activation protocol in Section V. The performance studies of our protocols are reported in Section VI. We review the related work in Section VII and conclude the paper in Section VIII.

978-1-4244-4634-6/09/$25.00 ©2009 IEEE

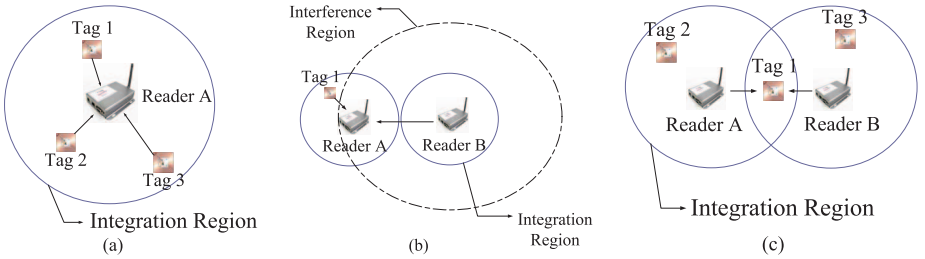


Fig. 1. Collisions in RFID systems. (a) Tag-Tag collision: Tags 1, 2 and 3 respond to reader A simultaneously and causing collision at A; (b) Reader-Tag collision: Response from tag 1 to Reader A is "drowned" by the signal from the reader B; (c) Reader-Reader collision: Signal from reader A and B collide at tag 1

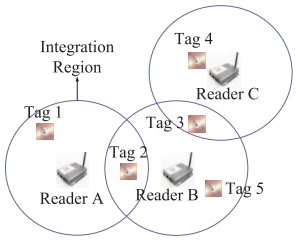


Fig. 2. Illustration of an example in which scheduling less readers will read more tags.

TABLE I
NOTATIONS USED IN PAPER

Symbol	Meaning
I_t	a set of readers for scheduling at time t
$\mathbf{s}_j$	a region partitioned by the interrogation regions of all readers
$X_t(i)$	the tags arrived for reader i at time t
$X_t(\mathbf{s}_j)$	the expected number of tags in the subregion $\mathbf{s}_j$ at time t
$\omega(I,t)$	the weight of a scheduling I at time t (# of tags read)
$I_t^{(i,j)}$	the optimum scheduling for **square**(i,j) under X_t.
$\mathcal{C}$	the capacity of the RFID system (it is a vector)

II. BACKGROUND ON RFID SYSTEM

In this section, we briefly review some basic definitions and models regarding RFID systems.

Interrogation and Interference Regions: Each RFID reader has an *interrogation region* and an *interference region*. The *interrogation region* of a reader is the region within which the tags can be successfully (without causing any collisions) read by that reader. The *interference region* refers to the region around a reader where the tag response will be affected by the reader. For simplicity of description, we assume both the interrogation region and interference region are disks with radius r and R respectively. Typically, the values of r vary from ten centimeters to hundred feets. Without loss of generality, we further assume $r = \beta R$ where $1 > \beta > 0$ is a constant. Our results still hold if the interrogation and interference regions satisfy that there is a value a and for any region $\mathbf{X}$ with area A, the number of readers that can be active simultaneously in $\mathbf{X}$ is at most $O(A/a^2)$. [4]Given a set of readers $\mathbf{R}$, we define the *region monitored by the readers* $\Re$ as the union of the interrogation regions of $\Re$. These regions can be explored by a RF site survey. Such surveys can be achieved by using certain localization device together with radio signal strength measurement device. Remember that all the readers are deployed in a planned manner, we may expect those surveys to be practical.

Collisions in Multi-Reader Systems: Due to the interference effect, simultaneous transmissions in RFID systems may lead to collisions. Typically, there are three types of collisions.

Tag-tag collision(TTc): When multiple tags located in the interrogation region of the same reader transmit data at the same time, it will lead to tag-tag collision. See Figure 1(a). To design a feasible scheduling that avoids the tag-tag collision, we may schedule the activities of tags through certain link-layered protocol such as framed Aloha [15] or tree-splitting [11], [13]. Here we assume this is already in place.

Reader-tag collision(RTc): This happens when a reader is in the interference region of another one. For example, in Figure 1(b), transmission from reader B can affect the response from $\mathtt{Tag}_1$ to A. We use RTc to denote this kind of collision. In this case, any reader suffering RTc can not read any tag. Thus, we need to carefully schedule the activation of different readers.

Reader-reader collision(RRc): This happens when some tags are located within the overlapping interrogation regions of two different active readers. In such a case, the tags can not tell the difference between the signals from those two readers. We denote this collision as RRc. Note that it is different from RTc, even though RRc will make the tags within the overlapping interrogation region of two active readers fail to access, the readers can still read other tags that are inside only its own interrogation region. For example, in Figure 1 (c), both $\mathtt{Tag}_2$ and $\mathtt{Tag}_3$ except $\mathtt{Tag}_1$ still have a chance to be served successfully.

Similar as the work done by Zhou *et al.* [4], by using STDMA style scheduling, we mainly study the reader activation scheduling problem in a multiple-reader environment. Specially, we try to tackle Reader-tag and Reader-reader collision problems in this work. The basic idea of this paper is to utilize synchronized time slots among the readers, and select a set of appropriate readers to active in each time slots such that the system can work a a stable way. As we mentioned before, since the TTc can be avoided through certain appropriate link layer protocol (such as framed-Aloha based [15] or a tree-splitting protocol [13]). Thus, we will not make any change to the link layer protocol which is already in place. Our RASPberry protocol is for reader activation.

III. PROBLEM FORMULATION AND PROTOCOL OVERVIEW

In this section we describe our scheduling model in detail and give an overview of our RASPberry protocol.

A. Problem Formulation

Consider a set of n readers $\Re = \{r_1, r_2, \cdots, r_n\}$ deployed statically at a deployment region. Each reader r_i has its own exogenous *i.i.d.* arrival stream of tags $\{A_t(i)\}$ where $A_t(i)$ is

the set of *new* tags appeared in its interrogation region at time slot t. After tag $\mathtt{Tag}_i$ accessing some reader, we say that it leaves the system, that is, there is no further request for $\mathtt{Tag}_i$ as long as it has been served. In the rest of our paper, we use $|S|$ to denote the size of a set S.

For each time slot t, the set of activated readers I_t contains all the readers that should be activated at slot t. Note that the time required to read tags is proportional to the number of tags to be read without any collision [13], [15]. Assume that each reader r_i can read α tags within its intermigration region successfully in one time-slot, where α is some constant depending on certain link layer protocol. There may be conflicts in the simultaneous activation of certain combination of readers. Here conflicts refer to either RTc or RRc in this paper. A *valid scheduling set* of readers activated at a certain slot should not contain any combination which will lead to RTc. However, the active readers with only RRc can still be considered as a valid scheduling set. The number of readable tags may be smaller than the sum of tags readable by all readers due to RRc. Specially, a set of readers that are eligible to be activated simultaneously must avoid RTc.

For a time-slot t, we denote the set of active readers as $I_t = \{r_i : r_i \text{ is activated at time slot } t\}$. Let $\mathcal{I}$ be the collection of all possible valid scheduling sets. At each time slot t, an active reader set $I_t \in \mathcal{I}$ is scheduled. A scheduling policy $\{I_t : t = 1, 2, \cdots, \infty\}$ dictates which activation reader set is used at each slot t. We define $X_t(i)$ as the set of unread tags that are in the interrogation region of reader r_i at time slot t. The tag distribution vector is $X_t = \{X_t(i) : i = 1, \cdots, n\}$ and the corresponding vector of amount of unread tags is $|X_t| = \{|X_t(i)| : i = 1, \cdots, n\}$. Note that the tags that may have a chance to access readers are those that are within the interrogation area of exactly one activated reader. We formally define the weight of scheduling I_t at time slot t as the following:

Definition 1: Given a valid scheduling I_t, the weight $\omega(I_t, t)$ of I_t at time slot t is defined as the total number of tags that may have a chance to access, without causing RRc, during time slot t, *i.e.*, tags that are positioned in the interrogation region of *one and only one* activated reader.

See Figure 2 for an illustration. Assume that three readers A, B and C are independent from each other that is, none of them is located at the interference regions of other readers. If we choose all of them as an independent set for scheduling, *e.g.*, $I_t = \{A, B, C\}$, then its total weight is $|\{\mathtt{Tag}_1, \mathtt{Tag}_2, \mathtt{Tag}_3, \mathtt{Tag}_4, \mathtt{Tag}_5\}| - |\{\mathtt{Tag}_2, \mathtt{Tag}_3\}| = 3$ since $\mathtt{Tag}_2$ and $\mathtt{Tag}_3$ are inside the overlapping region of two readers. Interestingly, if we only choose A and C as activated readers, *e.g.*, $I_t = \{A, C\}$, the weight is $|\{\mathtt{Tag}_1, \mathtt{Tag}_2, \mathtt{Tag}_3, \mathtt{Tag}_4\}| - |\{\emptyset\}| = 4$ which is larger.

So far we assume that we know the tags that are covered by any single reader r_i, and the tags that are covered simultaneously by any two readers r_i and r_j. Thus, we can compute the weight $\omega(I, t)$ of any possible scheduling I. In practice, we can hardly know such information exactly without any estimation scheme. Therefore, we may estimate the number of tags covered by each reader based upon the distribution of the arrival rate, and we can also implement existing estimation scheme [18] to get the knowledge. Here we show how to estimate it when the tags arrive with a Poisson process with rate λ, *i.e.*, the expected number of new tags that arrive in a unit area during every time-slot is λ. Assume that the boundaries of any two interrogation regions of two readers have at most two intersection points. Let $\mathbf{s}_1$, $\mathbf{s}_2$, $\cdots$, $\mathbf{s}_m$ be the m subregions partitioned by the interrogation regions of all n readers $\Re$. Clearly $m \leq n^2$. Assume that we know the locations of all readers $\Re$ and the exact interrogation region of each reader r_i. We clearly can compute the area, denoted as $\mathrm{Area}(\mathbf{s}_j)$, of each subregion $\mathbf{s}_j$, $j \leq m$. Assume that, given a scheduling I_t of readers, among all tags that can be successfully read by some reader r_i, reader r_i will read at most α of them randomly in a time-slot. Let $X_t(\mathbf{s}_j)$ be the expected number of tags in the subregion $\mathbf{s}_j$ at time t. Given a schedule I, let $x(I, \mathbf{s}_j) \in \{0, 1\}$ denote whether subregion $\mathbf{s}_j$ is uniquely covered by only a single reader in I. If $x(I, \mathbf{s}_j) = 1$, let $r_i(I, \mathbf{s}_j)$ be that unique reader whose interrogation region $\mathbf{R}_i$ contains $\mathbf{s}_j$, and let $\beta(I, \mathbf{s}_j) = \frac{\mathrm{Area}(\mathbf{s}_j)}{\sum_{\mathbf{s}_a \in \mathbf{R}_i} (x(I, \mathbf{s}_a) \cdot \mathrm{Area}(\mathbf{s}_a))}$, *i.e.*, the proportion of the area of subregion $\mathbf{s}_i$ in all subregions readable by reader r_i under I. Let $X_{t-1}(\mathbf{s}_i)$ be the expected number of unserved tags after time $t-1$, before time t. The weight of a schedule I is then the *expected* number of tags that can be read successfully by I. Then, at time t, given a potential valid schedule I, define a variable $Z_t(I, \mathbf{s}_i)$ as $Z_t(I, \mathbf{s}_i) = X_{t-1}(\mathbf{s}_i) + \lambda \cdot \mathrm{Area}(\mathbf{s}_i) - x(I_{t-1}, \mathbf{s}_i) \cdot \alpha \cdot \beta(I_{t-1}, \mathbf{s}_i)$. Given a valid scheduling I at time t, its weight $\omega(I, t)$ is

$$\omega(I, t) = \sum_{\mathbf{s}_j} (x(I, \mathbf{s}_j) \cdot Z_t(I, \mathbf{s}_j)). \tag{1}$$

As analyzed below, we will always try to find a scheduling I with maximum weight $\omega(I, t)$ at each time slot t. After we pick I_t, let $S_t(\mathbf{s}_j)$ be the actual number of tags served in the subregion $\mathbf{s}_j$. Then the expected number of unread tags $X_t(\mathbf{s}_i)$ evolves as $X_t(\mathbf{s}_i) = X_{t-1}(\mathbf{s}_i) + \lambda \cdot \mathrm{Area}(\mathbf{s}_i) - S_t(\mathbf{s}_i)$.

B. Read throughput with Random Tag Arrival

Now we are ready to introduce a key definition which will be used throughout this paper.

Definition 2: The RFID system is considered **stable** if the amount of unread tags process $\{|X_t|\}_{t=1}^{\infty}$ approaches a steady state where the expected amount is bounded. In other words, the scheduling is **stable** if $\sup\{E[|X_t|]\} < \infty$.

Here we do not mean that these finite number of unread tags will never be read. We only mean that at a given time-slot instance, there is a finite number of tags unread. These unread tags will be read in next finite number of timeslots. We only ensure that, with high probability, at any given time-slot, the number of tags that arrived but were not read is finite.

Denoting $E[|A_t(i)|]$ by a_i, the arrival rate vector is $a = \{a_j : j = 1 \cdots, n\}$. A throughput vector a is achievable if there is some scheduling policy under which the system is stable when the arrival rate vector is a.

The *capacity region* $\mathcal{C}$ (a point in the capacity region of the given RFID system is a vector a of the mean arrival rate $E[A_t]$) is the strict convex closure of all valid scheduling $\mathcal{I}$: $a \in \mathcal{C}$ if and only if there exists a scheduling such that the system is stabilized. A scheduling policy is *throughput-optimal* if it can achieve the optimal capacity region $\mathcal{C}$.

Definition 3: The *efficiency ratio* of a scheduling policy is the largest number γ such that the scheduling policy can stabilize the system under any load vector $a \in \gamma \cdot \mathcal{C}$.

A protocol is efficient if its efficiency-ratio is close to 1. We denote the set of all valid schedulings at time slot t as $\mathcal{I}_t$. If a load factor a belongs to $\mathcal{C}$, then the system is stable under the policy that selects for activation at time t the set I_t [3]

$$I_t = \arg \max_{I_t^i \in \mathcal{I}_t} \omega(I_t^i, t) \tag{2}$$

It implies that, the *optimal* scheduling policy is equivalent of computing a maximum weighted independent set (MWIS) of readers in the interference graph. This problem is usually of high computational complexity and even NP-Hard. Hence the maximum read throughput scheduling based on Eq. 2 is not amenable to implementation.

Thus, we will rely on approximation algorithms. A scheduling policy I is called *imperfect scheduling policy with ratio* $0 < \gamma \leq 1$(see [1] for more discussions) if at *each* time-slot t, it will compute a schedule $I_t \in \mathcal{I}_t$ such that

$$\omega(I_t, t) \geq \gamma \cdot \max_{I_t^i \in \mathcal{I}_t} \omega(I_t^i, t)$$

As analyzed later, if we only need to compute an imperfect scheduling for each time slot t, the time complexity is much lower than calculating the optimal one. However, we are more interested in whether the system will remain stable under an imperfect scheduling policy. The following two propositions[1] offer us positive answers, and they will be the foundation for studying the stability of our scheduling protocols.

Proposition 1: [1] Fix $\gamma \in (0, 1]$. if the arriving rates a lie strictly inside $\gamma \cdot \mathcal{C}$ (*e.g.*, a lies in the interior of $\gamma \cdot \mathcal{C}$), then any imperfect scheduling policy I_γ can stabilize the system.

Observe that the scheduling policy I_γ must be able to find the γ-approximation *surely* at *every* time slot t. This requirement may be too strong to be satisfied. Surprisingly, as long as we can guarantee that our scheduling protocol can find a γ-approximation scheduling with a certain constant positive probability, the following proposition was proved.

Proposition 2: [2] Given any $\gamma \in (0, 1]$, suppose that a randomized algorithm has a probability at least $\delta > 0$ of generating a valid set of readers with weight at least γ times the weight of the optimal. Then, capacity $\gamma \cdot \mathcal{C}$ can be achieved by switching activated readers to the new set when its weight is larger than the previous one (otherwise, previous set of activated readers will be kept for scheduling). The algorithm should generate the new scheduling I_t from the old scheduling I_{t-1} and current distribution X_t of unread tags.

In light of Proposition 1 and Proposition 2, it is concluded that if our scheme can generate a near optimal scheduling in terms of weight with *at least a certain constant probability* at each time slot, we can still ensure the stability by *switching activated readers to a new set when its weight is larger than the previous one*. Thus, it is not necessary to always find the optimal scheduling of every time-slot for stability.

C. Overview of Stable Reader Activation Scheduling Protocols

In the following, we will focus on designing our **R**eader **A**ctivation **S**cheduling **P**rotocols, called *RASPberry*, with low complexity in both centralized and distributed manner. We further prove that these novel protocols can ensure certain level of stability. Since a valid scheduling scheme should avoid the Reader-Tag collision(RTc) as well as Reader-Reader collision(RRc), we next give a brief illustration to show how we deal with these two possible collisions in our protocol design. We first define the interference graph of a given RFID reader network as the following:

Definition 4: **Interference graph** of the readers $\Re$ is a graph where every reader in $\Re$ has a corresponding vertex, and any two vertices have an edge between them if and only if one reader is located in the interference region of the other. In other words, any two adjacent readers in interference graph cannot be active simultaneously due to RTc.

Since we assume the interference range of each reader is R, we will connect any two readers in the interference graph if the distance between them is at most R.

Avoid Reader-tag Collision: If two readers connected with an edge in the interference graph are active simultaneously, it will lead to RTc. All the readers from any independent set can be activated simultaneously without causing RTc, *i.e.*, a valid scheduling.

Avoid Reader-reader Collision: If readers with overlapping interrogation regions are activated simultaneously, RRc collision happens. Thus, to avoid possible RRc, we exclude the tags located at the overlapping interrogation regions from the weight $\omega(I_t, t)$ as shown in Definition 1.

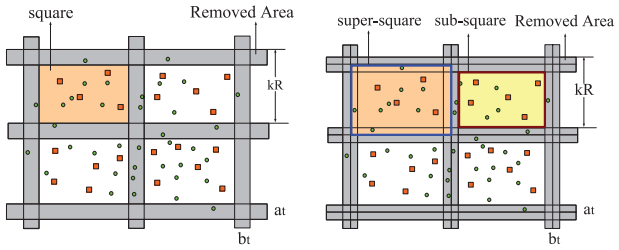
IV. Centralized Reader Activation Scheduling

Assume that the spatial distribution of the tags are known based on a certain estimation scheme. For each time t, we find an active reader set I_t with the maximum or near maximum weight.

A. Efficient Deterministic Scheduling Protocol

Recall that we already know the geometric positions of all readers. We first partition the 2D deployment space into grids using horizontal lines $x = i$ and vertical lines $y = j$ for all integers i and j. The distance between any two neighbor horizontal(or vertical) lines is $\max\{2r, R\}$. For simplicity, by assuming $2r \leq R$, we have $\max\{2r, R\} = R$. A vertical strip *with index* i is $\{(x, y) \mid i < x \leq i + 1\}$. Similarly, we can define a horizontal strip with index j and **cell**(i, j)

[1]Note that Proposition 1 and Proposition 2 are originally proved for the link scheduling problem. In order to apply them in our reader activation scheduling problem, we need slight modification to the original proof. The details are omitted here to save space.



(a) centralized method (b) distributed method

Fig. 3. Divide the space into grids for time-slot t. Here the square nodes are the solution computed at time slot t. Readers fall in the shaded area are removed. (a) space partition method for centralized method. (b) space partition method for distributed method.

Algorithm 1 Centralized Deterministic Scheduling

Input: Location of nodes, queue size of reader r_i, and k.
Output: Feasible active reader set I_t for time slot t.

```
tmp = 0;
for a_t = 0 to k − 1 do
    for b_t = 0 to k − 1 do
        Partition(k, a_t, b_t);
        Compute a MWIS I_t^{(i,j)};
        if ω(⋃_{i,j} I_t^{(i,j)}, t) > ω(tmp, t) then
            tmp = ⋃_{i,j} I_t^{(i,j)};
I_t = tmp;
```

as the intersection area of a vertical strip i and a horizontal strip j. Note that each cell has side length R. See Figure. 3(a) for illustration, where the shaded areas are strips. It is not difficult to find that for any two readers separated by a strip, they can be scheduled for transmitting simultaneously without RTc and RRc. To divide the problem of finding a maximum weighted feasible scheduling into subproblems that are solvable in polynomial time, we divide readers into groups based on grid partition. To ensure that the union of solutions of subproblems are still independent, as a standard approach, we will add a separation between adjacent subproblems as follows: At any time slot t, we will "remove" the readers (*i.e.*, not activate these readers) located inside either vertical strips i with $i = a_t \mod k$ or horizontal strips j with $j = b_t \mod k$. Here $a_t, b_t \in [0, k-1]$ are adjustable numbers. As illustrated in Figure 3(a), we remove all readers inside the gray strips. We define the preceding operations as Partition(k, a_t, b_t), *i.e.*, divide the space into grid-cells and remove some readers.

Given (a_t, b_t), define *square*(i, j) to be the set of cells $\{cell(x, y) \mid x \in [ik + a_t + 1, (i+1)k + a_t - 1], y \in [jk + b_t + 1, (j+1)k + b_t - 1]\}$. A subproblem is then, given a *square*(i, j), to find a MWIS of all readers inside this square. Here each square has size $R(k-1)$ and two readers who are closer than R cannot be activated simultaneously. Thus, the size of *any* set of interference-free readers for a *square*(i, j) is at most $\Lambda = (R(k-1))^2 / \frac{\pi R^2}{4} < 4k^2/\pi$. Thus, a MWIS for each *square*(i, j) can be found by a simple enumeration in time $n_{i,j}^{\Lambda}$, where $n_{i,j}$ is the number of readers inside *square*(i, j).

We compute a scheduling as follows: At time slot t, we choose a partition (corresponding to some specific (a_t, b_t)) and compute a MWIS of readers for each *square*(i, j) that is not empty of readers inside. Let $I_t^{(i,j)}$ be the optimum scheduling solution for *square*(i, j) under current tag distribution X_t. Obviously, there are k^2 different partitions since there are k^2 different choices for (a_t, b_t) and each of them corresponds to a distinct partition. Accordingly, we can choose the "best" partition among the k^2 partitions. Here the best partition refers to the partition such that the total weight of all MWIS for the squares is maximum among all k^2 different partitions. Let I_t be the union of the optimum solutions for all squares in the best partition. Pseudo-codes are listed in Algorithm 1.

Lemma 1: I_t generated by Algorithm 1 is valid.

Proof: According to our algorithm, any two readers r_i and r_j from two different squares are separated by distance at least the strip width R. Thus, they are always interference-free regarding to RTc and RRc since $\max\{2r, R\} = R$. ■

Lemma 2: At each time slot t, the weight of scheduling I_t computed by Algorithm 1 is at least

$$\omega(I_t, t) \geq (1 - \frac{1}{k})^2 \cdot \omega(I_t^{OPT}, t)$$

Theorem 1: Algorithm 1 achieves $(1 - \frac{1}{k})^2 \mathcal{C}$ capacity.

Proof: As stated in Proposition 1, for a fixed $\gamma \in (0, 1]$, if the arrival rates a lies strictly inside $\gamma \cdot \mathcal{C}$ (*e.g.*, a lies in the interior of $\gamma \cdot \mathcal{C}$), then any imperfect scheduling policy $\gamma \cdot I_{OPT}$ can stabilize the system. Remember that a scheduling policy I is called *imperfect scheduling policy with ratio* $0 < \gamma \leq 1$ if at *each* time-slot t, it will compute a schedule $I_t \in \mathcal{I}$ such that $\omega(I_t, t) \geq \gamma \max_{I_t \in \mathcal{I}(t)} \omega(I_t, t) = \gamma \cdot \omega(I_t^{OPT}, t)$. Based on Lemma 2, we know that $\omega(I_t, t) \geq (1 - \frac{1}{k})^2 \omega(I_t^{OPT}, t)$. Then we can finish the proof by setting $\gamma = (1 - \frac{1}{k})^2$. ■

Theorem 2: Algorithm 1 runs in time $O(k^2 \cdot n^{\Lambda})$ where n is total number of readers and Λ is some constant.

B. Efficient Randomized Scheduling Protocol

In Section IV-A, we designed an algorithm based on shifting strategy. In order to find the best partition (k, a_t, b_t) at each time slot t, we need to check every possible partition, which will lead to high time complexity and energy consumption. Here we propose a novel efficient randomized scheduling algorithm. The surprising result is that without decreasing the level of stability, we can remove the factor k^2 from the time complexity of Algorithm 1.

Recall that using space partition, I_t (similar to Algorithm 1) is composed of optimum solutions from each *square*(i, j). If we keep the *same* space partition (same (a_t, b_t) for all t) for all time-slots, clearly, we can produce the solution $I_t^{(i,j)}$ for each *square*(i, j) and $\omega(I_t^{(i,j)}, t) \geq \omega(I_{t-1}^{(i,j)}, t)$ from the optimality of $I_t^{(i,j)}$ for *square*(i, j) with weight X_t. Consequently, $\omega(I_t, t) \geq \omega(I_{t-1}, t)$. However, using the same partition for all time-slots clearly violates the property that I_t has constant approximation ratio with constant probability when $t \to \infty$. The key observation is that, after *fixing* a partition, the removed

Algorithm 2 Centralized Randomized Scheduling

Input: Location of nodes, current distribution of tags.
Output: Feasible active reader set I_t for time slot t.

1: $tmp = I_{t-1}$;
2: Randomly choose a_t and b_t from $[0, k-1]$.
3: Compute Partition(k, a_t, b_t);
4: Compute a MWIS $I'^{(i,j)}_t$;
5: **if** $\omega(\bigcup_{i,j} I'^{(i,j)}_t, t) > \omega(I_{t-1}, t)$ **then**
6: $\quad temp = \bigcup_{i,j} I'^{(i,j)}_t$;
7: $I_t = temp$;

readers (that fall inside the removed strips) will accumulate unread tags in its interrogation region since they will *never* be served now. Thus, to ensure the constant probability of getting good solution, we need *randomly* choose a partition for every time-slot. The challenge now is to ensure that I_t is consistently no worse than I_{t-1} at time slot t.

Like Algorithm 1, Algorithm 2 is also based on shifting strategy. We first partition the 2D deployment region into horizontal and vertical strips with width R. The main difference comes from the next step: Instead of enumerating every possible partition (k, a_t, b_t) to find the best one, we randomly choose one partition for time slot t and compute a candidate scheduling I'_t under this partition. By comparing the weight of the scheduling under current randomly chosen partition and the one under the partition used in previous time slot $t-1$, we choose the better one as I_t: if $\omega(I'_t, t) \geq \omega(I_{t-1}, t)$, we will choose I'_t as I_t; otherwise, we still use I_{t-1} as the scheduling for time slot t.

Lemma 3: Algorithm 2 has a probability of at least $1/k^2$ to generate a set of independent readers with weight at least $(1-\frac{1}{k})^2$ of the optimal one at each time slot t.

In light of Proposition 2, we give the following theorem to evaluate the performance of Algorithm 2. The surprising result is that with a significant improvement in terms of time complexity, Algorithm 2 can still achieve the same level of stability as Algorithm 1 does.

Theorem 3: Algorithm 2 achieves $(1-\frac{1}{k})^2\mathcal{C}$ capacity.

Proof: According to Lemma 3, we know that with probability at least $1/k^2$, the weight of scheduling I_t computed by Algorithm 2 is at least $\omega(I_t, t) \geq (1-\frac{1}{k})^2\omega(I_t^{OPT}, t)$ at each time slot t. Further, because in Algorithm 2, we always choose the better one between scheduling under current random partition and the one under previous partition as I_t. Theorem follow from Proposition 2 by setting $\delta = 1/k^2$. ■

Theorem 4: Algorithm 2 has time complexity $O(n^{\Lambda})$.

C. Discussions

The above described RASPberry protocols for reader activation can be easily generalized to three dimensions. Essentially, in 3D, we will divide the space into cubic cells and randomly pick a partition (a_t, b_t, c_t) for time-slot t.

Lemma 4: In 3D, when $k \geq 3$, Algorithm 2 has a probability of at least $\frac{1}{k^3}$ to generate an independent set of readers with weight at least $(1-\frac{1}{k})^3$ of optimal solution, *i.e.*, $\Pr\left(\omega(I_t, t) \geq (1-\frac{1}{k})^3\omega(I_t^{OPT}, t)\right) \geq \frac{1}{k^3}$. Algorithm 2 runs in time $O(n^{6k^3/\pi})$ in 3D.

V. Distributed Reader Activation Scheduling

As we have mentioned before, in order to implement the above centralized algorithm, it is necessary for us to estimate the global spatial distribution of tags efficiently and accurately. However, it is often very hard to do so in practice. Thus, we design a randomized distributed reader scheduling algorithm in which each reader only needs to estimate its *local* spatial distribution of tags. As assumed always, every reader knows its geometry location. We partition the whole space into cells with side length $\max(2r, R)$. Then every reader knows exactly which cell it belongs to. See Figure 3 for illustration. For centralized algorithm, our method guarantees finding a $(1-1/k)^2$-approximation of MWIS at each slot t by finding the best partition. Clearly, this is impossible when we need low-complexity distributed scheduling. Similarly to the approach used in Algorithm 2, we will adopt the *pick and compare* approach. Randomly picking a partition (using random (a_t, b_t) at time slot t) guarantees that, with probability at least $1/k^2$ we will end up with the best partition, and a MWIS whose weight is at least $(1-1/k)^2$ of the optimum. The challenge now is to compare such candidate solution I'_t with previous solution I_{t-1} and then find the better one efficiently.

Since Algorithm 2 is a centralized algorithm, it is trivial to compute the candidate scheduling I'_t under a random partition, and it is also easy to compare $\omega(I'_t, t)$ with $\omega(I_{t-1}, t)$ based on global information. However, when implementing the algorithm in a distributed manner, we must deal with the following challenging issues:

1) If we compute the scheduling locally, how do we ensure that the union of all *local* solutions $I_t^{(i,j)}$ is still a valid scheduling? In other words, how can we guarantee that $\bigcup_{i,j} I_t^{(i,j)}$ is still RTc free even when each local scheduling $I_t^{(i,j)}$ is valid?
2) By assuming the union of local scheduling $\bigcup_{i,j} I_t^{(i,j)}$ is still valid, the next challenging problem is, if we make a selection between current candidate scheduling and previous one *locally*, how do we ensure that the union of all local selections is still a correct global solution? This indicates, by following a local selection, the desired algorithm should ensure that the total weight under resulted scheduling is not smaller than the previous one from a global point of view.

To address these issues, for a *square*(i, j) partitioned in time t, when we compute a solution $I_t^{(i,j)}$, we will compare the local optimum solution under X_t, with some special partial solution of I_{t-1} that are locally known to *square*(i, j), and the one with larger weight is chosen as $I_t^{(i,j)}$.

To describe our method in detail, we define some terms first. A *sub-square*(i, j) is the set of cells: $\{cell(x, y) \mid x \in [i*k+a_t+2, (i+1)*k+a_t-1]y \in [j*k+b_t+2, (j+1)*k+b_t-1]\}$. A *super-square*$(i, j)$ is the set of cells $\{cell(x, y) \mid x \in [i*$

$k+a_t+1,(i+1)*k+a_t]y \in [j*k+b_t+1,(j+1)*k+b_t]\}$. Clearly, the collection of super-squares is a *space partition*. A *sub-square*(i,j) is contained inside the *super-square*(i,j). See Figure 3(b) for illustration, where the larger square region is a *super-square*(i,j) and the smaller one is a *sub-square*$(i+1,j)$. In the following contents, let $I_{t-1}^{(i,j)}$ be the set of active readers from I_{t-1} that fall inside *super-square*(i,j) at time slot t.

At any time slot t, we will "remove" the readers positioned within either vertical strips i **and** $i+1$ with $i = a_t \mod k$ or horizontal strips j **and** $j+1$ with $j = b_t \mod k$, *i.e.*, readers are inside the gray region of Figure 3(b) will be removed. Note that in order to ensure that the union of $I_t^{sub(i,j)}$ is still valid, we remove *two* consecutive strips for every k strips instead of one. Our method then works as follows:

At time slot 0, we do a partition using $(a_0,b_0) = (0,0)$. Then we compute an optimum MWIS $I_0^{(i,j)}$for each *sub-square*(i,j) that is not empty of readers.

For any time-slot t, we partition the space using (a_t,b_t). We choose (a_t,b_t) as $(t \bmod k, t \bmod k)$ when $k \geq 5$. Observe that when $k=3$ (or 4), some cells will always be "removed". Therefore, when $k=3$ (or 4), we let (a_t,b_t) map to one distinct partition of the total 9 (or 16) different partitions (we can use a random partition), the partition will repeat after k^2 time-slots. We then compute the optimum MWIS, $I_t^{sub(i,j)}$, for all *sub-square*(i,j) under current tag distribution X_t. Recall that we denote $I_{t-1}^{(i,j)}$ as the set of activated readers from I_{t-1} that fall in *super-square*(i,j) at time t.

This set can clearly be computed locally as follows: If $\omega(I_{t-1}^{(i,j)},t) > \omega(I_t^{sub(i,j)},t)$, let $I_t^{(i,j)} = I_{t-1}^{(i,j)}$, otherwise $I_t^{(i,j)} = I_t^{sub(i,j)}$. Then we use $\bigcup_{i,j} I_t^{(i,j)}$ for all super-squares as a global solution.

Please refer to Algorithm 3 for details. Note that for each *super-square*, one reader will be selected as the only coordinator. It then computes the activation set inside this *super-square* (by computing a candidate solution in *sub-square* and comparing it with previous scheduling within *super-square*) under current tag distribution. We further assume the message RESULT($I_t^{(i,j)}$) contains all the necessary information of the activation set selected by the coordinator inside *super-square*(i,j) in time slot t. Initially, each reader has color White; if it is selected to be active, it will color itself Red , and Black otherwise.

We first show that the resulted scheduling I_t is indeed a valid scheduling (proof omitted due to space limit).

Theorem 5: I_t generated by Algorithm 3 is valid.

As shown before, the selection for $I_t^{(i,j)}$ is made locally by comparing $\omega(I_{t-1}^{(i,j)},t)$ and $\omega(I_t^{sub(i,j)},t)$. However, since I_t is decided by uniting all local selections, will it still be a good scheduling compared with I_{t-1}, *i.e.*, does $\omega(I_t,t) \geq \omega(I_{t-1},t)$ still hold? The following theorem gives us a positive answer.

Theorem 6: The scheduling I_t which is randomly generated by Algorithm 3 is no worse than I_{t-1} in terms of weight under current tag distribution X_t.

Proof: Based on our local comparison scheme, we have

Algorithm 3 Distributed Scheduling by a reader v

Input: k, a_t, b_t.
Output: Active or not for each reader at time slot t.

1: state = White; active = NO; Coordinator = NO;
2: Calculates which cell Z reader v resides in regarding to the current partition(k, a_t, b_t,);
3: Coordinator=YES if v is the coordinator for all readers in the same square.
4: **if** Coordinator = YES **then**
5: Collect the tag distribution $X_t(i)$ for all readers within the same *super-square*(i,j), and the scheduled readers $I_{t-1}^{(i,j)}$ in current *super-square*(i,j) at previous time-slot $t-1$ also.
6: Computes MWIS $I_t^{sub(i,j)}$ in *sub-square*(i,j);
7: If $\omega(I_{t-1}^{(i,j)},t) > \omega(I_t^{sub(i,j)},t)$, we set $I_t^{(i,j)} = I_{t-1}^{(i,j)}$; otherwise, we set $I_t^{(i,j)} = I_t^{sub(i,j)}$.
8: Broadcasts RESULT($I_t^{(i,j)}$) in *super-square*(i,j);
9: **if** state= White **then**
10: **if** receives message RESULT($I_t^{(i,j)}$) **then**
11: **if** $v \in I_t^{(i,j)}$ **then**
12: state = Red; active=YES;
13: **else**
14: state = Black; active=NO;

$\omega(I_t^{(i,j)},t) \geq \omega(I_{t-1}^{(i,j)},t)$ for any *super-square*(i,j). Since the collection of super-squares is a *space partition*, and $I_t = \bigcup_{i,j} I_t^{(i,j)}$, we further have $\omega(I_t,t) = \omega(\bigcup_{i,j} I_t^{(i,j)},t) \geq \omega(\bigcup_{i,j} I_{t-1}^{(i,j)},t) = \omega(I_{t-1},t)$. This finishes the proof. ■

Lemma 5: When $k \geq 5$, Algorithm 3 has a probability of at least $\frac{1}{k}$ to generate a valid scheduling set with weight at least $(1-\frac{4}{k})$ of optimal solution, *i.e.*, $\Pr\left(\omega(I_t,t) \geq (1-\frac{4}{k})\omega(I_t^{OPT},t)\right) \geq \frac{1}{k}$. When $k=3$ or 4, Algorithm 3 has a probability of at least $\frac{1}{k^2}$ to generate a valid scheduling set with weight at least $(1-\frac{2}{k})^2$ of optimal solution, *i.e.*, $\Pr\left(\omega(I_t,t) \geq (1-\frac{2}{k})^2\omega(I_t^{OPT},t)\right) \geq \frac{1}{k^2}$.

Proof: Remember that $(a_t,b_t) = (t,t)$ when $k \geq 5$, therefore we have k different partitions. Each cell(i,j) appears in the "removed" strips for at most 4 times. Suppose the optimal solution for time slot t is $\omega(I_t^{OPT},t)$, then there exists at least one good partition such that the removed part from the optimal solution, *i.e.*, weight of the readers located in the gray area, is at most $\frac{4}{k}\omega(I_t^{OPT},t)$. Since the result generated by Algorithm 3 for this good partition is optimal in the remaining area, $\omega(I_t,t)$ is at least $(1-\frac{4}{k})\omega(I_t^{OPT},t)$. Therefore the best partition generates a valid scheduling set with weight at least $(1-\frac{4}{k})$ of optimal solution. With probability $\geq \frac{1}{k}$, (t,t) is the best partition.

When $k=3,4$, there are k^2 different partitions. Each cell is "removed" for exactly $4k-4$ times. The rest of the proofs follow similarly with the case $k>4$. ■

Theorem 7: Algorithm 3 achieves $(1-\frac{4}{k})\mathcal{C}$ capacity for any $k \geq 5$ and $(1-\frac{2}{k})^2\mathcal{C}$ capacity for any $k=3$ or 4.

The claim holds by Proposition 2 and Lemma 5. The efficiency ratio can be improved to $(1-\frac{2}{k})^2$ using random (a_t,b_t)

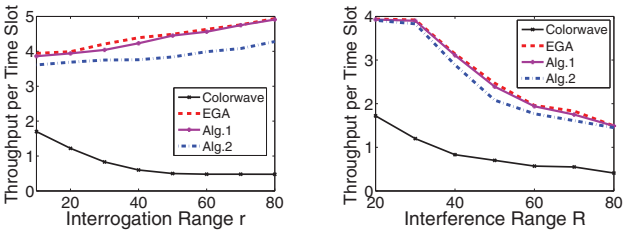


Fig. 4. Throughput evaluation in *static* environment with the increase of the interrogation radius r and the interference radius R.

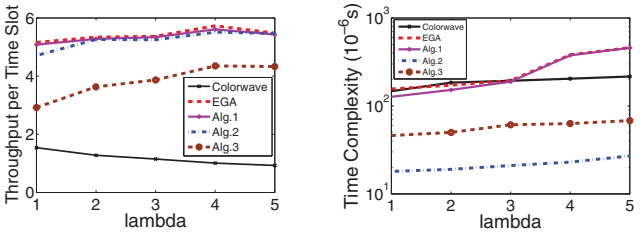


Fig. 5. Throughput and time complexity evaluation in *dynamic* environment with the increase of λ.

partition for $k \geq 5$.

VI. Performance Evaluation

We conducted extensive simulations with our own simulator to study the time complexity (the time needed by readers to find the activation scheduling) and stability of our methods.

A. Throughput Performance

First, we evaluate the performance of our methods in terms of numbers of tags that can be read on average in one-slot in both static and dynamic environments.

1) Static Scenario: For *static* scenario, we uniformly and randomly distribute 50 readers and 1200 tags in a square region of side-length 100 units. We compare our methods with the Colorwave algorithm [16] and EGA algorithm [4]. Figure 4 shows the performance with varying interrogation range r and interference range R respectively, with $k = 5$. All algorithms perform worse with the increase of R while all work better with the increase of r. As shown in Figure 4, our schemes are more effective. During the tag reading process, EGA algorithm tries to find a best b_t for each a_t in every sub-rectangles of the subgraphs partitioned by a_t. As discussed in Section IV, our centralized deterministic method (Algorithm 1) computes a best (a_t, b_t) pair from a global point of view. In other words, we trade some read throughput for less computing complexity. Observe that there is no noticeable throughput difference between these two algorithms. As shown in Figure 4(a), Algorithm 1 produces nearly the same throughput as EGA, *e.g.*, the average number of tags to be read in each time slot computed by our scheme is approximately optimal with significantly reduced time complexity. Algorithm 2 performs a little worse than those two algorithms.

2) Dynamic Scenario: To simulate the *dynamic* multi-reader environment, we assume the arrival rates of tags as a Poisson distribution with arrival rate λ. The interrogation and interference regions are disks with radius $r = 15$ and $R = 35$ units respectively. We set a random distribution of 1200 tags in the region as an initial state. Figure 5(a) shows the performance of various algorithms with varying λ. We compare the total time slots taken by various methods to finish the tag reading in the monitoring region. As expected, Algorithms 1 and 2 perform significantly better than Colorwave algorithm for all range of values, which is similar to what we observed in the *static* scenario. We notice that the performances of Colorwave worsen to infinite with increase of λ due to its random access scheme. In contrast, our distributed protocol serves more and more tags in each time slot with the increase of arrival rate of tags, and it is always comparable with the centralized methods. Moreover, our centralized algorithms perform nearly the same with the EGA algorithm as λ increases. This exemplifies the effectiveness of our algorithms.

B. Time Complexity

Before comparing our algorithms with related work in terms of stability, another experiment is designed to evaluate the time cost by various algorithms to find scheduling in *dynamic* tag reading scenario. We run our programs on a PC with dual core dual CPU, 2.53GHz, 2GB RAM. We take 700 iterations for each algorithm, with λ range from 6 to 20. The statistics we got from the experimentation are presented in Figure 5(b), with time unit of microsecond. As described in above static evaluation, we still distribute 50 readers and 1200 tags to participate in the tag reading process in the rectangular region. Figure 5(b) shows the performance with varying interrogation range r and interference range R respectively, with $k = 5$. Obviously, in contrast to EGA algorithm, our randomized algorithms have very low time complexities.

C. Stability of the protocols

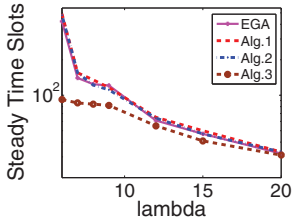


Fig. 6. Stability evaluation in *dynamic* environment.

We now evaluate the stability performance of our algorithms with an even bigger value of λ, such that the reading process may not terminate in a regular way. Recall that we define a **stable** scheduling as a scheduling under which the expected amount of unread tags is always bounded. Here the expected bound serves as an input parameter in our simulation. Specially, we adjust the expected average unread bound for readers from 180 to 220 as λ changes, *i.e.*, the expected bound will be slightly increased while the arrival rate of tags increases. In this way, the bias for the performance evaluation due to different new tags arrival rates in each time slot can be alleviated, and the reader scheduling schemes will play a more important role. Here we

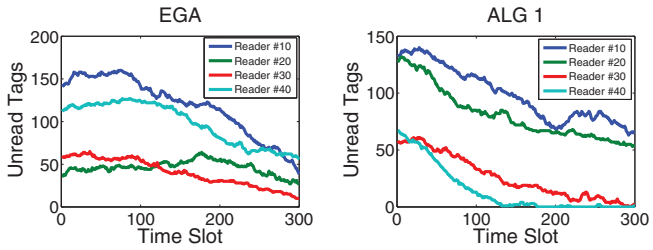


Fig. 7. *Dynamic* environment: Number of unread tags vs. time slot vs. reader ID, with λ=3.

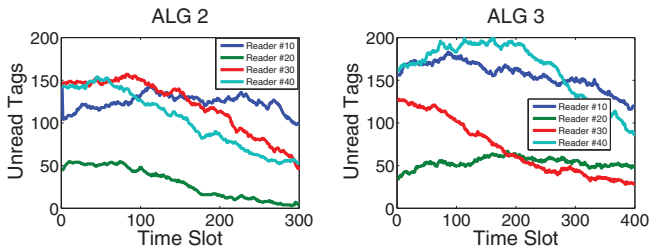


Fig. 8. *Dynamic* environment: Number of unread tags vs. time slot vs. reader ID, with λ=3.

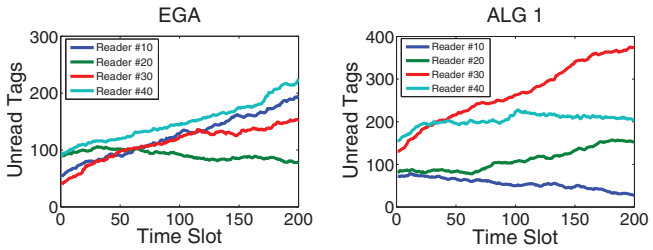


Fig. 9. *Dynamic* environment: Number of unread tags vs. time slot vs. reader ID, with λ=7.

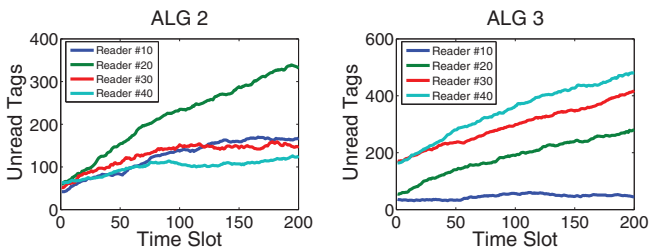


Fig. 10. *Dynamic* environment: Number of unread tags vs. time slot vs. reader ID, with λ=7.

randomly distribute 1200 tags at the beginning, with $r = 15$ and $R = 35$ units respectively. As the tag reading process runs, we collect the average numbers of unread tags in the monitoring region for each time slot. The running process will be suspended in the time slot when the average number of unread tags for each reader exceeds the expected value. We evaluate the stability of the above algorithms in terms of the total number of running time slots till the reading process terminates. Figure 6 shows the performance of various algorithms as λ increases.

In Figure 6, all algorithms suspend at an earlier time as the arrival rate of tags gets bigger. It is shown that our designed centralized algorithms perform slightly better than EGA algorithm for almost all range of values, *e.g.*, for the same tag arrival rate, Algorithm 1 and Algorithm 2 will stay in stable state as long as EGA algorithm or even better. To minimize the total reading time, EGA algorithm tries to have each location in the monitoring region be well-covered by a reader in some time slot. It is straightforward that unread tags accumulated at some readers tend to exceed the expected bound before getting served as more tags arrive in each time slot. We also notice that the performance of our distributed algorithm are comparable with centralized algorithms as λ increases, and are similar to them for bigger values.

Figure 7 and Figure 8 show the number of accumulated unread tags within each reader's (we randomly select four readers as an illustration) interrogation range by every time slot. Here we set $\lambda = 3$ to ensure that all algorithms can terminate. It is shown that for a given λ of small value, all algorithms experience an accumulation of unread tags at the beginning. This is because there are a number of unread tags covered by a reader initially. As more and more tags get served, the number of unread tags decreases gradually. In the results reported here with $\lambda = 3$, the number will decrease to zero since the average throughput is larger than the arrival rate of new tags. As shown in the simulation results, EGA algorithm takes the least time to finish the reading process at the price of stability loss, *i.e.*, accumulating more unread tags for several time slots. In contrast, Algorithm 1 has the best stability performance. Note that Algorithm 1's corresponding curves are smoother, which indicates the ever decreasing number of unread tags accumulated within the readers' interrogation range. Algorithm 1 performs similarly to EGA algorithm, with similar peak points and run way. All results listed above indicate that our centralized schemes performs as good (or even better) as EGA algorithm in terms of stability with significantly lower complexity. Moreover, our distributed algorithm remains stable for long term, even though it takes more time to finish the process.

We then study the case when the arrival rate of tags exceeds the average throughput. It is straightforward that unread tags will be accumulated in this case. Here we set $\lambda = 7$ and record the number of unread tags for each reader scheduled by centralized algorithms in the first 200 time slots, while recording the according statistics in the first 150 time slots for our distributed scheduling algorithm. Figure 9 and Figure 10 show that our centralized schemes perform better in terms of reducing average number of accumulated tags. By the 200th time slot, the number of unread tags for each reader scheduled by EGA algorithm is less than that scheduled by our centralized randomized scheduling scheme. However, the average number of unread tags for all readers is not less than ours since more readers have lots of unread tags. The total number of unread tags scheduled by EGA is larger, which

indicates lower stability. Observe that Algorithm 1 still has the best stability since both the number of unread tags for each reader and the number of readers with lots of unread tags is smaller than others. Our distributed algorithm also achieves acceptable stability in terms of minimizing the number of readers with too many unread tags.

VII. Related Work

Recently, several protocols have been proposed to avoid collisions in RFID systems. We classify these results into three categories according to the type of collisions they addressed.

Several link layer protocols, *e.g.*, [7], [11], [13], [15], have been proposed in order to avoid TTc. For example, one of the most popular solutions is the tree walking algorithm (TWA) [11], [13]. In TWA, entire ID space of tags is partitioned into two subsets, and the readers try to recognize the tags belonging to one of the subsets. Running recursively along this way until a subset has no tag at all or exactly one tag. Recently, [14] proposes optimizations to tree traversal. In slotted Aloha protocol [6], a query frame is selected with a sufficiently large number of time slots and each tag sends a response at a random chosen time slot. When the reader hears a response correctly, it sends confirmation. If there is a collision, the colliding tags will choose another random slot to send a response.

For avoiding RRc or RTc, Colorwave [16] is one of pioneer works to study RRc. Specially, it constructs an "interference graph" over the readers, wherein there is an edge between pair of readers if and only if they may cause RRc when transmitting at the same time. Colorwave then tries to color the readers randomly such that each pair of interfering readers can gain different colors. In [9], the authors suggest k-coloring of the interference graph in which k is the number of available channels. In the recent EPCGlobal Gen 2 standard [5], the authors propose a dense reading mode, in the dense reading model the tag responses happen in different channels but not the readers.

In [10], they design a Q-learning process. After a training period, it adopts a resource (channel and time slot) allocation scheme. The training process determines allocation of channel and time slots to different readers when a new tag read request arrives. There is no performance guarantees provided in this paper. Recently, [4] proposes a tag access scheduling protocol (EGA) based on STDMA. In the case where the tag distribution is known, they assign time slots to readers, such that each location of the deployment space is well-covered by some reader among one of the time slots. They generalize and optimize their solution for the model where the tag distribution is unknown and multi-channels are available. They assume that the tag distributions are static and no new tags will appear in the system.

VIII. Conclusion

We proposed an efficient multiple-reader scheduling protocol, RASPberry, for scheduling RFID readers such that it will reduce the number of unread tags. Our results can be easily extended to the case when a tag should be read periodically after it arrived, by adjusting the arrival rates of tags accordingly. When the number of tags is more than the system capacity (*i.e.*, system is not stable), some tags will not be read. Although at every timeslot there may be tags that are not read, every tag will be read within a small number of time-slots after the tag arrived when the system is stable.

An interesting question left for future research is to study the placement of readers to maximize the overall read throughput, when we can estimate and predict the tag arrivals. Another challenge is to relax the Poisson distribution of the tag arrivals.

IX. Acknowledgment

The authors would like to thank Jakob Eriksson for his constructive feedback. The research of authors is partially supported by NSF CNS-0832120, National Natural Science Foundation of China under grants No. 60828003 and No. 60825205, the Natural Science Foundation of Zhejiang Province under Grant No.Z1080979, National Basic Research Program of China (973 Program) under grants No. 2010CB328100 and No. 2006CB30300, the National High Technology Research and Development Program of China (863 Program) under grant No. 2007AA01Z180, Hong Kong RGC HKUST 6169/07, the RGC under Grant HKBU 2104/06E, and CERG under grant PolyU-5232/07E.

References

[1] X. Lin And N.B. Shroff The impact of imperfect scheduling on cross-layer congestion control in wireless networks. *IEEE/ACM Trans. Netw. 14*, 2 (2006), 302–315.

[2] S. Sanghavi, L. Bui, and R. Srikant Distributed link scheduling with constant overhead. In *ACM SIGMETRICS* (2007), pp. 313–324.

[3] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Trans. on Automatic Control*, 37(12):1936-1948, December 1992.

[4] Zongheng Zhou and Gupta, H. and Das, S.R. and Xianjin Zhu Slotted Scheduled Tag Access in Multi-Reader RFID Systems. In *IEEE ICNP 2007*

[5] Class 1 gen. 2 UHF air interface protocol standard version 1.0.9, 2005.

[6] N. Abramson. The ALOHA system - another alternative for computer communication. In *Joint Computing Conference Proceedings*, 1970.

[7] I. Chlamtac, C. Petrioli, and J. Redi. Energy-conserving access protocols for identification networks. In *IEEE/ACM ToN*, 1999.

[8] V. Deolalikar *et al.* Perturbative time and frequency allocations for RFID reader networks. Technical report, HP Labs, 2005.

[9] H. Gupta *et al.* Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *IEEE/ACM ToN*, 2006.

[10] J. Ho, D. Engels, S. Sarma., and Hi Q A hierarchical q-learning algorithm to solve the reader collision problem. In *SAINTW*, 2006.

[11] D. Hush and C. Wood. Analysis of tree algorithms for RFID arbitration. In *Proc. of ISIT*, 1998.

[12] S. Jain and S. R. Das. Collision Avoidance in a Dense RFID Network. In *WiNTECH*, 2006.

[13] C. Law, K. Lee, and K. Siu. Efficient memoryless protocol for tag identification. In *Intl. Wksp. on Dis. Algos. and Methods*, 2000.

[14] J. Myung andW. Lee. Adaptive splitting protocols for RFID tag collision arbitration. In *ACM MobiHoc*, 2006.

[15] H. Vogt. Efficient object identification with passive RFID tags. In *Proc. of the Intl. Conf. on Pervasive Computing*, 2002.

[16] J. Waldrop, D. Engels, and S. Sarma. Colorwave: A MAC for RFID reader networks. In *IEEE WCNC*, 2003.

[17] S. E. Sarma, S. A. Weis, and D. W. Engels, RFID Systems and Security and Privacy Implications," In *Workshop on Cryptographic Hardware and Embedded Systems*. LNCS, vol. 2523, 2002, pp. 454-470.

[18] M. Kodialam and T. Nandagopal. Fast and reliable estimation schemes in RFID systems. In *ACM MobiCom*, 2006.