

# Reader Anti-Collision in Dense RFID Networks With Mobile Tags

Essia Hamouda<sup>1,2</sup>, Nathalie Mitton<sup>2</sup> and David Simplot-Ryl<sup>2</sup>

<sup>1</sup> University of California Riverside, <sup>2</sup>INRIA Lille-Nord Europe, Univ. Lille 1, CNRS

<sup>1</sup>essia@cs.ucr.edu, <sup>2</sup>{nathalie.mitton,david.simplot-ryl}@inria.fr

**Abstract**—In a Radio-Frequency Identification network, while several readers are placed close together to improve coverage and consequently read rate, reader-reader collision problems happen frequently and inevitably. High probability of collision not only impairs the benefit of multi-reader deployment, but also results in misreadings in moving RFID tags. In order to eliminate or reduce reader collisions, we propose an Adaptive Color based Reader Anti-collision Scheduling algorithm (*ACoRAS*) for 13.56 MHz RFID technology where every reader is assigned a set of colors that allows it to read tags during a specific time slot within a time frame. Only the reader holding a color (token) can read at a time. Due to application constraints, the number of available colors should be limited, a perfect coloring scheme is not always feasible. *ACoRAS* tries to assign colors in such a way that overlapping areas at a given time are reduced. To the best of our knowledge *ACoRAS* is the first reader anti-collision algorithm which considers, within its design, both application and hardware requirements in reading tags. We show, through extensive simulations, that *ACoRAS* outperforms several anti-collision methods and detects more than 99% of mobile tags while fitting application requirements.

## I. INTRODUCTION

Radio Frequency IDentification (RFID) systems consist of Radio Frequency (RF) tags and networked RF tag readers. They are primarily designed to identify objects with unique identifiers. Readers initiate communication with the tags and query them [6]. One of the major issues in RFID systems is communication interference that leads to collision problems. Collisions may be caused by reader-to-reader collision [3], reader-to-tag collision [16] or tag-to-tag collision [15].

Methods, proposed in the literature to solve such collision problems, focus mainly on tag collision problems to ensure the detection of all tags laying in the reader field. Best existing algorithms [2], [15] are part of patents [16] and are integrated in market solutions. They allow to detect more than 200 tags per second for 13.56 MHz tags in a reader-collision-free environment. Little attention, however, has been given to reader collision problems which occur when the fields of two readers overlap in space. A tag located in this overlapping (or interference) area will not be able to send its data to any reader. As a result, two or more readers can be active at the same time and are able to successfully read tags if and only if they do not interfere. This technology is applicable to several fields such as distribution centers, inventory, transportation, e-passports, baggage tracking in airports etc<sup>1</sup>. In distribution

centers for example, RFID readers are deployed in traditional choke points where tag recording is primarily limited to entry and exit through key areas, such as dock doors. The objective is to keep track of inventory (what comes in and what leaves the warehouse). Note that in this case both readers and tags are stationary at reading time.

In this work, we consider an RFID-device network where tags are mobile and readers are fixed. As an application, consider a hospital where administrators need to track valuable medical equipments and products that move constantly and must be cleaned regularly. Deploying fixed readers in the hospital and integrating tags in equipments and products allows administrators to keep track of such assets at any time. We focus on the problem of RFID reader-reader collision problem to model the task of activating a set of RFID readers over time. More specifically, we propose a reader activity scheduling algorithm that minimizes reader collision and increases throughput. The algorithm must also comply with hardware and application requirements. We propose *ACoRAS* an Adaptive Color based Reader Anti-collision Scheduling algorithm which assigns colors or tokens to readers. *ACoRAS* runs in two steps. In the first step, readers are assigned colors that allow them to communicate. The number of colors a reader can hold may be greater than one (the larger the number of colors a reader holds the more chances it is given to communicate). However, due to the number of tags to be read, the number of colors and thus the reading time slot may not handle such an assignment. Therefore, the second step aims at reducing the number of colors a reader can hold to satisfy system requirements. In the case that no color needs to be removed, *ACoRAS* ensures that every tag that spends at least  $T_{min}$  seconds in the field of a reader is detected by optimizing the number of active readers and ensuring no reader collision. Our extensive simulations show *ACoRAS* outperforms existing methods. Results show that the proportion of unread tags is less than 0.75% for various number of readers and various number of tags.

The rest of the paper is organized as follows. Section II describes our problem, its constraints and its assumptions. Section III provides a literature review of the subject. Section IV presents our algorithm which we evaluate in Section V. Section VI concludes this paper by exploring future research and extensions to this work.

This work was partially supported by FP7 Aspire European project and by CPER Nord-Pas-de-Calais/FEDER Campus intelligence ambiante.

<sup>1</sup><http://www.ti.com/rfid/shtml/apps.shtml>

## II. PROBLEM STATEMENT

In this section, we define our objective, present our problem and system requirements and introduce the network model.

**Objective:** The main motivation of this work is the challenging system requirements faced by RFID applications nowadays. Our aim is to provide an efficient and scalable solution to reduce reader collision and optimize tag detection while ensuring application and hardware requirements. To the best of our knowledge, existing reader anti-collision methods lack to incorporate such requirements in their design.

**Reader collision:** In passive RFID, tags have no energy embedded. They are activated only when they pass through the electromagnetic field of a reader. When a wave bounces on a tag, the reader can read data stored in the tag [6]. However, when a tag enters an interference area, reader electromagnetic fields will overlap, transmitted signals will collide thus, tags will be unable to answer readers queries. This is called reader collision. The tag then becomes unresponsive and according to the kind of tags, may not be detected until it leaves all readers' fields (and not just the interference area). It will be responsive again once it enters an area where there is no active field. For instance, in Figure 1, only tags  $T1, T2, T5$  and  $T7$  are detected by readers  $R1, R2$  and  $R3$ . Although they are within range of readers  $R1$  and  $R2$ , tags  $T4$  and  $T6$  are not read since they are in an interference zone of two different readers. As for  $T3$ , it is not detected because it is out of range of the available readers.

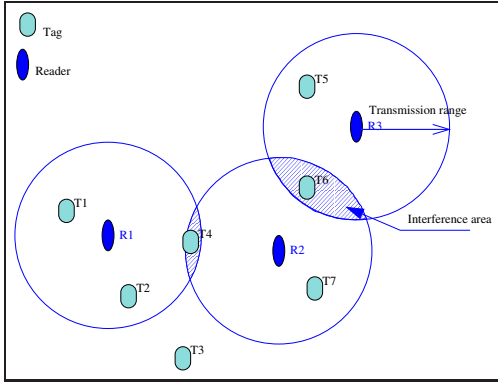


Fig. 1. Readers-to-reader collision.

**Application requirements:** RFID-tagged components are generally carried on moving equipments such as conveyors in warehouses. We assume that tags move at a given constant speed and will spend at least  $T_{min}$  seconds traversing the reader coverage area. One of the application requirements is to ensure that tags that spend at least  $T_{min}$  seconds in the coverage area of a reader have to be detected and read. If the tag spends less than  $T_{min}$  seconds in a field, we assume that it will eventually enter another reader coverage area and will be read. Therefore, every reader has to be activated at least once every  $T_{min}$  seconds.

**Hardware requirements:** In a collision-free environment, a reader requires  $t \times N_{tag}$  seconds to read a set of  $N_{tags}$  tags in its field where  $t$  is the minimum amount of time needed to read a tag ( $t$  is set to 5ms for 13.56 MHz tags [2]). Consequently, to ensure that every tag laying in a reader field is detected,

the reader has to be ON during at least  $t \times N_{tag}$  seconds. Therefore, if we associate colors to readers as communication tokens, to read all  $N_{tags}$  tags, then the maximum number of colors  $n_{max}$  allocated to each reader must be  $n_{max} = \frac{T_{min}}{t \times N_{tag}}$ . **Network Structure:** We focus on the reader network and denote by  $V$  the set of readers. We model the reader network as a graph  $G = (V, E)$  where  $V$  is the set of vertices (readers) and  $E$  is the set of edges (interference links). We assume there exists a link  $(uv) \in E$  between two readers  $u$  and  $v$  if and only if readers  $u$  and  $v$  share an interference area (fields where the transmission areas of  $u$  and  $v$  overlap). This is illustrated in Figure 3 where Figure 3(b) plots the graph associated with the scenario depicted by Figure 3(a)<sup>2</sup>. Let  $N(u)$  be the set of neighbors of reader  $u$ , i.e.  $N(u) = \{v | (uv) \in E\}$ . Let  $\Delta_{max} = \max_{v \in V} |N(v)|$  be the maximal degree in graph  $G$ . We assume that this graph is computed in a centralized way, where every reader has a knowledge of overlapping fields.

## III. RELATED WORK

While reader-tag and tag-tag [15], [9], [13] collisions have been widely studied and incorporated in communication network normalization and standards (EPCGen2 [16], ETSI<sup>3</sup>), reader-reader anti-collision mechanisms still remain an important research issue. Reader-reader anti-collision protocols from the literature are of several kinds. Some, like HiQ-learning [10] use a hierarchical architecture to provide an online learning of collision patterns of readers and assign frequencies to the readers over time. Others, such as [1], [7], [11], [12] apply a carrier sense multiple access (CSMA) based algorithms to detect collisions. These solutions, although efficient, cannot be used in mobile tag applications since they either need a stabilization time to learn certain patterns or are probabilistic. None of the above solutions ensure that every tag spending a minimum time in a reader area will be detected.

Some time division multiple access (TDMA) based reader-reader anti-collision mechanisms have been proposed in the literature. The idea is that every reader is assigned a color and is allowed to read only during the time slot that corresponds to its color. The main difficulty faced by such anti-collision schemes lies in the color assignment which is generally done in such a way that two nodes sharing an edge should not have the same color. Finding such a coloring scheme is NP-hard [5].

In probabilistic reader anti-collision algorithms [18], [17], [3] a reader selects a time slot, and if it is the only reader in the slot, it is granted that time slot to identify the tags. Otherwise, it backs off and tries at a later time.

Centralized solutions [14] use an iterative procedure. A reader is allocated a color in an order determined based on the number of neighboring readers already colored, choosing at each step the color with the smallest identifier/color, the assignment process continues till all readers hold a color.

In Distributed Color Selection (DCS) [18], each reader randomly selects a time slot in a frame for transmission. If the current color of a reader corresponds to the time slot for

<sup>2</sup>Note that for sake of simplicity, the field shapes of readers are represented as disks but our model and our algorithm can be applied to any field shape.

<sup>3</sup><http://www.etsi.org>

reading, the reader proceeds to identify tags during that time slot and reserves the same color in the next frame. In the case of a collision, readers randomly select and reserve new colors in the next frame. After the new color assignment, readers broadcast their colors to interfering neighboring readers in order to prevent them from colliding in the next frame. The readers then compare their colors for the next frame. If they are the same, they randomly change their colors to different colors among the maximum permissible colors. DCS with the fixed maximum color is simple. However, if the number of adjacent readers is much larger than the maximum number of colors, many readers may experience collisions. But when the maximum color size is greater than the number of readers, many colors may be wasted.

To cope with the problem of DCS, the Variable-Maximum DCS (VDCS or colorwave) [17] which can adjust the maximum number of colors has been proposed. In colorwave, each reader monitors a rate of success and dynamically changes its maximum color according to the percentage of successful transmissions during a certain period. Hence, readers suffering from many collisions increase their maximum colors, and readers with few collisions decrease their maximum colors. When collisions occur, unlike DCS, a reader chooses a new random color within the color range. Colorwave outperforms DCS because it is able to adjust the frame size by monitoring the collision probability. Both algorithms, however, allocate the same time slot size to readers regardless of the number of tags and thus do not consider hardware requirements (see Section II). In addition, in Colorwave, since the maximum number of colors dynamically changes over time, the reading cycle duration is also dynamic and thus Colorwave cannot always ensure the fulfillment of application requirements.

In [4], mobile readers communicate with a centralized server which grants service to readers for tag identification on a first-come-first-served basis. If the requests of some readers arrive at the server simultaneously, the server randomly arranges the order of service for the readers. This algorithm does not consider hardware requirements.

A common characteristic to existing color based anticollision algorithms is the assignment of a single color to readers. However, as we will show later, the larger the number of colors a reader holds, the higher the throughput is. Moreover, a rigorous color assignment method can tremendously reduce reader collision.

#### IV. ACoRAS

In this Section, we introduce ACoRAS, a novel anti-collision scheduling algorithm. It operates in two steps: a color assignment and a color refinement described in the following. Once colors are assigned to a time slots and to readers, every reader is activated during the slot corresponding to one of its colors. The scheduling scheme is summarized in Algorithm 1.

##### A. Initial setting

ACoRAS applies a TDMA based *scheduling* mechanism that allows readers to take turns reading mobile tags to avoid interfering with each other's readings. It starts by dividing time

in frames of length  $T_{frame}$  seconds. Every frame is further divided in  $n$  time slots of equal length  $T_{slot}$  seconds; *i.e.*,  $T_{frame} = n \cdot T_{slot}$ . As each slot is allocated a communication token represented by a color,  $n$  is then the number of colors used during the duty cycle  $T_{frame}$ . Clearly,  $T_{frame}$  is predefined by the application requirement. If every reader has to be activated at least once every  $T_{min}$ , we have to set  $T_{frame} = T_{min}$ . Since  $n$  is a variable,  $T_{slot}$  is also a variable that depends on  $n$ . The larger  $n$ , the smaller  $T_{slot}$  and the shorter the length of time during which readers are activated. To fulfill hardware requirements, it is easy to show that  $n$  cannot exceed  $\frac{T_{frame}}{t \times N_{tag}}$  (see Section II).

---

#### Algorithm 1 Schedule( $V, T_{frame}, t \times N_{tags}, n$ )

---

```

1:  $(\mathcal{C}, n) \leftarrow \text{Coloring}(V, T_{frame}, t \times N_{tags})$ 
2:  $T_{slot} \leftarrow \frac{T_{frame}}{n}$ 
3: for every slot do
4:   for all reader  $v \in V$  do
5:     if SlotColor  $\in \mathcal{C}(v)$  then
6:       {The color associated to the current slot belongs
7:        to the set of colors of reader  $v$ . }
8:     else
9:       Switch reader  $v$  to OFF
10:    end if
11:  end for
12: end for

```

---

##### B. Coloring Procedure

The general idea is to assign to each reader a set of colors and allow it to communicate with a tag only when it gets a hold of color. The color allows it to read tags only during the time slot associated to that color.

The challenge here is: *given a limited number of available colors, how should ACoRAS assign colors to neighboring readers to minimize their interference area?* Clearly, if two neighboring readers hold the same color, reader collision is imminent. However, two non neighboring readers may hold the same color. A reader may also hold more than one color as long as it does not share the same color with any of its neighbors. To achieve this objective, ACoRAS starts by assigning colors to readers to satisfy application and hardware constraints. Then it optimizes the color assignment by removing excess colors assigned to readers to reduce interference and tag losses. We will describe both steps in the following.

In ACoRAS, the color assignment function is centralized. The algorithm assigns every reader  $u$  a set of colors  $\mathcal{C}(u) \in \Omega$ , ( $\Omega = [0, \Delta_{max}]$  is the set of available colors) such that these colors are unique in the neighborhood of  $u$ , *i.e.*  $\mathcal{C}(u)$  is such that  $\forall c \in \mathcal{C}(u), \nexists v \in N(u) \mid c \in \mathcal{C}(v)$ . Since not all colors in  $\Omega$  may be used, we denote by  $\Theta \subseteq \Omega$  the set of colors actually used. Algorithm 2 summarizes the color assignment process. It returns a graph in which every vertex is assigned a set of colors.

1) *Initial Color Assignment:* In this section, the initial color assignment step is performed by function *ColorInit()*

---

**Algorithm 2** Coloring( $V, T_{frame}, t \times N_{tag}$ )
 

---

```

1:  $(\mathcal{C}, |\Theta|) \leftarrow \text{ColorInit}()$ 
2: if  $t \times N_{tag} \leq \frac{T_{frame}}{n}$  then
3:   {There are too many colors to allow every reader to be active long
   enough to read every tag within their range.  $n' = |\Theta| - \lfloor \frac{T_{frame}}{N_{tag}} \rfloor$ 
   colors have to be removed.}
4:    $(\mathcal{C}', |\Theta|) \leftarrow \text{RemoveColors}(n - \lfloor \frac{T_{frame}}{N_{tag}} \rfloor)$ 
5:   Return  $(\mathcal{C}', |\Theta|)$ 
6: else
7:   Return  $(\mathcal{C}, |\Theta|)$ 
8:   {Initial coloring provides a set of colors which already fulfills
   hardware requirements.}
9: end if
  
```

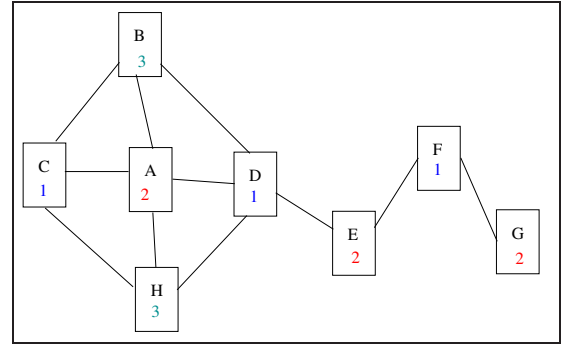
---

(Algo. 2 line 1). Besides providing a collision-free environment, *ACoRAS* and more specifically *ColorInit()* aims at maximizing network throughput. For instance, it can be verified that the smaller the number of colors used, the longer  $T_{slot}$  is and the more tags are detected. For instance, in Fig. 3(a), a 2-color scheduling algorithm can be efficient for a network composed of readers  $R1$ ,  $R2$ ,  $R3$ . Readers  $R1$  and  $R3$  can be assigned color  $c_0$  and thus be both active in slot 0 while reader  $R2$  is assigned color  $c_1$  and is active during slot 1. Now, consider the case of a non-homogeneous network where nodes are more concentrated in one area than in another (see Fig. 2(a)). Node  $G$  can be in conflict only with node  $F$  however since it is assigned a unique color, it can have a single communication slot leading to a very long unnecessary idle time and low network throughput. To avoid such drawbacks, *ACoRAS* initial color assignment aims at giving a set of colors to every reader in such a way that: (i) every color is unique within the neighborhood of a reader, (ii) the overall number of colors is minimized (minimize  $|\Theta|$ ) to reduce the complexity of the color-assignment scheme, (iii) there are no idle slots *i.e.* within a reader neighborhood and at any given time slot, either reader  $r$  or one of its neighbors will be active as long as there are tags to read.

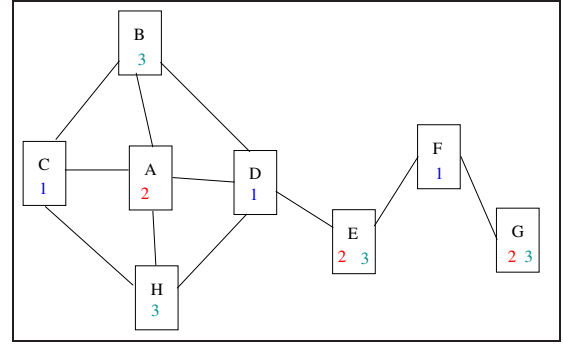
The initial color assignment (Algorithm 3) runs in two steps. The first step (line 2) uses the notion of minimum independent set [8] defined as follows.

**Definition 1 (Minimum Independent Set (MIS)):** The set  $A \subset V$  is a Minimum Independent Set (MIS) if  $\forall u, v \in A, (u, v) \notin E$ .

The coloring algorithm starts by building a MIS from the set of readers. Every node in the MIS is assigned the same color  $c$  and removed from the initial set of readers. Color  $c$  is removed from the set of available colors as well. This process reiterates till all initial nodes are colored. As an example, consider a reader network composed of nodes  $A - G$ . Fig. 2(a) shows the network as a graph where a MIS composed of readers  $C, D, F$  is computed. These readers are allocated color 1. A second MIS is computed based on readers  $A, E, G$ . In this case readers are assigned color 2. Finally, the last MIS is computed, it contains the remaining readers  $B$  and  $H$ . In the second step in Algorithm 3, readers are sorted according to their degrees. Readers with lower degrees generate less interferences. The algorithm then recursively adds a color to each reader till



(a) Not optimal color assignment



(b) Optimal coloring

Fig. 2. Reader-Initial Color assignment. Even though Fig. 2(a) plots a coloring example where a minimum number of colors has been used, it can not be considered as optimal since the neighborhood of reader  $G$  is silent during slot 3 and thus,  $G$  could use that slot without generating interference. *ACoRAS* thus allows nodes having several colors to enhance reading performance.

saturation or till all colors have been assigned. A color  $c$  is added to reader  $r$ , whose neighbor is  $u$ , if and only if (i)  $c$  has been selected in the first step ( $c \in \Theta$ ) to ensure that the number of colors does not increase and (ii)  $r$  is not assigned color  $c$  to avoid reader collisions. For instance, in Fig. 2(a), reader  $G$  with the lowest degree is first considered and assigned color 3. Reader  $E$  can be assigned color 3 also. At this point, no color can be added to the remaining readers (all nodes are saturated). Fig. 2(b) shows the final color assignment. Thus, the time frame will be divided into 3 time slots. Readers  $C, D$  and  $G$  will be active during time slot 1, reader  $A$  will be active during time slot 2 and readers  $E$  and  $G$  will be active during both time slots 2 and 3.

Once a color assignment is identified and a duty cycle duration is set, it is easy to compute the time slot duration:  $T_{slot} = \frac{T_{frame}}{n}$ . Two cases can be distinguished;

(i)  $t \times N_{tag} \leq T_{slot}$ : every reader can be activated during the slots corresponding to its colors, so there will be no collisions,

(ii)  $t \times N_{tag} > T_{slot}$ : in this case, hardware requirements are not fulfilled since readers will not have enough active time to detect all tags in their fields. To increase the time slot and consequently the required duty cycle duration, some colors have to be removed which leads to *ACoRAS* color assignment optimization step.

2) *Color Assignment Optimization:* In this step, the algorithm's objective is to adjust the number of colors assigned to readers. More specifically, the objective is to eliminate any colors (reduce  $|\Theta|$ ) that may cause any type of collision while maximizing throughput. Note that, this step has to be part

**Algorithm 3** ColorInit()

---

```

1: {input: a graph  $G(V, E)$  and a set of colors  $\Omega$ .}
2: {Step 1: set a first coloring assignment}
3:  $A \leftarrow V, \Theta \leftarrow \emptyset$ 
4: while  $A \neq \emptyset$  do
5:   {Compute a MIS on the set of remaining nodes}
6:    $B \leftarrow \text{MIS\_Compute}(A)$ 
7:   for all  $u \in B$  do
8:     {Every node of the MIS  $B$  is assigned the same color.}
9:      $c \leftarrow \text{rand}(\Omega \setminus \Theta), \mathcal{C}(u) \leftarrow \mathcal{C}(u) \cup \{c\}$ 
10:     $A \leftarrow A \setminus B$ 
11:   end for
12:    $\Theta \leftarrow \Theta \cup \{c\}$ 
13: end while
14: {Step 2: add colors among already used colors.}
15:  $\text{SortReadersByDegree}()$ 
16: {Sort readers in decreasing degree order in order to assign more colors to the ones generating the least conflict first}
17: while New color added do
18:   for  $\forall$  reader  $u$  in increasing degree order do
19:     if  $\exists c \in \Theta \mid \exists v \in N(u) \mid c \in \mathcal{C}(v)$  then
20:        $\mathcal{C}(u) \leftarrow \mathcal{C}(u) \cup \{c\}$ 
21:     end if
22:   end for
23: end while
24: Return  $\{\mathcal{C}, |\Theta|\}$ 

```

---

of the algorithm for system requirements as well. The more colors we use, the smaller the time slot  $T_{slot} = \frac{T_{frame}}{|\Theta|}$  is, keeping in mind that  $T_{slot}$  cannot be smaller than the required time to read tags in a reading area ( $t \times N_{tag}$ ). We can then infer that  $n' = |\Theta| - \left\lceil \frac{T_{frame}}{t \times N_{tag}} \right\rceil$  is the number of unused colors that can be removed.

The RemoveColors( $c$ ) function in Algo. 4 iterates  $n'$  times. It first selects the color which has to be removed (in line 2) then, only when needed, it replaces it with another color. For scalability, the algorithm selects the color that is allocated to the smallest number of readers. Obviously, if every reader is holding a single color, this step is omitted from the algorithm. In Fig. 3 for example, all readers are holding a single color except for  $R5$ . Therefore, one of these colors, 2 or 3, has to be removed. If color 2 (color 3) is removed, a new color has to be assigned to reader  $R2$  ( $R4$ ) as it is holding a single color. In the general case, a color is removed from all readers holding it and replaced only at readers holding that single color.

Assigning a new color (to reader  $R4$  in the previous example) should be done carefully so no interference will be introduced while maintaining a small number of colors. Step 2 (line 10) of RemoveColors( $c$ ) function chooses as new color that minimizes the area of the interference region not covered by another color (but covered by another reader in a different time slot). This step can be computed by computing the number of area units overlapping. Note that minimizing interference alone is not enough as we may end up with floating tags that are not covered by any reader. In our previous

**Algorithm 4** RemoveColors( $c$ ).

---

```

1: while  $c \neq \emptyset$  do
2:    $a \leftarrow \text{LessUsedAlone}()$ 
3:   {Color  $a$  has been selected as the one to be removed.}
4:    $\Theta \leftarrow \Theta \setminus \{a\}$ 
5:   for all  $u \in V$  do
6:     if  $a \in \mathcal{C}(u)$  then
7:        $\mathcal{C}(u) \leftarrow \mathcal{C}(u) \setminus \{a\}$ 
8:     if  $\mathcal{C}(u) = \emptyset$  then
9:       {Reader  $u$  has no more color.}
10:       $\mathcal{C}(u) \leftarrow \text{MinimizeInterference}(\Theta)$  {Assign color which minimizes interference area in  $u$ 's neighborhood not covered at another time slot.}
11:    end if
12:  end if
13: end for
14: end while
15: Return  $(\mathcal{C}, |\Theta|)$ 

```

---

example and as an illustration of the algorithm, reader  $R4$  has to choose a new color. To do so, it selects one of the assigned colors and computes the corresponding interference area not covered by another reader at another time slot. Choosing color 4 for instance, would generate two interference areas one with readers  $R4$  ( $A_{4a}$ ) and one with reader  $R7$  ( $A_{4b}$ ) as illustrated by Fig. 3(a). None of these areas is covered by any other reader. Color 1, on the other hand, would generate a larger interference area  $A_1$  ( $A_{4a} + A_{4b} < A_1$ ), see Fig. 3(a). However,  $A_1$  is covered by reader  $R2$ . For this reason,  $R4$  chooses color 1.

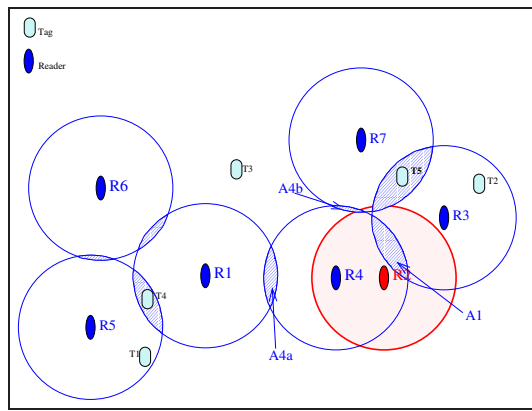
## V. ACORAS DESIGN EVALUATION

## A. Simulation setting

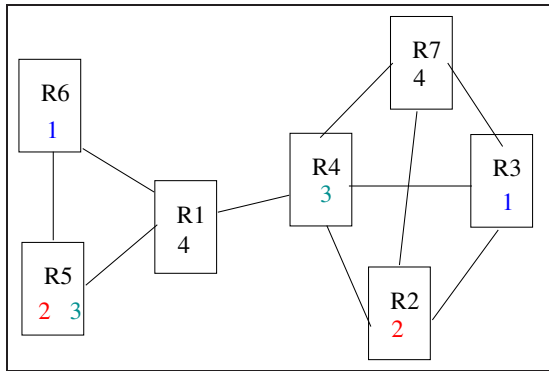
We evaluate the performance of our algorithm *ACoRAS* in terms of the number of successfully detected tags. For accurate results, we differentiate between failure to detect a tag due to reader collisions and due to lack of reader coverage.

We use a  $C$  simulator and deploy  $N$  RFID readers with the same transmission radius  $R = 100\text{m}$ . The readers are randomly scattered over a  $1000 \times 1000$  square. We also deploy at random a number ( $N_{tag}$ ) of mobile RFID tags and use the random way point mobility model; *i.e.* at bootstrap, every tag chooses a direction and a speed in  $[0..S_{max}]$ . In our simulation we set  $S_{max} = 6\text{km/h}$ . Note that we are aware that random way point mobility model is not realistic nevertheless, it states the worst case mobility scenario and thus we use it as a reference. We simulate the system for 30,000ms and set the duty cycle duration to  $T_{min} = 600\text{ms}$ , leading to 50 duty cycles. Results presented in this section are average values of more than 1000 simulation runs. Hardware requirements are such that  $t = 5\text{ms}$  [2].

We compare *ACoRAS* to DCS algorithm and to a variant of the anti-collision algorithm presented in [14] based on a Brelaz mechanism. In order to ensure that the coloring scheme provided in [14] fulfills hardware requirements and to fairly compare the three schemes, we run Algorithm 4 to remove



(a) Example of topology



(b) Graph correspondence

Fig. 3. *RemoveColors()*. Color 3 has to be removed from  $R4$  and  $R5$ . It is replaced by Color 1 on  $R4$ , which generates the  $A_1$  interference area with  $R1$  but  $A_1$  is covered by  $R2$  at slot 2. So, even with less color, every tag reading is ensured.

colors in all three methods. In addition, in order to evaluate the impact of assigning more than one color, we compare *ACoRAS* to a *one-coloring* scheme, which is similar to our adaptive coloring scheme. The only difference is that it assigns exactly one color per reader.

### B. Color assignment

1) *Color assignment illustration*: Fig. 4 illustrates the coloring assignment obtained simulating 20 readers. It shows that, in order to fulfill hardware requirements, *ACoRAS* has to reduce the overall number of colors from 5 to 3 (colors 1 and 4 were removed). Colors were removed from node 17 but were not replaced since node 17 holds other colors (3 and 0). To remove conflict in node 9, color 1 has been replaced by color 3. Although this coloring change leads to a conflict with reader 15, it is chosen because the interference area associated with color 3 is covered by reader 10 at slot 2 ensuring that tags in the interference area can still be detected and read in a duty cycle.

2) *Color assignment evaluation*: Fig. 5 shows the number of colors ( $|\Theta|$ ) needed to ensure no collision and the number of colors actually used to fulfill hardware requirements. Fig. 5(a) shows that the number of *colors needed* is independent of the number of tags (for a given number of readers, the curve is almost flat). However, as expected, the number of *colors needed* is closely related to the number of readers *i. e.* as the number of readers increases, more colors are needed to

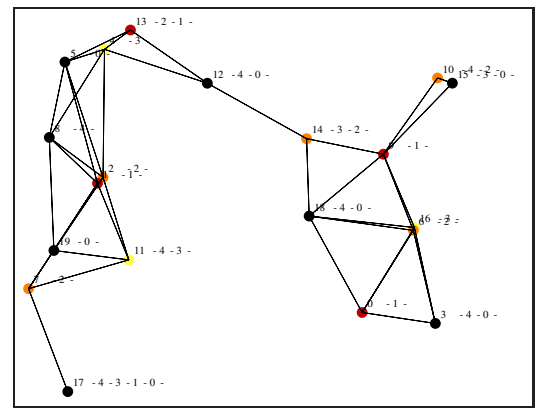
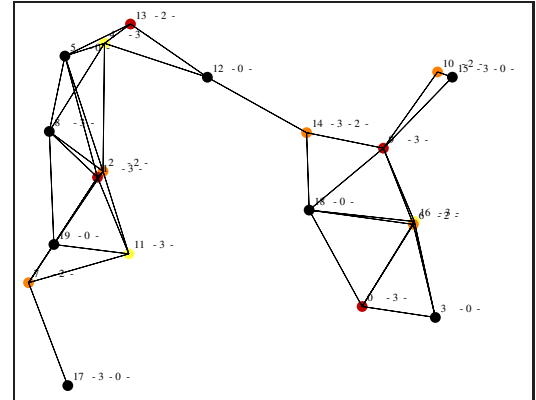
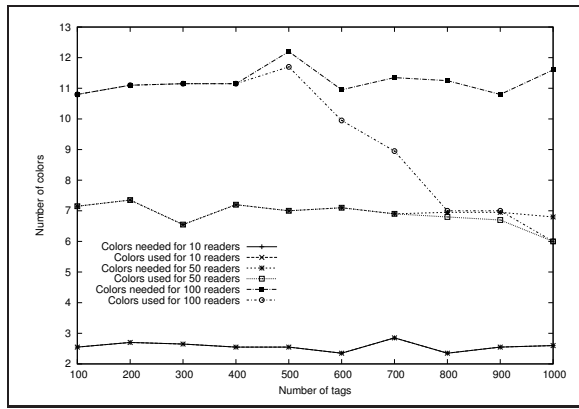
(a) *ACoRAS* before color refinement.(b) *ACoRAS* after color refinement.

Fig. 4. Coloring scheme. Nodes represent readers, the sequence of numbers displayed at each node represent the colors assigned to them. Links are interference edges.

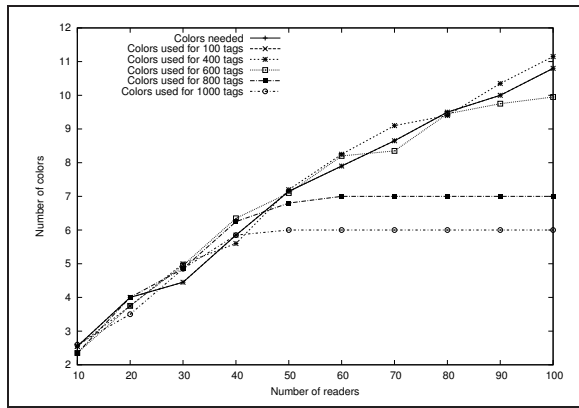
ensure no interference. For a low number of readers (10 in Fig.5(a)), since the number of needed colors is lower than the maximum number allowed ( $t \times N_{tag}$ ), all colors are used independent of the number of tags. When the number of tags increases, the maximum number of colors allowed decreases and thus *ACoRAS* has to optimize and remove colors from readers. For instance, Fig. 5(a) shows that for 100 readers, *ACoRAS* allows up to 11 colors, and color removal starts as soon as the number of tags reaches 500. It also shows that for a small number of tags (up to 300), all needed colors are used to fulfill hardware requirements. But when the number of tags increases (over 300), colors have to be removed for high densities of readers. Fig. 5(b) shows that the number of used colors increases as the number of tags increases up to an equilibrium point where color assignment stabilizes. When the number of tags is 1000, *ACoRAS* starts to decrease the number of colors needed to 6 when the number of readers reaches 40. This is due to the fact that the maximum number of colors allowed has been reached.

### C. Reading performance

In this section, we evaluate tag detection and reading performance of all algorithms by focusing on the proportion of unread tags. We consider only the proportion of tags that traversed a reader field during a duty cycle but were not detected.

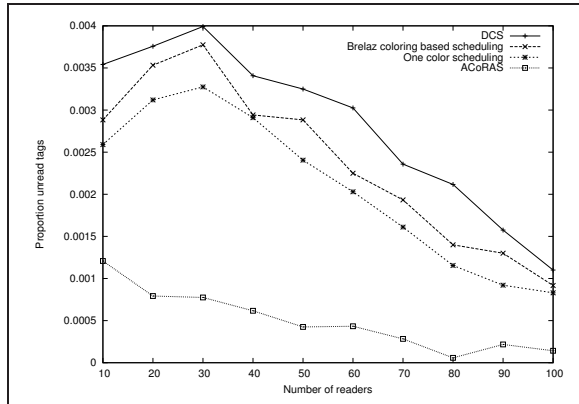


(a) scenario 1: Fixed number of tags but variable number of readers.

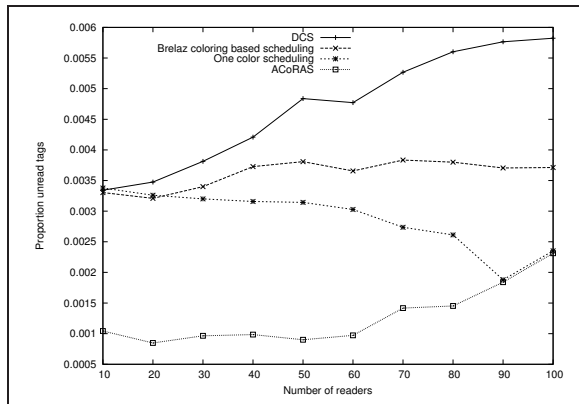


(b) scenario 2: Fixed number of readers but variable number of tags.

Fig. 5. Color assignment for different reader and mobile tag deployments.



(a) 200 tags



(b) 1000 tags

Fig. 6. Proportion of unread tags per duty cycle, for fixed number of tags and variable number of readers, for various scheduling schemes.

1) *Reader perspective*: Fig. 6 plots the proportion of unread tags per cycle as a function of the number of readers for the various scheduling methods. Results show that *ACoRAS* outperforms the various schemes, followed by *one-coloring scheduling* and *Brelaz-like schemes*. The worst performance is achieved by the *DCS* scheduling method. This is due to the fact that colors are assigned at random, which does not prevent two neighboring readers from having the same color. For a low number of tags ( $N_{tags} = 200$ , see Fig. 6(a)), all needed colors to prevent interference can be used since their number is less than  $N_{tags} \times t$ . As a result, the larger the number of readers, the more coverage we have hence the higher the number of tags detected tags.

For a high number of tags, the number of available colors decreases (see Fig. 6(b)). As a consequence, the number of collisions using *DCS* scheme increases with the number of readers since the probability to get two neighboring reader holding different colors, in this case, is small. *One-color based scheduling* and *Brelaz-based scheme* manage to reduce colors with a minimum conflict due to the *RemoveColor()* algorithm. As a result, the proportion of unread tags decreases with the number of readers since there may be tags in interference area that cannot be covered by another reader at another slot. However, *One-color based scheduling* achieves better results than *Brelaz-based scheme*. *ACoRAS* is the best performing protocol. It maintains the proportion of unread tags low when the number of readers is less than 60. Then after the number of unread tags increases. This is the point at which conflicts cannot be fully resolved. When the number of readers reaches 90, its performance is about the same as the ones of the *One-color based scheduling*. This means that at that point, *ACoRAS* has to assign one color per reader.

Fig. 7 provides a closer look at our scheduling mechanism for different number of tags. Results show that the proportion of unread tags globally decreases with the number of readers, especially for low number of tags (up to 600). For such tag densities, there is no reader collision, since all needed colors can be used. The unread tags are due to the fact that tags are spending less than  $T_{min}$  seconds in a given reader field. The proportion of unread tags thus decreases with the number of readers. For higher tag densities ( $> 600$ ), the proportion of unread tags stops decreasing to slightly increasing when reader density forces color removal (*i.e.* 40 readers for 800 tags, 50 readers for 1000 tags). Since the number of colors had to be reduced, tags will not be detected due to the presence of interference areas. Nevertheless, it is worth noting that the proportion of unread tags is not directly proportional to the size of the interference area since in our algorithm, a tag is read when it enters an area covered by a color different from the ones held by readers causing the interference. In general, the proportion of unread tags remains very low, less than 1%.

2) *Tag perspective*: Fig. 8 shows the proportion of unread tags per cycle as a function of the number of tags for the various schemes. Results show that *ACoRAS* outperforms all schemes. *DCS* scheme achieves the worst performances, followed by *Brelaz-based scheme* and *One-color based scheduling*. For low number of readers (Fig. 8(a)), the proportion of

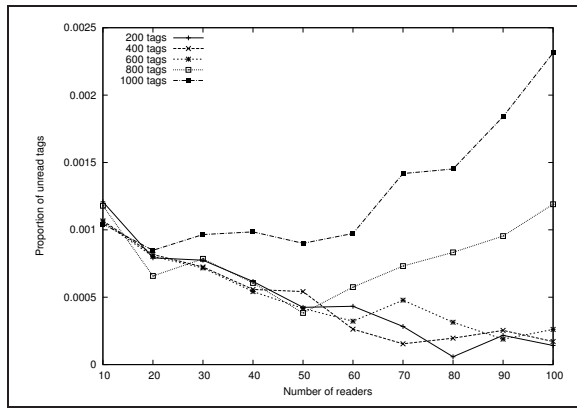
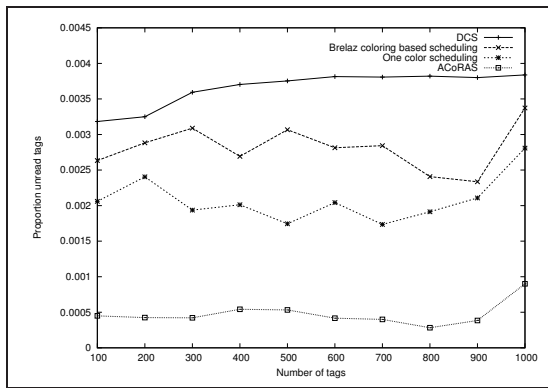
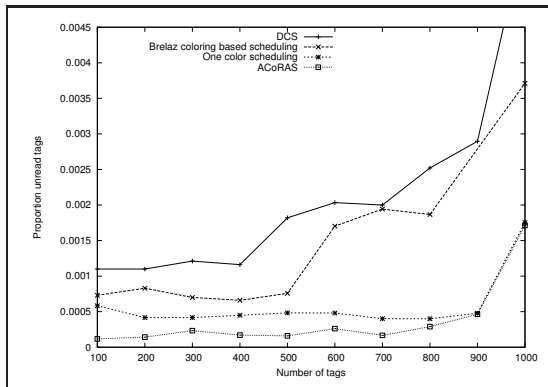


Fig. 7. Proportion of unread tags per duty cycle for various numbers of readers and fixed numbers of tags.

unread tags is globally constant for all schemes till reaching 900 tags. At this point, the available colors are not enough to resolve all conflicts. The same behavior can be observed for high number of readers (Fig. 8(b)) but at a lower number of tags threshold (800 tags). Indeed, the more readers, the more colors are needed to resolve conflicts.



(a) 50 readers



(b) 100 readers

Fig. 8. Proportion of unread tags per duty cycle, for variable number of tags and fixed number of readers, for various scheduling schemes.

## VI. CONCLUSION

In this paper, we design a novel scheduling algorithm *ACoRAS* for reader anti-collision in RFID system with mobile tags. Our scheduling is based on a color assignment which ensures that every tag spending at least  $T_{min}$  seconds in the field of a reader is detected and read. To reduce collisions

and increase throughput, readers are assigned colors (a reader can hold more than one color), as communication tokens (only readers holding a specific token during a given time slot can communicate with tags in their coverage area). This color assignment is further refined to comply with hardware requirements. This consists of removing excess colors assigned to readers to reduce the size of readers interference area and improve the performance of the algorithm. *ACoRAS* is scalable. Extensive simulation results show that *ACoRAS* outperforms several scheduling schemes. More than 99% of the mobile tags are successfully detected and read.

As an extension to this work, we are interested in studying RFID networks where both tags and readers are mobile. We are also considering RFID reader that can adapt their communication range. We expect the problem to become considerably more difficult with each additional feature and it is not clear how *ACoRAS* will behave.

## REFERENCES

- [1] S. Birari and S. Iyer. Pulse : A mac protocol for rfid networks. In *Proc. of Int. workshop on RFID and Ubiquitous Sensor Networks (USN)*, Japan, 2005.
- [2] M. Bolic, M. Latteux, and D. Simplot-Ryl. Framed aloha based anti-collision protocol for RFID tags. In *Proc. ACM Workshop on Convergence of RFID and Wireless Sensor Networks and their Applications (SenseID)*, Australia, 2007.
- [3] D. Engels and S. Sarma. The reader collision problem. In *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, 2002.
- [4] J.B. Eom and T.-J. Lee. RFID reader anti-collision algorithm using a server and mobile readers based on conflict-free multiple access. In *Proc. of IEEE Int. Performance, Computing, and Communications Conference (IPCCC)*, 2008.
- [5] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proc. IEEE Conf. on Computational Complexity (CCC)*, 1996.
- [6] K. Finkenzerler. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification.*, chapter 3, pages 29–61. John Wiley & Sons, 2003.
- [7] K.-I. Hwang, K.T. Kim, and D.-S. Eom. Dica: Distributed tag access with collision-avoidance among mobile RFID readers. In *Proc. of Embedded and Ubiquitous Computing Workshops (EUCW)*, Korea, 2006.
- [8] M. Ikeda, S. Kamei, and H. Kakugawa. A space-optimal self-stabilizing algorithm for the maximal independent set problem. In *Proc. of Int. Conf. on Parallel and Distributed Computing, Applications and Technologies (DCAT)*, Japan, 2002.
- [9] Texas Instrument. Tiris, 2003.
- [10] S.E. Sarma J. Ho, D.W. Engels. Hiq: a hierarchical q-learning algorithm to solve the reader collision problem. In *Proc. of Int. Symposium on Applications and the Internet Workshops (SAINT)*, USA, 2006.
- [11] S. Jain and S.R. Das. Collision avoidance in a dense RFID network. In *Proc. of ACM International Workshop on Wireless Network Testbeds, Experimentation and Characterization (WiNTECH)*, USA, 2006.
- [12] G. P. Joshi, K. M. A. Mamun, and S. W. Kim. A reader anti-collision mac protocol for dense reader RFID system. In *Proc. of Int. Conference on Communications and Mobile Computing (CMC)*, USA, 2009.
- [13] S. Piramuthu. Anticollision algorithm for RFID tags. In *Proc. of conference on Mobile and Pervasive computing (CoMPC)*, India, 2008.
- [14] J. Riihijärvi, M. Petrova, and P. Mähönen. Frequency allocation for w lans using graph colouring techniques. *Ad Hoc & Sensor Wireless Networks*, 3(2-3):121–139, 2007.
- [15] D. Simplot-Ryl, I. Stojmenovic, A. Micic, and A. Nayak. A hybrid randomized protocol for RFID tag identification. *Sensor Review*, 26(2):147–154, 2006.
- [16] EPCglobal Standard specification. EPC TM radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 mhz - 960 mhz version 1.2.0, 2007.
- [17] J. Waldrop, D. W. Engles, and S. E. Sarma. Colorwave : A mac for RFID reader networks. *Proc. of IEEE Conference on Wireless Communication and Networking (WCNC)*, 2003.
- [18] J. Waldrop, D. W. Engles, and S. E. Sarma. Colorwave: An anticollision algorithm for the reader collision problem. In *Proc. of IEEE International Conference on Communications (ICC)*, 2003.