Fast Identification of Mobile RFID Tags

David Benedetti, Gaia Maselli, Chiara Petrioli
Computer Science Department
University of Rome "La Sapienza"
Via Salaria 113, 00198, Rome, Italy
Email: {benedetti,maselli,petrioli}@di.uniroma1.it

***Abstract*—We consider the problem of efficient and fast identification of mobile tags in RFID networks. So far only a few works have addressed identification of mobile tags, and in very specific scenarios (i.e., tags placed on a moving conveyor). In this paper we address more general scenarios, involving tags that are free to move and may stay in the reader range for very short time (e.g., a few seconds), making their identification a real challenge for the reader. We propose a protocol, called PrIME (for Priority-based tag Identification in Mobile Environments), that is based on a probabilistic model and performs continuous reading cycles during which tags may enter and leave the system at any time. Through extensive ns2-based simulations we show that PrIME is very efficient, as it is able to identify 98-99% of mobile tags and to reduce the identification delay drastically with respect to other protocols.**

I. Introduction

Radio Frequency Identification (RFID) has recently emerged as a key technology for affordable and reliable item identification. An RFID system is composed of a reader device and a number of tags, communicating over a radio channel. The reader sends queries to tags that need to be identified. As tags are *passive* devices (e.g., without any power source), communication is arbitrated by the reader and happens through backscattering: the tag receives energy from the reader query and uses it to transmit a packet back to the reader.

One of the major challenges for RFID systems is efficient and fast tag identification [1]. The main issue here is that tag-to-reader replies may collide, resulting in no identification. The problem of reliable and fast identification is exacerbated by tags *mobility*. Indeed, the operational range of RFID systems is limited to a few meters, resulting in the need for a short identification time even in case of slow pedestrian mobility. In many scenarios tags may be in the reader range only for a few seconds, making their identification a real challenge for the reader. As RFID may be used to monitor movements of a large variety of entities, the development of solutions for mobile RFID systems is an important and challenging topic. Consider for instance the possible use of mobile RFID systems in airports and hospitals. In airports, passengers and baggage can be tagged so that their movements across gates can be tracked. Streams of people can walk through the reader while other passengers may temporarily stop under the reader range. In hospitals, tags may enable tracking of patients, medical staff, and equipment (as envisaged by healthcare projects such as CHIRON [2]). Equipment and bedridden patients are mainly static, while medical staff, autonomous patients, and visitors are free to move. Both scenarios involve a low (even zero) percentage of static tags remaining in the monitored area for quite a long time, and a high percentage of mobile tags traversing the reader range and disappearing in very short time. The identification of mobile tags becomes a crucial issue in such scenarios.

Current standards and existing communication protocols do not distinguish among tags with different level of mobility. The main consequence is a high percentage of "missed"(unidentified) mobile tags. To show the impact of this effect we performed experiments with a commercial reader [3] implementing the EPC global standard [4]. The reader was placed on a side of a corridor of our University department. Up to 20 students with tags were passing by the reader either in a single line or freely at normal pace. We measured the percentage of identified tags. In the case of a stream of people walking slowly in a single line, the reader was able to identify 90% of tags. Keeping the single line model, and increasing the number of tags walking across the reader range (we put two tags on each person to represent the case of people carrying a tagged object), the reader identified less than 60% of tags. The identification percentage dropped drastically below 50% when people were walking in group instead of a single line. The poor performance was mainly due to the many tags leaving the reader range before being identified.

Our experiments clearly show the need to revisit existing anti-collision protocols considering the heterogeneity of tags, especially in terms of the expected time they stay within the reader range. To the best of our knowledge, the field of anti-collision protocols for mobile tags is quite unexplored. So far several protocols have been proposed for tags identification in static environments, but only a few works have investigated dynamic scenarios. *Adaptive* protocols [5] consider scenarios in which the set of tags to be identified partially changes across consecutive readings. They thus introduce the idea of using information on the outcome of a reading cycle to optimize the following reading cycles querying tags in a predefined order. However they do not consider high mobility, performing poorly in such case. Other solutions [8] [9] consider tags moving on fixed paths. For example tags can be placed on a moving conveyor belt. The goal in this case is to find the minimum distance at which tags should be placed to be identified. In case the reader has no prior knowledge of or control on how tags are distributed on the conveyor belt, the addressed problem is how to dynamically adjust the conveyor

speed so that the reader is able to reliably identify tags.

Although these solutions represent a first step toward tags mobility support, they do not manage more general mobility models in which mobile entities cannot be controlled, freely move, and the sets of tags to identify over different identification cycles differ significantly.

Filling this gap is the objective of our paper. Specifically, we propose a new MAC protocol for single reader RFID systems, called PrIME (for Priority-based tag Identification in Mobile Environments), which provides fast and efficient identification of mobile RFID tags. PrIME addresses scenarios in which tags are free to move and do not follow any predefined mobility path. The innovative aspect in PrIME is the introduction of the concept of *priority* in the identification process that gives precedence to new tags entering the system with respect to static and already identified mobile tags. Specifically, the contributions of this paper are:

- A priority-based insertion model for tags, that defines how tags should participate in the reading process depending on their type (i.e., static, mobile).
- A cardinality estimation method for the reader to properly tune the identification process in a highly dynamic scenario.
- A ns2-based comparative performance evaluation of PrIME and relevant solutions proposed in the literature [1]. By means of extensive simulations in scenarios involving both static and mobile tags (in different percentages), we show that PrIME is always able to identify 98-99% of mobile tags. In addition PrIME reduces the tag identification delay length of over 100% with respect to other protocols.

The rest of the paper is organized as follows. Section II presents the state of the art on RFID identification protocols. Section III describes the PrIME protocol. Section IV discusses the results of a simulative performance evaluation of PrIME, ABS, FSA, and BSTSA. Section V concludes the paper.

II. Related work

A significant amount of research has been conducted on anti-collision protocols for RFID systems with the goal of speeding up tag identification in *static* environments. Targeted applications include automatic object inventory, tracking, and management. State of the art solutions in this field can be classified into two main classes: *tree*-based and *aloha*-based protocols. Tree-based anti-collision protocols build on serial tree algorithms (also known as walking tree algorithms) [10] [11]. Tree based protocols iteratively query subsets of tags which match a given property until all tags are identified. These protocols are called tree-based since the identification process can be represented as a tree where the root is the set of tags to be identified, intermediate nodes represent groups of colliding tags answering the same reader request, and the leaves correspond to single-tag responses. The main representatives of tree-based protocols differ in the way tags are queried: based on a counter stored in the tags in Binary Splitting (BS) [5], and on the binary structure of tag IDs in Query Tree (QT) [12] solutions.

The second group of anti-collision protocols assumes a time-slotted channel and is based on the Aloha scheme. The start of each slot is signaled by the reader with a short message. Slots are grouped into frames. The frame size is set and communicated by the reader at the beginning of each frame. Upon receiving the reader query, each tag randomly and uniformly selects one slot in the frame to transmit its ID. At the end of the frame, tags that are successfully identified become silent, while tags that generated collisions keep trying in the following frames. This process is iterated until all tags are identified. The main problem in Aloha protocols is that the number of tags to be identified is not known by the reader and must be estimated. Optimal frame sizing and allocation of tags to frames are critical design choices which affect the protocols performance. Tree Slotted Aloha (TSA) [13], reduces collisions by organizing frames in a tree: instead of issuing a new frame for all tags that collided in the previous frame, several new frames are issued, one for each colliding slot in the previous frame. Only the tags that collided in the same slot participate in the same new frame. This solution reduces the size of colliding groups, making the protocol more efficient. Chebyshev's inequality is used to estimate the tags that participated in a frame. This number is then used to optimally set the size of the next frame [1].

The problem of properly setting the initial frame size has been addressed by the Binary Splitting Tree Slotted Aloha (BSTSA) protocol [1]. In BSTSA, an initial splitting phase is performed to group tags into sets whose size can be efficiently estimated. Then TSA is applied to each group. The work in [1] shows that idle slots are shorter than colliding slots. This observation is then used to optimize the frame size for fast tag identification. BSTSA is shown to outperform slotted Aloha protocols, QT and BS. However, BSTSA (as all other anti-collision protocols for static environment) does not cope efficiently with highly dynamic scenarios typical of mobile RFID (see results in Section IV).

Some of these protocols for static environments have been modified to fit scenarios in which tags need to be continuously identified, and the set of tags in the environment may partially change. This is typical of applications that want to identify which objects have moved into and out of the monitored environment (e.g., stockroom locations) across successive readings. The proposed protocols, called *adaptive*, introduce the idea of using information on the outcome of a reading cycle to optimize the successive reading by querying tags in a predefined order. In particular, the work in [5] describes two adaptive tree protocols: Adaptive Binary Splitting (ABS) and Adaptive Query Splitting (AQS). After the first reading (performed respectively through BS or QT), both protocols try to avoid idle and collision slots by querying directly previously identified tags. Specifically, ABS assigns unique slots to tags, while AQS uses a temporary queue in which all prefix strings that led to identification are put. Adaptive protocols represent a first step toward dynamic scenarios but do not address high

mobility.

Only recently, the application of RFID in *mobile* scenarios has been considered. The solutions in [8], [9] consider tags moving on fixed paths, for instance on a moving conveyor. The goal in this case is to find the minimum distance at which tags should be placed to be identified. In case the reader has no prior knowledge of how tags are distributed on the conveyor, the reader tries to dynamically adjust the conveyor speed so that it is able to reliably identify tags. These solutions represent a step towards mobile scenarios, but do not manage more general mobile environments in which tags move freely and a timely identification is required.

Finally, the work in [7] proposes the use of multiple readers to address more general mobility scenarios. The focus of the work is mainly on reader-reader collision, as they try to synchronize readers queries to avoid interference. However, in this paper we consider the single-reader scenario in order to keep the system cost low.

III. The PrIME Protocol

A. Basics

The objective of PrIME is to reliably identify all tags that pass through the system. To this end, PrIME has to address new significant challenges introduced by tag mobility. First, mobile tags do not remain long under the reader range and hence require a timely identification. Second, mobility causes high variability in the tags population, as both the amount and the identity of tags change continuously, making tags cardinality estimation a challenging issue. To address these two important problems PrIME introduces innovative concepts such as: priority identification for mobile unknown tags and temporal distributed identification of static and mobile known tags. Hereafter we explain how these two important features are realized in the PrIME protocol.

Tags have a *type* **T** and a *priority* **P**. Type can be *static* ($T = 0$) or *mobile* ($T = 1$) and is set at the moment of assigning a tag to an entity (e.g., in the hospital environment, people will be designated as mobile while equipment will be set as static). Priority expresses how fast a tag needs to be identified and depends on the tag type: mobile tags may have *high* or *medium* priority, while static tags have *low* priority. As mobile tags are supposed to remain for a short period of time in the reader range, they should be identified as soon as possible. When a mobile tag enters the system it is unknown and sets its priority to high. When it gets identified it decreases its priority to *medium*, so that it does not compete with (and consequently delay the identification of) unidentified tags. It can however still be identified which is important to implement tracking services. The higher the priority, the sooner the tag has to be identified. Tags communicate their type and priority to the reader together with their ID. When receiving this couple of bits, the reader can distinguish among different classes of tags:

- $\langle$ **T = 0**, **P = 0** $\rangle$: *static tags* (S)
- $\langle$ **T = 1**, **P = 0** $\rangle$: *mobile tags with medium priority* (M_m)
- $\langle$ **T = 1**, **P = 1** $\rangle$: *mobile tags with high priority* (M_h)

As described later, this information is useful to the reader to estimate the number of tags entering and remaining in the system.

Given the need for continuous reading, PrIME performs consecutive *reading cycles* (in the paper we also use the notation *cycles*). Each reading cycle is composed of multiple *group identifications*, each devoted to identify a subset of the tags that are within the reader area during the cycle. Reading cycles are defined only for internal protocol operation and are completely invisible to the user that instead sees tags reading as a continuos process.

Reading cycles follow a process similar to BSTSA [1]. Typically, there is an initial *grouping* phase, that divides tags into sets of computable size, and defines the number of groups that will be identified in a reading cycle. Then there is an *identification* phase that actually reads tags and is performed by applying the TSA protocol to each group [1]. As mobile tags can enter the system at any time, PrIME defines the way mobile tags can join the group identification phase. The main idea in PrIME is to allow mobile tags to join with high probability the groups that are going to be identified next, and with low probability the groups that have to wait longer before being identified. The main innovation in PrIME relies in the way mobile tags enter a reading cycle, are served based on their priority, and are estimated by the reader to properly tune the protocol. Hereafter we first present the main protocol phases and then explain the joining policies for mobile tags, specifying the protocol details from the tag and the reader sides.

B. Phase I: Grouping

At the beginning, PrIME does not know anything about the tags it has to identify (e.g., number of tags, their type) and hence performs a preliminary grouping phase, whose participants depend on the type of the tags that are in the system at that moment. The reader sends a splitting query only to static tags. If they are present in the system, they will reply to the query and will execute the grouping phase.[1] A reading cycle starting with a grouping phase that is executed only by static tags is called *static split*. In case there are no static tags, then the reader sends another query, that is addressed to the mobile tags that are present. In this case, grouping starts a reading cycle that is called *mobile split*. In both types of cycles, mobile tags entering the system during grouping, wait until the end of this phase. At that point they will join a group and will be identified according to the joining policy described later.

Grouping is performed by applying a binary splitting process, that recursively divides tags into two subgroups of decreasing size until the first single-tag group is obtained (i.e., the first identification occur). As shown in Fig. 1, the first reader query interrogates the initial set of tags, whose size n is usually not known a priori by the reader. As tags collide, they

[1]Mobile tags do not participate to this phase.

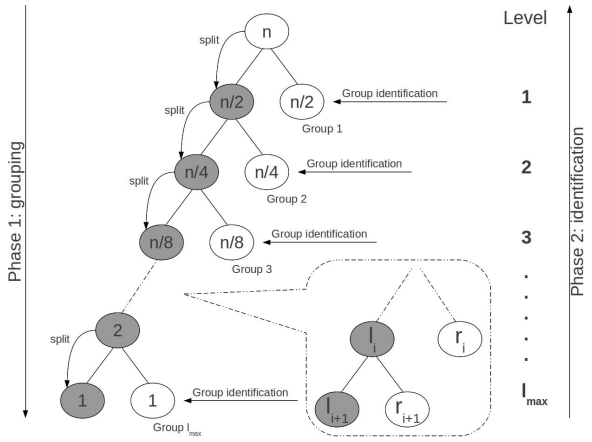

Fig. 1. Example of PrIME grouping and identification.

generate a random binary value, so that they are split into two subsets: those who generated zero and those who generated one. Only the set of tags who generated zero replies to the next query. This process is recursively applied until the answering group is composed of a single tag, that is identified. It is worth noting that each query carries a feedback (e.g., one bit) about the previous query's outcome, meaning whether there has been a collision (1) or not (0). In terms of tree representation, the first tag identification corresponds to visiting the leftmost leaf of the tree (see Fig. 1). At the upper levels, tags are split into groups (i.e., right siblings on the tree) of increasing size, as we go up on the tree. In particular, at each level of the tree, each node contains approximately half of the tags of its parent node.

C. Phase II: Group joining and Identification

In this phase, each group of tags, positioned at a different level of the tree, will be identified by applying the TSA protocol [13], starting from the bottom of the tree.

In the first frame TSA queries all the tags in the group, then a new set of child frames is allocated, each devoted to solving the collisions which have occurred in a given slot of the first frame. Only the tags which transmitted in that slot participate in the corresponding frame. The approach is repeated: if collisions occur in one of the frames allocated to solve collisions (say, frame f), new frames are allocated to solve such collisions (one for each collision slot in frame f). In order to properly set the size of each child frame, the reader first estimates the number of colliding tags in the ancestor frame through the Chebyshev's formula and then divides this value for the number of collisions detected [13]. The resulting value is then overestimated to get the optimal frame size [1].

At the end of a group identification, the number of identified tags is used to estimate the number of tags that have to be identified at the "upper" level. As grouping is performed through binary splitting, at each level $i+1$ we expect the right sibling r_{i+1} to contain a group of tags whose cardinality is similar to the number of tags found on its left sibling l_{i+1} (see Fig.1). Going up one layer (say to layer i-th), the sum of the number of tags belonging to the two sibling child nodes r_{i+1} and l_{i+1} gives the number of tags we expect to find in the right sibling of the parent node (i.e. r_i). This information is used to apply TSA with a proper initial frame size[2] to identify the tags in r_i. This process starts with TSA identifying the tags in the lowest right leaf of the tree. The number of tags found in this group is summed to the tag identified in the lowest left leaf, resulting in the number of tags belonging to the left sibling of the upper level. This gives the number of tags that are expected to be found in the right sibling of the same level, and so on.

Mobile tags entering the system during the grouping or the identification phases need an insertion policy to join a group. They are high priority tags (i.e., M_h) and hence they should select with high probability the next queried group so that they are identified soon. Basically, at the beginning of each group identification, M_h tags that are not yet identified, select and join the next group with a probability that is calculated as follows. Let us suppose the reader is currently querying the group at the i-th level of the tree ($1 \leq i \leq l_{max}$), then the tags M_h that have to be identified will join this group (at level i) with probability

$$P_i\left(L^{M_h} = i, l_{max}\right) = \frac{1}{2^{l_{max}-i+1}} + \epsilon + P_{acc}\left(i, l_{max}\right) \quad (1)$$

where $(\frac{1}{2^{l_{max}-i+1}})$ represents the probability of choosing level i based on a distribution that is opposite to the one used to place static tags on the tree (in the splitting operation each tag has probability $1/2^i$ to select the group at the i-th level). As tags repeat this selection at each level until they succeed, it may be possible that they do not choose any level in a reading cycle. For this reason, it has been introduced the ϵ term that strengthens the distribution. It represents the *residual probability* needed to ensure that each tag chooses one level of the current tree and is defined in eq. 2.

$$\epsilon = \frac{1 - \sum_{j=1}^{l_{max}} \left(\frac{1}{2^{l_{max}-j+1}}\right)}{l_{max}} \quad (2)$$

The term $P_{acc}\left(i, l_{max}\right)$ in eq. 1 further strengthens the distribution in order to minimize the waiting time of M_h tags. This *accumulated probability* $P_{acc}\left(i, l_{max}\right)$ is defined as the sum of the joining probabilities of the groups which have been already queried (i.e. before the i-th group), considering a tree of height l and its generic group i, $1 \leq i \leq l$. Equation 3 defines the term $P_{acc}\left(i, l_{max}\right)$ that increases the probability of choosing level i based on the groups already identified: the higher the current level, the bigger the probability.

$$P_{acc}\left(i, l\right) = \sum_{k=i+1}^{l} \frac{1}{2^{l-k+1}} + \epsilon \quad (3)$$

As tags may fail to join group i, they will try again when the next group will be queried (i.e. $(i-1)$-th). This probabilistic

[2]We adopt the optimal frame size as defined in [1].

selection (eq. 1) is repeated until tags either choose a group to join or leave the system. Somehow, the term P_{acc} introduces the concept of *aging*, as it increases the probability of selecting the next group for tags that failed in the previous group selection.

Mobile tags already identified and still present in the system (i.e., M_m) instead make the selection once for all at the beginning of the group identification phase. They select a group to join based on the following probability:

$$P\left(L^{M_m} = j, l_{max}\right) = \frac{1}{2^{l_{max}-j+1}} + \epsilon \tag{4}$$

With respect to M_h tags (eq. 1), medium priority tags do not need the P_{acc} term because they do not urge to be identified as new tags. The *residual probability* ϵ makes the distribution for L^{M_m} sums to 1, independetly of l_{max}.

$$\begin{aligned}\sum_{j=1}^{l_{max}} P\left(L^{M_m} = j, l_{max}\right) &= \sum_{j=1}^{l_{max}} \left(\frac{1}{2^{l_{max}-j+1}} + \epsilon\right) \\ &= \sum_{j=1}^{l_{max}} \left(\frac{1}{2^{l_{max}-j+1}}\right) + l_{max} \cdot \epsilon \\ &= 1 \end{aligned} \tag{5}$$

Similarly, it can be proven that the probability for M_h tags to join a level increases as the identification process goes up the tree, until it sums to 1 at the top level. This means that everytime a tag fails to join a group it has higher probability to join the next one. Thus, in the worst case the tag joins the top level group of the tree, regardless of l_{max}. This means that a new tag will be identified within the reading cycle during which it enters the system.

D. Adaptive operation

Once the first reading cycle ends, PrIME behavior depends on the type of splitting previously performed: static or mobile. In case of a static split, the PrIME protocol enters the *steady state*, performing continuous reading cycles by exploiting the outcome of the previous ones. As static tags remain in the system for long time, they do not change across multiple reading cycles, and the performed grouping operation remains valid for the following reading cycles. Hence from the second reading cycle on, they are directly identified through an ABS process [5]: they are assigned to specific slots and a frame dedicated only to static tags is performed. As ABS assigns specific slots to tags, the identification of static tags is very fast and does not involve any idle or collision slots.

Mobile tags are instead identified through the TSA protocol. There can be two types of mobile tags. Those that just entered the system and those that arrived during the previous reading cycles and are still present in the system. The former are high priority tags (M_h) because they have to be identified for the first time, while the latter can be identified again, but with medium priority (M_m). The M_h tags join reading groups as specified in eq. 1[3]. While M_h tags perform group selection at each level of the tree until they succeed (based on the probability defined in eq. 1), M_m tags make their selection once for all at the beginning of the reading cycle, according to the distribution defined in eq. 4.

In case of a mobile split in the first reading cycle, PrIME checks the presence of possible new static tags. If there is any, then PrIME performs a static split and after the execution of the reading cycle enters the steady state described above. In case new static tags enter during the steady state, they join the identification process by selecting a group i based on the probability $1/2^i$. The new static tags will be managed by the ABS protocol that is designed to handle sporadic changes. If there are only mobile tags, they perform a new mobile split. In this case the adaptive aspect of the protocol lies in the estimation process of the tags that may enter the system. The new tag population will be estimated by considering the previous history.

E. Cardinality estimation

The reader is responsible for estimating the cardinality of tag groups and properly size TSA frames. To estimate tag groups the reader has to consider the tags already in the system and that have not yet left (M_m) plus the tags that just entered the system (M_h). To assess the total number of tags currently in the system, the reader estimates the *mobility model* of tags, and the *number* of new and already present tags.

1) Mobility model estimation: The reader estimates the mobility model of tags by calculating the *entrance frequency* of mobile tags M_h and the *permanence factor* of M_m tags.

As the addressed scenarios imply high mobility, with bursts of tags that might enter the system at any time, we estimate the mobility model through an *exponential mean*, of the form $\lambda_{i+1} = \alpha_i \cdot \Delta_i + (1-\alpha_i) \cdot \lambda_i$, where Δ_i represents the outcome obtained at time i and λ_i the history observed up to time i. The term α_i weights the history and its value changes according to the estimation error (details on the α_i values will be given later in this paper).

The *entrance frequency* is estimated by considering the number of mobile tags that entered the system per unit of time in the previous group identification. Let us consider the end of the j-th group identification in the i-th reading cycle, then we estimate the entrance frequency for $(j+1)$-th group $\lambda_{j+1}^{M_h}$ of mobile tags M_h, as in eq. 6.

$$\lambda_{j+1}^{M_h} = \alpha_j \cdot \left(\frac{n_j(M_h)}{T_{j-1}} + d_j\right) + (1-\alpha_j) \cdot \lambda_j^{M_h} \tag{6}$$

where $1 \leq j \leq l_{max,i}$. T_{j-1} represents the time taken by the previous group identification. Hence, the term $n_j(M_h)/T_{j-1}$ expresses the number of tags that entered the system per unit of time, computed in the iteration just ended. The term d_j is a drift factor used to correct the estimation error and it is detailed later.

[3]Such reading groups are those created in the grouping phase

The entrance frequency is updated at the end of every group identification in a reading cycle.

The *permanence factor*, instead, is calculated at the end of each reading cycle and estimates the average number of mobile tags M_m remaining in the system and joining the lowest group of the splitting tree. This value will then be used to estimate the M_m tags that join the upper level groups. Let us consider the end of reading cycle i. Then the permanence factor for the lowest group of reading cycle $i+1$ is estimated by considering the M_m tags identified in the bottom level of reading cycle i and the history observed up to reading cycle i

$$\lambda_{i+1}^{M_m} = \alpha_i \cdot (n_{l_{max}}(M_m) + d_i) + (1 - \alpha_i) \cdot \lambda_i^{M_m} \quad (7)$$

where $n_{i,l_{max}}(M_m)$ is the number of M_m tags identified in the lowest group during i-th session.

In case of a mobile split reading cycle, this factor is not estimated. In this case the number of M_m tags at each level is provided by the BSTSA estimation process.

Now we explain how the weight factor α and the drift factor d are set and updated. First, we calculate the deviation factor $\hat{f}_i = e_i - v_i$, defined as the difference between the estimated value e_i and the observed value v_i at time i. For instance, in eq. 7 e_i represents the estimated value $\lambda_i^{M_m}$ and v_i stands for the observed value $n_{l_{max}}(M_m)$. Then, we obtain the standard deviation σ_i about all values observed until the i-th update. We use $\hat{f}_i$ and σ_i to keep the estimation error as small as possible ($\hat{f}_i \to 0$). We now discuss the rules to update α_i and d_i.

i) $\sigma_i < \hat{f}_i$. When this condition occurs, we know the estimation has gone beyond the standard deviation (i.e. excessive overestimation), so we want the current history do not affect the estimation more. We set $\alpha_i = 0.4$ and $d_i = -1$. Then, if the condition persists α_i remains unchanged but d_i decreases its value with a *slow-start* strategy: first exponentially, then (i.e. when an exponential decrease exceeds σ_i) linearly.

ii) $\sigma_i/2 < \hat{f}_i \leq \sigma_i$. If this condition occurs, we know there has been an acceptable overestimation. Thus, we set $\alpha_i = 0.9$ and $d_i = -1$, with slow-start decrease if this condition holds again. In this way we want to hold the estimation over real values, but with the intention of lowering its value cautiously.

iii) $0 \leq \hat{f}_i \leq \sigma_i/2$. If the overestimation is under the guard limit then, in order to hold the behaviour of $\hat{f}_i$ as much as possible, we set $\alpha_i = 0.9$ and $d_i = \sigma_i/2$. The value of d_i is intended only to avoid future underestimations.

iv) $-\sigma_i/2 < \hat{f}_i < 0$. This condition suggests the estimation is close to the average value, although an underestimation has occurred. As before, we set $\alpha_i = 0.9$ and $d_i = \sigma_i/2$, so as to keep the error close or greater than zero.

v) $\hat{f}_i \leq -\sigma_i/2$. If the underestimation is over the negative guard limit ($-\sigma_i/2$), we weight the current history a lot ($\alpha_i = 0.9$) and set the drift to $\sigma_i/2$. In this way a single update should be adequate to approach the error close to zero. If this condition lies again, we set the drift directly to σ_i, so as to decrease the updates number before the error tends to zero.

The update rules for α_i and d_i change according to the distance between $\hat{f}_i$ and some guard limits (i.e. the range $[-\sigma_i/2, \sigma_i]$). It is worth noting that lower limit $-\sigma_i/2$ is half of the upper one. This is because we want to be valid $\hat{f}_i \to v_i + \delta$ ($\delta \geq 0$) as much as possible, preferring an overestimation rather than more collisions. Finally, we note there is the guard limit $\sigma_i/2$ in the middle, used to avoid excessive overestimations ($\delta >> 0$).

2) Group size estimation: The entrance frequency of M_h and the permanence factor of M_m tags allow to estimate the number of tags placed in each group the reader has to identify. The estimated values are used by PrIME to properly tune the initial frame size for the TSA identification process applied to each level of the tree, starting from the bottom level up to the top one.

To estimate group cardinalities we distinguish between the lowest level group (i.e., at l_{max} level) and the groups in the rest of the tree (from level 1 to $l_{max} - 1$). The lowest level group is the first to be queried in each reading cycle. Its cardinality depends on the type of the reading cycle being executed: static split, steady state, and mobile split. In the static split, there is only one tag in the lowest level group as there is no history about mobile tags (as static split is executed at the protocol beginning).

In a steady state reading cycle, we need to estimate how many tags entered during the last group identification of the previous reading cycle and how many have remained in the system since it. In this type of reading cycle only mobile tags participate in TSA protocol. Thus, the estimated number $\hat{n}_{l_{max}}$ of tags that are present in the bottom level group is calculated as in eq. 8.

$$\hat{n}_{l_{max}} = 1 + \lambda_i^{M_m} + \left\lceil \lambda_{i-1,1_{i-1}}^{M_h} \cdot P_{l_{max}}\left(L^{M_h} = l_{max}, l_{max}\right) \cdot T_{i-1,1_{i-1}} \right\rceil \quad (8)$$

where the term 1 avoids to perform empty frames in the case that no mobile tags are present. The term $\lambda_i^{M_m}$ is the permanence factor defined in eq. 7 and the last term represents the M_h tags that were placed in current group, out of the M_h tags that entered the system during the previous group identification (corresponding also to the last identified group in the previous reading cycle).

In a mobile split reading cycle, medium priority tags participate in the splitting process and hence only high priority tags that may have entered the system have to be estimated. Eq. 9 estimates this number.

$$\hat{n}_{l_{max}} = 1 + \left\lceil \lambda_{i-1,1_{i-1}}^{M_h} \cdot P_{l_{max}}\left(L^{M_h} = l_{max}, l_{max}\right) \cdot T_{BS} \right\rceil \quad (9)$$

The term 1 represents the presence of at least one medium priority tag. The second term represents the M_h tags that were placed in current group, out of the M_h tags that entered the system during the splitting operation. The entrance frequency for *high-priority* tags is weighted by the execution time of the splitting operation just ended.

Going up on the tree, the size of tag groups is estimated exploiting the outcome of the previous group identification in the same reading cycle. The number of M_m tags present in a group is estimated by considering the probability distribution used by these tags to select a group. Specifically, the number of M_m tags at level l, $1 \leq l < l_{max}$, is calculated as the half of the tags M_m identified at the level below (i.e., $l+1$). Eq. 10 estimates this quantity.

$$\hat{n}_l(M_m) = \frac{n_{l+1}(M_m)}{2} \tag{10}$$

The number of M_h tags at level l, $1 \leq l < l_{max}$, is estimated as by eq. 11.

$$\hat{n}_l(M_h) = \lambda_l^{M_h} \cdot P_l\left(L^{M_h} = l, l_{max}\right) \cdot T_{l+1} \tag{11}$$

In equation (11) the entrance frequency $\lambda_{l+1}^{M_h}$ is weigthed by the probability that a tag joins the very next group $P_l\left(L^{M_h} = l, l_{max}\right)$ and the execution time of the iteration just ended (T_{l+1}).

The estimated values $\hat{n}_l(M_m)$ and $\hat{n}_l(M_h)$ are then used to calculate the expected number of mobile tags in each group, starting from the next to last (i.e., $l_{max}-1$) up to the first (i.e., 1). These estimations depend on the type of the reading cycle and are defined as follows.

For a static split reading cycle, the expected number of tags in the group at level l, $1 \leq l < l_{max}$, is given by eq. 12.

$$\hat{n}_l = n_{[l+1,l_{max}]}(S) + \hat{n}_{l+1}(M_m) + \hat{n}_{l+1}(M_h) \tag{12}$$

where $n_{[l+1,l_{max}]}(S)$ is the number of static tags identified until level $l+1$. The values $\hat{n}_{l+1}(M_m)$ and $\hat{n}_{l+1}(M_h)$ are estimated as in eq. 10 and 11.

For a steady state reading cycle, the expected number of tags in the group at level l, $1 \leq l < l_{max}$, is given by eq. 13.

$$\hat{n}_l = 1 + \hat{n}_l(M_m) + \hat{n}_l(M_h) \tag{13}$$

where the term 1 again avoids that PrIME executes empty frames.

Finally, for a mobile split reading cycle, the expected number of tags in the group at level l, $1 \leq l < l_{max}$, is given by eq.14

$$\hat{n}_l = n_{[l+1,l_{max}]}(M_m) + \hat{n}_l(M_h) \tag{14}$$

where $n_{[l+1,l_{max}]}(M_m)$ is the number of M_m tags identified up to level l.

Note that all the complexity of the protocol lies on the reader side. The reader decides the type of phase, specifies the parameters of the protocol (estimating mobility model, cardinality, identifying the proper weight to give to past history). Indeed the reader is a powerful device. Tags simply have to compute a different probability to join the next group depending on their type and on the level of the tree.

IV. Performance evaluation

In this section we present the results of an extensive ns2 based performance evaluation, comparing PrIME with the main representative anti-collision protocols in the literature. Specifically, we selected the BSTSA and FSA protocols as Aloha protocols and the ABS protocol as tree-based protocol. BSTSA is the most recent and best performing Aloha protocol available in literature for static scenarios while FSA has been included to have a comparison also with the most popular protocol for RFID systems. To make it work in a mobile scenario, BSTSA has been implemented so that is allows to participate in a reading cycle only those tags that receive the first query of the splitting process (BS query). The other tags that enter the system during the execution of the reading cycle will wait until a new cycle begins. In this way BSTSA is able to properly estimate the tags that are participating in a reading cycle, and can identify them in very short time. The entrance of new tags during the execution of a reading cycle would alter the estimation process. FSA behaves in a similar way. Only those tags receiving the first message of the first frame of the new reading cycle will participate to the reading cycle. The others will wait until a new cycle begins. An FSA reading cycle is composed of multiple frames (properly sized, based on the Chebyshev's estimation). New frames are issued until no tags answer (empty frame). At this point the reading cycle ends. In order to maximize the efficiency of the protocols and make a fair comparison with PrIME, optimal frame tuning [1] is applied to both BSTSA and FSA.

Being adaptive, the ABS protocol is the closest competitor of PrIME in dynamic scenarios. Even for the ABS, tags entering the system during the execution of a reading cycle wait for the begin of a new reading cycle to participate in the protocol. In this way the adaptive operation of ABS is not neutralized by the entrance of new tags.

A. Simulation setup

To analyze the protocol performance we focus on the following metrics: (i) *number of identified tags* measured as the average number of identified tags per reading cycle (this value includes identification of M_h and M_m tags); (ii) *ratio of high priority identified tags* measured as the average number of identified M_h tags over the number of tags M_h present in the system, per reading cycle; (iii) *identification latency* defined as the mean time in seconds needed to identify each tag; and (iv) *time system efficiency* defined as the ratio between the identification time and the total time used by the protocol [1]. We consider an RFID system with a single reader that has a transmission range of 2m. We generate results varying the number of tags between $n = 10, ..., 500$ and the percentage of mobile tags, from 20% to 100%. For sake of space we show only the results for 500 tags (results for other number of tags did not show any significant difference from the 500 tag scenario). The channel data rate is 40 Kbps and reader-tag communication occurs at a frequency 866 Mhz as specified by the EPCglobal standard [4]. Tag IDs are 96 bits long, which is the most commonly used ID length [4], and they are uniformly

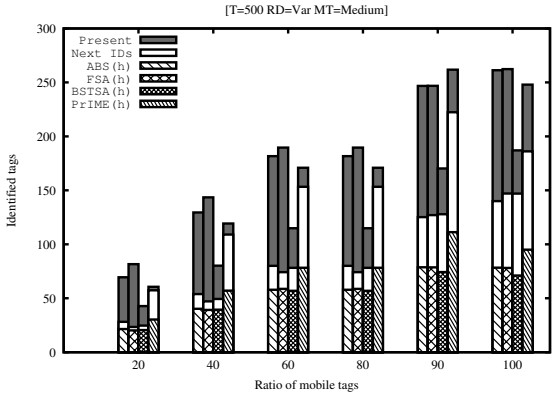

Fig. 2. Number of identified tags, including first and subsequent identifications (high and medium priority tags).

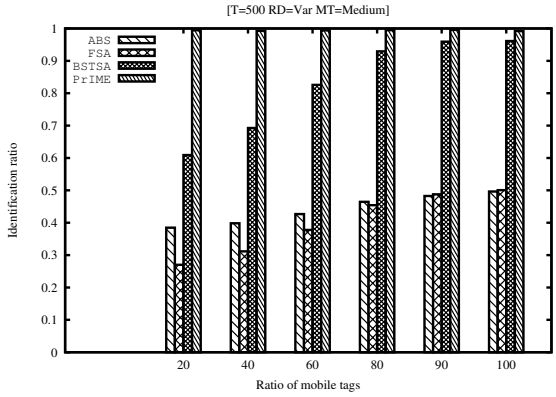

Fig. 3. Ratio of mobile tags identified for the first time in each reading cycle.

distributed. Results have been obtained by averaging over 100 reading cycles.

We assume a pedestrian mobility model, in which each tag has a speed picked randomly in the range $[1.29,\ 1.49]m/s$, whose extremes represent respectively the average speed of an elderly and the average speed of a young person. However, tag speed is not constant (it can slightly slown down). Thus, assuming for the antenna a 2 meters communication range, the crossing time of the reader range for mobile tags may vary in the interval $[1.40, 1.73]s$. Results are shown only for mobile tags, as static tags are always all identified by all protocols.

B. Results

Results in Fig. 2 show the average number of identified tags per reading cycle when varying the percentage of mobile tags. The bars in the figure give an insight on the type of tags (high and medium priority) identified by each protocol, displaying with different patterns (depending on the protocol) the number of tags identified for the first time (lower part of the bar), the number of tags already identified in the previous reading cycle and identified again in the current one (white mid section of the bar), and finally the total number of tags present in the system in the current reading cycle (whole bar, with the grey upper section of the bar representing unidentified tags in the current cycle). PrIME identifies the largest number of both high (M_h) and medium (M_m) priority tags (shown by the sum of the patterned and white bars), independently of the percentage of mobile tags in the system. All the other protocols (i.e., ABS, FSA, BSTSA) identify between 45% and 50% of tags less than PrIME in case of 20%-90% of mobile tags. This gap decreases to about 23% when all tags are mobile.

To show the capability to identify new tags (M_h), Figure 3 reports the percentage of mobile tags that are identified for the first time by the different protocols. PrIME identifies always 99% of high priority tags independently of the percentage of mobile tags in the system, while ABS and FSA identify always less than 50% of new tags. BSTSA shows good results for high percentage of mobile tags, with an identification ratio that increases from 60% to 96% when the percentage of

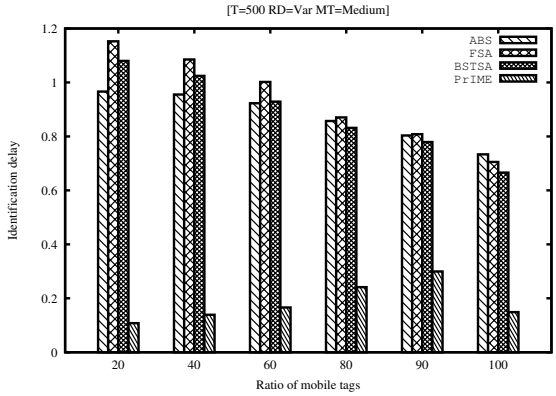

Fig. 4. Tag identification delay for the first identification.

mobile tags raises from 20% to 100%. The increasing ratio of identification experienced by BSTSA when increasing the percentage of mobile tags is due to the fact that BSTSA allows newly arrived mobile tags to join the identification process only at the beginning of a new reading cycle. Decreasing the number of static tags, the splitting process creates less groups, with a consequent reduction of the time taken by the reading cycle. Shorter reading cycles reduce the number of new mobile tags which wait for a new reading cycle. As a consequence new tags have more chances to be identified before leaving the system.

The difference in the identification delay between BSTSA and PriME is however very significant. Figure 4 shows the time taken by each protocol to identify high priority tags. PrIME outperforms all other protocol, being able to identify 99% of mobile tags in less than 0.3 sec. PrIME identification delay is always very small: it increases from 0.1s to 0.3s when the percentage of mobile tags increases from 20% to 90%. This behavior is due to the consequent decreasing height of the splitting tree that causes an increment in the value of the residual probability ϵ (see eq. 4). This means that more M_m tags will join lower level groups of the tree causing an increasing identification delay for high priority tags. In the case of 100% mobile tags, M_m tags participate in the splitting

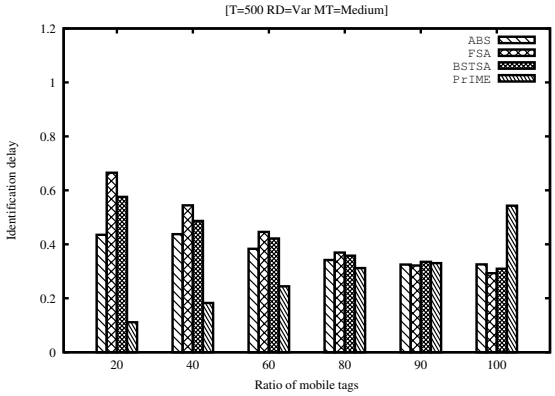

Fig. 5. Tag identification delay for the next identification.

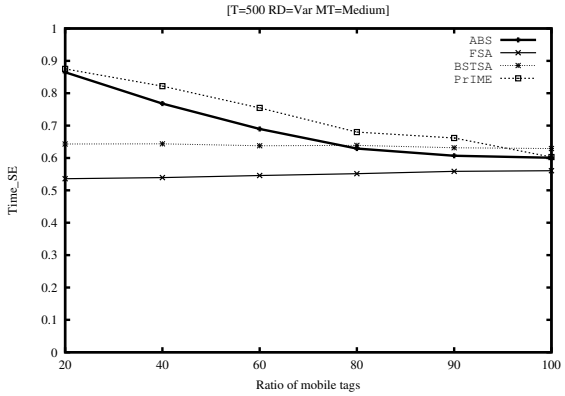

Fig. 6. Time System Efficiency.

process, placing more tags in the higher levels of the tree, allowing for a faster identification of new tags. The higher identification delay of other protocols is due to the fact that they do not allow the entrance of new tags while executing a reading cycle. A new tag must wait for the start of the next reading cycle to participate in the identification process.

The speed of PrIME in identifying new tags results in a higher delay in subsequent identifications when there are only mobile tags in the system (see Figure 5). This is an obvious consequence of an identification process based on priority: the higher the number of mobile tags entering the system, the longer the delay for subsequent identification of tags already identified.

Figure 6 shows the average temporal system efficiency per reading cycle, varying the percentage of mobile tags in the system. PrIME shows superior performance with respect to all the other protocols. One may wonder why PrIME and ABS show a similar decreasing trend while BSTSA and FSA time system efficiency remains almost constant by varying the percentage of mobile tags. Also BSTSA and FSA show a similar trend that instead remains constant by varying the percentage of mobile tags. PrIME behaves similarly to ABS because it uses an ABS process to identify static tags. When the majority of tags is static (first scenario with 20% of mobile tags) PrIME and ABS are very similar, and ABS is able to efficiently manage the entrance of mobile tags, showing a time system efficiency that is comparable to that of PrIME. PrIME achieves higher efficiency (10% increase) than ABS when mobile tags in the system range between 40% and 90%, due to its ability to manage new mobile tags. The other protocols, BSTSA and FSA, show instead a constant time system efficiency, as they do not distinguish between new and already identified tags, and each new reading cycle considers all the participating tags as new (no priority policy is used). In case of 100% mobile tags, all the protocols show a comparable time system efficiency (i.e., 62% for BSTSA, 60% for PrIME and ABS, 55% for FSA). This is not a weak aspect of PrIME. Instead this result shows that PrIME is able to properly estimate the number of tags participating in a reading cycle even if new tags (M_h) continuously enter the system, achieving the same efficiency of protocols that do not allow new entrances while executing a reading cycle. Figure 7 shows in details the trend in the estimation error during PrIME execution. After a few group identifications, the estimation error is quite low (on average about 20 units), oscillating inside the guard limits (dashed black lines), and hence having a negligible impact on the protocol performance.

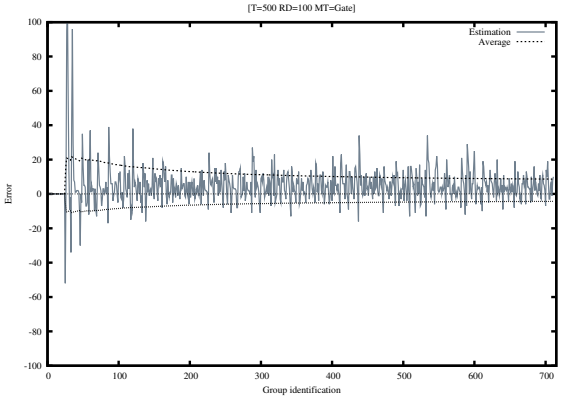

Fig. 7. PrIME estimation error for M_h tags during the identification process.

As a final experiment we stressed the PrIME protocol by investigating its performance under higher mobility. To this end we increased the flow of tags entering and leaving the system. Specifically, we simulated three new scenarios, with *high*, *very high*, and *extreme* mobility, and compared them with the previously studied scenario that we called *medium* mobility scenario. In the new three scenarios we increase the entrance flow of 10%, and the exit flow of respectively 0%, 3%, and 8% with respect to the medium mobility scenario. Figure 8 shows the average number of identified tags per reading cycle when the total number of tags is 500 and they are all mobile. The figure distinguishes between tags identified for the first time (patterned bars) and tags identified for a subsequent time (white mid section of the bar). Even with the highest number of mobile tags entering together the system (i.e. *Extreme* scenario), PrIME identifies over 98% of new mobile tags (i.e. M_h tags), showing consistent good performance under a wide

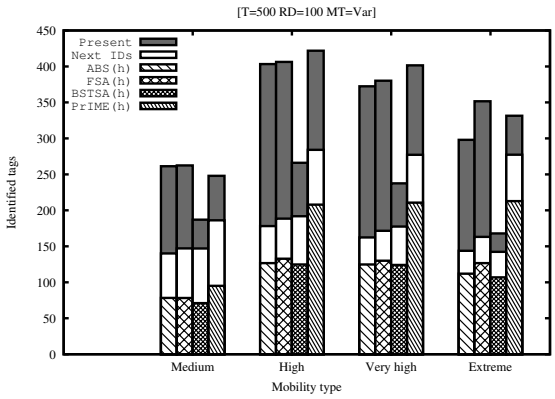

Fig. 8. Detailed identification of mobile tags varying the mobility density.

range of mobile scenarios.

V. Conclusions

In this paper we address the design of an anti-collision protocol for mobile RFID systems. We consider an hybrid system composed of a fixed reader, static and mobile tags. Mobile tags pass by the reader so that they are under its coverage radius for a limited amount of time. For this scenario we propose a new anticollision protocol, named PrIME, which builds on BSTSA and ABS, adding two main features: 1) different insertion policies in the identification tree for static and mobile tags, which ensure fast identification of mobile tags; 2) an accurate cardinality estimation of the different types of RFID in the system (mobile with high priority, mobile with medium priority, static), which is then used to dynamically estimate the protocol parameters. By means of extensive ns2-simulations PrIME is shown to significantly outperform previous solutions, represented by BSTSA, ABS and FSA. It is able to identify 98-99% of tags even under fast mobility, for all different combinations of mobile and static tags, and is very fast in identifying tags under the reader range.

Acknowledgment

This work was supported in part by the EC ARTEMIS CHIRON project (Contract n.100228.)

References

[1] La Porta, T.F.; Maselli, G.; Petrioli, C.; "Anticollision Protocols for Single-Reader RFID Systems: Temporal Analysis and Optimization", *IEEE Transactions on Mobile Computing*, vol.10, no.2, pp.267-279, Feb.2011

[2] CHIRON: Cyclic and person centric Health management: Integrated appRoach for hOme, mobile and clinical eNvironments. www.chiron-project.eu.

[3] Alien Technology; "Reader Interface Guide", 2010.

[4] EPCglobal Inc; "EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860MHz - 960MHz", ver.1.1.0, http://www.epcglobalinc.org/standards/uhfc1g2, 2006.

[5] Myung, J.; Lee, W.; "Adaptive binary splitting: a RFID tag collision arbitration protocol for tag identification", *in Proceedings of the 2nd International Conference on Broadband Networks*, vol.1, pp.347-355, 3-7 Oct.2005, Boston, USA

[6] Bolic, M.; Latteux, M.; Simplot-Ryl, D.; "Framed Aloha Based Anti-collision Protocol for RFID tags", *in Proceedings of SenseID 2007*, November 6, 2007. Sydney, Australia

[7] Hamouda, E.; Mitton, N.; Simplot-Ryl, D.; "Reader Anti-Collision in Dense RFID Networks With Mobile Tags", *in Proceedings of IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*, 15-16 September 2011. Barcelona, Spain

[8] Simon, L.; Saengudomlert, P.; Ketprom, U.; "Speed Adjustment Algorithm for an RFID Reader and Conveyor Belt System Performing Dynamic Framed Slotted Aloha", *in Proceedings of IEEE International Conference on RFID*, pp.199-206, 16-17 April 2008. Las Vegas, USA

[9] L. Xie, B. Sheng, C. Tan, H. Han, Q. Li, and D. Chen, "Efficient tag identification in mobile rfid systems", In *in Proceedings of IEEE INFOCOM* 2010. San Diego, USA

[10] J. Capetanakis, Tree algorithms for packet broadcast channels, *IEEE Transactions on Information Theory*, vol. 25, no. 5, pp. 505-515, 1979.

[11] J. Massey, Collision resolution algorithms and random-access communication, *Multiuser Communication Systems*, ed. G. Longo, Spriger, New York, no. 256, pp. 73-137, 1981.

[12] C. Law, K. Lee, and K.-Y. Siu, Efficient memoryless protocol for tag identification (extended abstract), in *in Proceedings of ACM DIALM 00*, New York, NY, USA, 2000, pp. 7584.

[13] Bonuccelli, M.; Lonetti, F.; Martelli, F.; "Instant collision resolution for tag identification in RFID networks", *Elsevier, Ad Hoc Networks*, vol.5, pp.1220-1232, 2007.