SafeRFID-MW: a RFID Middleware with runtime fault diagnosis

Rafik KHEDDAM, Oum-El-Kheir AKTOUF and Ioannis PARISSIS

Abstract: **In recent years, due to the proliferation of radio frequency identification (RFID) technology in everyday life, especially in critical domains such as health care and transportation systems, significant efforts have been made to enhance the dependability of this technology. From these efforts have emerged specific techniques and several middleware solutions to handle the large amount of the RFID data. These solutions are not suitable for all RFID system requirements especially for issues related to critical domains. In this paper, we propose a novel fault-tolerant RFID middleware providing two fault-tolerant mechanisms. The first mechanism is an online diagnosis algorithm based on a statistical analysis of the generated RFID data to identify faulty components of the system such as faulty readers or tags. The second mechanism is a verification process based on an extended finite state machine of the Low Level Reader Protocol (LLRP), the communication standard between RFID readers and RFID middleware. This process aims at identifying the causes of the diagnosed failures.**

Index terms**: probabilistic algorithm; online diagnosis; model based diagnosis; RFID middleware; dependability; LLRP; finite state machine.**

I. INTRODUCTION

Few years ago, radio frequency identification (RFID) systems were only deployed in single site scenarios, usually with few readers. Nowadays, this kind of architecture is not suitable for the new user business needs, such as data sharing with partner companies [1]. To meet these new requirements, a novel type of software artifact, called RFID middleware, has been designed for managing the various RFID sources and for processing the huge volume of generated raw data. Despite the technological advances in the radio frequency domain, RFID systems are still unreliable and very sensitive to their running environment. For example, the read rate of current RFID readers in real world deployments is often in the 60-70% range [2] [3] [4] [5] [6]. This means that for every 100 tag reading attempts, almost 70 of them succeed. Thus, it becomes crucial to design a middleware able to monitor and to diagnose RFID systems in addition to their usual features. But most existing fault-tolerant mechanisms allow only monitoring one reader at a time [7] (instead of considering the whole RFID system with all its readers) and are no longer suitable for the new RFID systems that are deployed in multiple sites scenarios with many readers connected together. Indeed, "not considering the whole system in the diagnosis process" usually leads to misdetection of some RFID failures. In addition, these monitoring mechanisms cannot locate the faulty component neither find the causes of the failure.

In this paper, we are interested in proposing on-line testing and fault tolerance facilities for RFID middleware. The first objective is to detect and to locate faulty components regarding the whole RFID system. Then, we focus on finding the causes of the detected failures. A RFID middleware called SafeRFID-MW regarding the research project name SAFERFID is designed to accommodate the proposed fault-tolerant mechanisms. This article is an extended version of a conference paper [8] published at Int. Conf. on Software, Telecommunications and Computer Networks (SoftCOM'2012).

The rest of the paper is organized as follows. Section II presents an overview of the RFID technology. Section III introduces some related work and the context of our research, followed by the presentation of the fault model in section IV. Sections V, VI and VII present the proposed middleware solution with its both diagnosis mechanisms. Finally, section VIII presents and discusses three different implementations of the proposed approach.

II. GENERAL DESCRIPTION OF A RFID SYSTEM

A RFID system is a contactless technology for uniquely and remotely identifying items by using radio waves [1] [9]. It is composed of three main layers: RFID devices with their drivers and air protocols, Middleware and User applications as shown in Fig. 1.

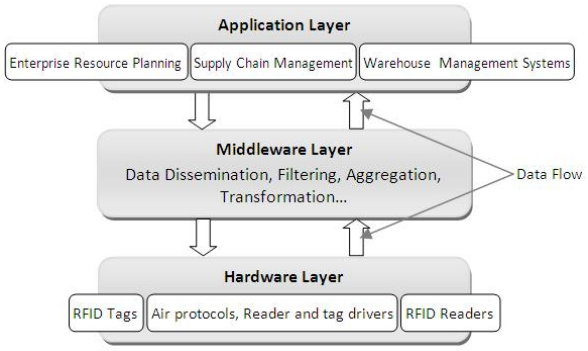


Fig. 1. RFID system architecture

- **The hardware layer** consists of Tags and Readers and their low level software.

Manuscript received December 15, 2012; revised February 28, 2013.

This work is part of a research project supported by the ANR agency (a French Research Agency) on RFID systems dependability called SAFERFID [www.Agence-Nationale-Recherche.fr].

Authors are with LCIS laboratory - Grenoble Institute of Technology, France. E-mails: firstname.lastname@lcis.grenoble-inp.fr

- Tags or transponders can be passive (*i.e.*, they get their energy from the signal of the reader and they use backscattering to communicate) or active (*i.e.*, they use their own energy contained in a small battery to operate). The tags uniquely identify an object as bar-codes do, and carry some useful information about it such as date of manufacture, expiry date of the product, preparation tips, *etc*.
- Readers are devices that read (*resp.*, write) the data from (*resp.*, to) the tag memories.

- **The middleware** is the heart of RFID systems. It processes the raw data generated by RFID devices, translates them into business events (such as "*Product ID = xxxx* has arrived at the warehouse on 01/22/2013 at 11:00 *am*, in an amount of 1000 *units*") and disseminates these events to client or enterprise applications. The main functions of a RFID middleware are:
  - Filtering the huge amount of received raw data; redundant or unnecessary data will be filtered.
  - Data dissemination; the middleware has to send exactly the requested information (neither more nor less) to the correct user application and it has to manage the privacy and security of the sent data (data access restrictions).
  - Data aggregation; all data of the same type or for the same application will be gathered to ease their diffusion.
  - Data interpretation; the middleware must know how to interpret the data for better management.
  - Management of the readers such as connection initiation, read/write operations, *etc*.

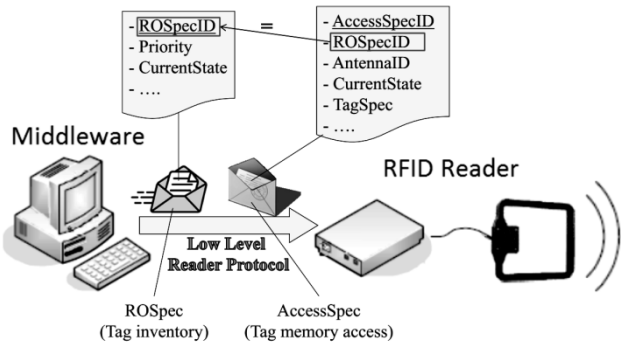


Fig. 2. LLRP Runtime Operation

RFID middleware communicates with the readers through an EPCglobal communication standard called Low Level Reader Protocol (LLRP). LLRP supports both main functions of a RFID reader, which are *tag inventory* (*i.e.*, the identification of all tags located in the reader field of view) and *tag memory access* as shown in Fig. 2. These two functions are provided by LLRP by using two XML based messages: *Reader Operation Specification* (ROSpec) and *Access Specification* (AccessSpec). ROSpec has an identification number, a state (Disabled, Inactive or Active), a priority and a set of commands and parameters required by the reader to perform a correct inventory. AccessSpec has an identification number, a state (Disabled or Active), a ROSpecID that specifies the corresponding ROSpec (*i.e.*, ROSpec that will be responsible of triggering it), a set of conditions (TagSpec) that specify on which tags this AccessSpec is applicable and a set of commands and parameters required by the reader to access correctly the various memories of the tags.

NB.
- An "active" ROSpec means the ROSpec is under execution.
- An "active" AccessSpec means the AccessSpec is ready for execution (*i.e.*, it is waiting for an active ROSpec to trigger it).

LLRP operates as follows. The middleware can define and send several ROSpec and AccessSpec (always in disabled state) to the reader. Each AccessSpec is affected to one ROSpec that will be responsible of triggering it. At a given time, only one ROSpec is under execution. Once some tags are identified by ROSpec and if these tags match the triggering conditions specified in the TagSpec of an active AccessSpec, this AccessSpec is triggered to perform additional operations on the selected tags such as retrieving data from their memories.
Fig. 3 shows the behavior of a LLRP compliant reader "with one ROSpec" in a three-state Finite State Machine (FSM). The transitions in this FSM are labeled "*X/Y*", where *X* represents the input of the reader (usually a request coming from the middleware) and *Y* is the output of the reader (usually a response to the middleware request). A ROSpec moves to "Inactive" state when the reader receives an "Enable(ROSpecID)" request, then waits for a start condition such as an event from a motion sensor or simply a request from the middleware, to start the inventory process (the ROSpec is then in "Active" state). When some tags are identified, the reader checks if these tags match any TagSpec of an "Active" AccessSpec belonging to the running ROSpec (AccessSpec moves to "Active" state by "Enable(AccessSpecID)" request from the middleware). If so, the reader performs the specified operations in the AccessSpec on the selected tags (tag memory access).

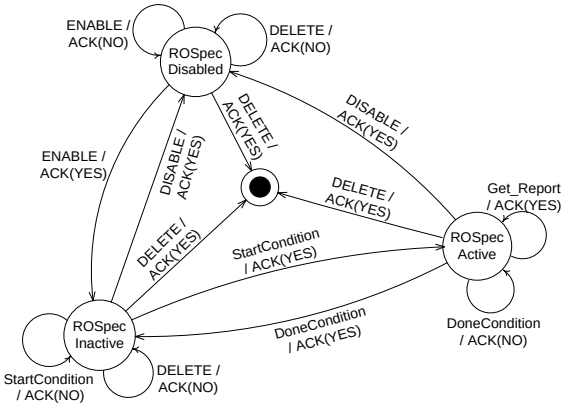


Fig. 3. A three state FSM of a reader

- **The user or enterprise applications** are programs that use the business events to provide different services for the end-users. The user applications are mainly Enterprise Resource Planning (ERP), Supply Chain Management (SCM), Warehouse Management Systems (WMS) or critical applications such as medical and transportation applications.

III. Related Work

RFID technology has been developed very rapidly over the last decade in many areas through a variety of applications. This requires a reliable deployment of RFID systems to meet user requirements that change over time. Issues related to the deployment of this technology rise when the used technology is passive RFID that is cheap but error prone; the read rate of such systems drops to 60% in presence of more than 5 tags [3]. This drop in accuracy is due to the tag signal collision. It can also be affected in general by environmental factors such as the presence of metal or liquid that reflect or distort the radio signal, or the presence of multiple readers (*i.e.*, radio wave interference), *etc*. Moreover, the increase of RFID applications and RFID sources makes the processing of the generated amount of data more difficult.

Several pieces of middleware have emerged to meet these RFID issues. WinRFID is a flexible and extensible RFID middleware developed by WINMEC RFID Lab [10] [11] capable of processing large amounts of data with real-time error fixing and recovering from faults and exceptions. RF²ID for "Reliable Framework for Radio Frequency IDentification" takes into consideration the unreliability of the RFID technology [12] [13]. RF²ID considers a mapping of the system (*i.e.*, the location of each reader) and manipulates some lists such as "observed Tag List" of the current reader, "expected Tag List" of the next reader, "missing Tag List", *etc.* to detect abnormal behaviors of the system such as the disappearance of a tag. Fosstrak for "Free and Open Source Software for TRAce and tracK" implements the EPCglobal *Inc.*[1] standards. It is designed to provide track and trace services that are specific to each user or enterprise application [14] [15]. Aspire RFID [16] [17] for "Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications" is an open source and OSGi based middleware solution. It implements several standards such as NFC and EPCglobal standards. It provides the main functions of RFID middleware such as data filtering and aggregation and it aims to facilitate and minimize the cost of deploying RFID technologies [18]. Sun RFID middleware is a Java based technology. It is designed to provide reliable integration of RFID technology in existing enterprise systems [19]. It is based on the notion of "event"; each received information such as "tag ID" is translated into an event that contains more useful information such as "the place where the tag is read", "the date", "the reader ID that identified the tag", *etc*. FlexRFID is a four-layer based architecture; hardware, application, processing and access management layers [20] [21]. FlexRFID operates as Sun RFID middleware with event processing. REFiLL [22] for "Rfid Event FiLtering tooLchain" is a flexible solution that gives the user the ability to define specific filtering rules for more adaptability to its needs.

The current RFID middleware solutions such as the ones listed above, focus mainly on managing the various data sources and processing the large amount of raw data. In return, they neglect or partially consider fault tolerance. The main methods used to handle faults in current RFID systems are *Remote Monitoring of the Reader Status* and *Reader Performance Monitoring* [7]. The Reader Status Monitoring is performed by a simple query using Simple Network Management Protocol (SNMP) such as "*ping*" to determine whether the reader is on and connected to the network or not. The Reader Performance Monitoring uses the variation of a given performance parameter to detect faulty components. This performance parameter can be:

- *Read Error Rate* (RER) of a reader [23]: is the number of erroneous reads during one inventory process[2].
- *Read Error to Total Reads* (RETR) [7]: is the number of erroneous reads over the total read attempts of a given reader (usually within several inventories).
- *Reading Accuracy or Read Rate*: is opposite to RETR; *i.e.*, it is the number of times that each tag is seen over all reading attempts.
- *Average Tag Traffic Volume* [7]: this approach needs a learning phase to determine a reference value. This reference value will be used to detect unusual tag traffic that is a possible indication of a faulty system.
- *The profile approach* [24]: is a set of read rates associated to each detected tag within the same group (*e.g.*, a pallet of products) according to a given reader. The ordered read rate curve represents the profile of this group of tags according to that reader. This approach verifies the compliance of a group of tags with its associated profile according to a given reader and *vice versa* (*i.e.*, the compliance of a reader with its associated profile according to a given group of tags). A couple "group of tags and reader" that does not match its associated profile is declared faulty; namely, one or both of them are faulty.

All these monitoring techniques present some lacks *i.e.*, poor diagnosis, especially in harsh environments, where failures come from various causes such as aging effects, medium disturbances (*e.g.*, electromagnetic bursts), *etc*. Indeed, these techniques do not have a global vision of the system as they monitor one reader at a time. So, when a RFID reader fails to read some tags, the aforementioned techniques cannot identify the real faulty components; the failure can be caused by a faulty reader[3], faulty tags or by a faulty air interface such as the presence of radio wave interference during the communication.

[1] EPCglobal *Inc.* specializes in the development of industry-driven standards to support the use of RFID [www.epcglobalinc.org].

[2] An inventory is a set of reader operations during which the reader tries to identify all tags located in its antenna scope.

[3] A faulty reader can be either a "broken" reader or a very "inaccurate" one (*e.g.*, a reader that has a damaged antenna).

RFID readers are very similar to distributed multiprocessor systems in their behavior. The readers are connected in a logical path due to the tag dataflow as the processors are connected together in a multiprocessor system. In this latter, we use the hardware redundancy to monitor the whole system as done by D. Fussell and S. Rangarajan in their work on multiprocessor system diagnosis [25] [26]. They used a comparison model to process failure probabilities of the processors. Because of these similarities between both systems, we propose a similar approach that uses reader performance parameters to on-line diagnose and monitor the RFID system. The proposed approach consists in a probabilistic algorithm that compares the performance parameters of the readers in order to detect and locate RFID failures. This approach focuses on large-scale deployments of RFID technologies such as airport baggage handling systems. The failures that we are interested in can be permanent or transient and they range from a broken device[4] (*i.e.*, broken reader or tag) to a damaged device (*e.g.*, damaged antenna) that does not block the Read/Write operations but affects the reader performances (*e.g.*, this leads to a high error frequency or to a low reading accuracy that will be less than 60%). When a failure is detected in the system, a complementary state verification process is triggered to check for the causes of the failure.

IV. FAULT MODEL

A. RFID failures

The failures that we are interested in are the following:

- The appearance or disappearance of a tag or group of tags (*e.g.*, a tag that is detected by a reader and not by another one).
- The non-response of a reader (*e.g.*, the reader is broken down).
- The low performance of a reader (*e.g.*, one of the reader antennas is damaged, radio frequency interference, *etc.*).

B. How the proposed approach can be used?

The first step of our approach, *i.e.*, the probabilistic algorithm, can be used to analyze the readers one by one or in groups in order to detect RFID failures.

B.1 Processing individual tags

In this case, we are interested in the tag ID that each reader identifies and we consider the following situations:

- Most readers identify correctly the tag; the tag is considered correct and the readers that do not identify it correctly are considered faulty.
- Most readers do not identify correctly the tag; these readers are declared:
  - **Faulty** (even if they represent the majority): if the remaining readers (the minority) are more than two and they identify the same tag ID. Indeed, the probability p to have N **faulty** readers ($N \geq 2$) that identify the same tag ID at the same time, knowing that the size of the tag ID is 96 bits, is very small ($p = \frac{1}{2^{96 \times N}}$).
  - **Fault-free**: in this case, if the minority is less than two; *i.e.*, there is only one reader that returns a correct tag ID (this tag identifier may be just a remaining tag ID in the memory of the reader and it is probably different from the real one). In this case, the tag and the said reader are declared faulty and the other readers (the majority) are fault-free.

The main concern of this approach is that processing the tags separately does not allow the detection of the failures whose consequences are reduced reading performance of the reader; *e.g.*, a damaged antenna of a reader that leads to a weak radio signal will not be detected because the reader will continue to identify the tags but with a weaker read rate than usual (*i.e.*, a reading accuracy less than 70%).

B.2 Processing tag groups

This case is twofold.

- We can consider the observed tag list of each reader. This approach was used in the aforementioned RF²ID middleware. It has the same drawbacks than processing individual tags. Indeed, it does not detect the failures that do not block the Read/Write operations but affect the reader performances.
- We can focus on the variation of a given reader performance parameter according to each group of tags. This requires a learning phase to determine the reference value of the chosen performance parameter. Examples of reader performance parameters were given in section III.
  - When most readers match the predefined performance parameter, the remaining readers are considered faulty; *i.e.*, the minority that does not match the performance parameter is faulty.
  - When most readers do not match the predefined parameter, the group of tags and the other readers are declared faulty.

Processing the tags in groups detects more failures (such as failures due to reduced reader performance) than monitoring the tags separately. However, it cannot locate which tag is really faulty in the group and in general, when a failure occurs, we cannot say if the failure comes from a faulty reader, faulty tags or from environmental disturbances.

In the rest of this paper, we consider monitoring of tag groups with the use of the profile performance parameter. The details of the profile approach can be found in [24]. We will focus on the performance result of each reader; *i.e.*, if the reader matches its performance profile or not (binary decision).

[4] A broken device leads to impossible Tag/Reader communication (No read/write operations).

V. SAFERFID-MW: A RFID MIDDLEWARE WITH RUNTIME FAULT DIAGNOSIS

A. Architecture

The SafeRFID-MW architecture we propose, takes in consideration the unreliable nature of RFID systems in addition to providing the main services of a common RFID middleware. This architecture can be represented as a three-layer model as shown in Fig. 4.

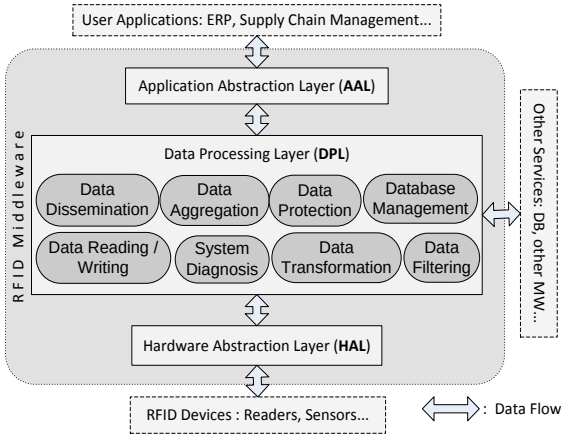


Fig. 4. SafeRFID-MW architecture

A.1 Hardware Abstraction Layer (HAL)

HAL is responsible of interfacing operations with hardware components such as:

- Unification of the way the middleware interacts with the hardware components.
- Reception of data from various sources, data control and transmission to Data Processing Layer (DPL) or to readers.

A.2 Data Processing Layer (DPL)

DPL is the main layer of the middleware. It is responsible of most functions of the middleware, such as data filtering, aggregation, *etc*. The main sub-components of this layer are:

- *Data Dissemination*; responsible of data diffusion according to predefined rules.
- *Data Aggregation*; responsible of counting and data aggregation according to a given criterion (same type of reader or data intended for the same application, *etc.*).
- *Data Protection*; responsible of protecting and managing application accesses to data (*i.e.*, data access restrictions) and services provided by the middleware.
- *Database Management*; responsible of organizing and sharing the processed data.
- *Data Reading/Writing*; responsible of collecting data through commands that it sends to readers and for data writing on the tag memories.
- *Data Transformation*; responsible of processing and formatting the raw data in various formats for greater flexibility.
- *Data Filtering*; responsible of removing unnecessary data.
- *System Diagnosis*; responsible of monitoring and diagnosing the whole RFID system through some fault-tolerant approaches that we will discuss later.

A.3 Application Abstraction Layer (AAL)

AAL provides user applications with an interface for accessing various services offered by the middleware.

B. Fault-tolerance features

SafeRFID-MW will accommodate two diagnosis mechanisms: Probabilistic Diagnosis Algorithm and Model Based Diagnosis that can be used together or separately. The first one consists in comparing the read results, more precisely the profile of the readers to identify outliers or abnormal behavior of RFID devices such as “a reader is not responding”, “a high error frequency of a reader”, “some tags are not read”, *etc*. A probability is associated to each identification of an outlier by using a probabilistic model. This probability informs the user about the accuracy of the diagnosis according to the runtime conditions. The second mechanism is a formalization of the LLRP protocol as a finite state machine (FSM). This FSM is used with usual test sequence generation techniques to identify some failure causes. An extension of this FSM is proposed and used in a state verification process to deal with more failures and their causes. In the rest of this paper, because of the complexity of both aforementioned diagnosis mechanisms, we will devote a single and entire section for each one of them.

VI. RFID DIAGNOSIS ALGORITHM

The proposed probabilistic diagnosis algorithm consists of three steps as shown in Fig. 5. The first one is partitioning the readers into several groups according to the data flow. The second one compares all reader results in the same group to identify faulty components (faulty readers or tags). These comparisons are based on the variation of the profile parameter. We recall that a profile is an ordered set of read rates of the tags of a same group that forms a curved line. If the couple (reader, group of tags) closely follows its predetermined profile (*i.e.*, the curved line), we consider that this couple matches its associated profile and that it is fault-free. Otherwise, the couple is declared faulty. The third step estimates the precision of the diagnosis. This precision is represented by a probability whose calculation will be explained later. In the rest of this paper, we refer to the proposed diagnosis algorithm by *RFID diagAlgo*; its formalization is given in the annex A.

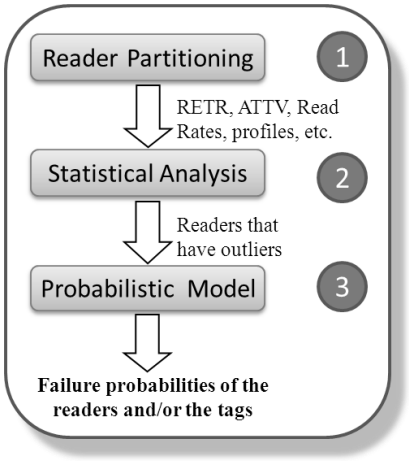


Fig. 5. RFID Diagnosis Algorithm

A. Reader Partitioning

The idea of grouping the readers according to the nature of the data flow which is the first step of our approach makes sense when the RFID technology is used in big warehouses, where the management of product storage is important. Thus, each type of the moving items (*e.g.*, pallets) always follows the same physical path from an entry point to the designated destination in the warehouse. We can also mention airport baggage handling systems equipped by the RFID technology. All passengers' luggage of the same flight follow the same physical path. Thus, we can monitor the readers located in the same dataflow path (*e.g.*, luggage of the same flight) together in order to detect their failures by *RFID diagAlgo*.

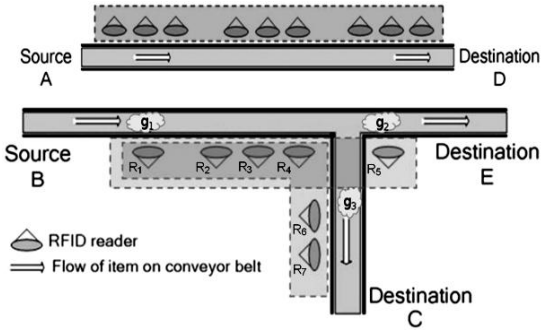


Fig. 6. Grouping of readers according to the data flow

Fig. 6 emphasizes the partitioning of the RFID readers of the baggage handling system in groups according to the origin and the destination of the tagged items. So, the readers $\{R_1, R_2, R_3, R_4, R_5\}$ will be analyzed together according to the tags that follow the path $\{B, E\}$; the readers $\{R_1, R_2, R_3, R_4, R_6, R_7\}$ will be analyzed according to the tags of the path $\{B, C\}$ and so on.

B. Reader Results Comparison (RRC)

The result comparison is the second step of our diagnosis algorithm. It consists of comparing the results of all readers that have analyzed some tags at a given time. TABLE I shows an example of reader results comparison, where readers from different paths are analyzed simultaneously to get global and consistent diagnosis decisions as some readers may belong to different paths. TABLE I indicates if a given reader R_i matches its associated profile ($M(R_i, g_j) = 1$) or not ($M(R_i, g_j) = 0$) regarding the group of tags g_j. $S(g_j)$ indicates which component (reader or group of tags) is faulty when analyzing the group of tags g_j. $R_i \in S(g_j)$ if R_i has a result ($M(R_i, g_j)$) different from the one of most readers. For example, $R_2 \in S(g_2)$ because $M(R_2,g_2) = 1$ and all the other readers belonging to the same path, *i.e.* $\{R_1, R_3, R_4, R_5\}$, have $M(R_i, g_2) = 0$, $i \in \{1, 3, 4, 5\}$. In addition, g_2 is indicated as faulty because of the results of the majority of the readers (*i.e.*, if most readers do not match their associated profiles according to a group of tags, these tags are declared faulty). $SF(\,)$ indicates which reader is faulty; $R_i \in SF(\,)$, if $\exists\ g_j$, $R_i \in S(g_j)$. $SF(\,)$ can be adjusted so as to minimize false-positives; *e.g.*, a reader R_i will be declared faulty only if it has been declared faulty twice according to two different groups of tags. So, in this example, R_6 and R_7 will not be declared faulty in $SF(\,)$.

TABLE I
READER RESULTS COMPARISON

M	R_1	R_2	R_3	R_4	R_5	R_6	R_7	$S(g_j)$
g_1	1	0	1	1	1	-	-	$\{R_2\}$
g_2	0	1	0	0	0	-	-	$\{R_2, g_2\}$
g_3	1	1	1	1	-	0	0	$\{R_6, R_7\}$
SF( )								***{R2, R6, R7}***

At this stage of our algorithm, we can determine the probability of failure of each tag. For example, g_2 is analyzed by five readers and declared faulty by four of them, so its failure probability is 4/5. In addition, the temporary failure probability of each reader is statistically estimated without taking into account the runtime conditions. This failure probability of a reader will be refined each time the read results of this reader are compared with the ones of the other readers; *e.g.*, R_2 is declared faulty twice after it reads 3 groups of tags, so, its failure probability is 2/3. These failure probabilities of the readers are used by the Probabilistic Model (*cf.* section VI.C) to process the precision of the diagnosis at that moment. This precision of the diagnosis can also be considered as the real failure probability of the identified faulty components.

In the rest of this paper, we take the example of TABLE I as a thread of the proposed approach that we will keep consistent throughout this paper.

C. Probabilistic Model

In this model, a probability is associated to each decision about the state of the reader. This probability will give the user an indication about the precision of the diagnosis according to the current system configuration and the runtime conditions. This probability is calculated on the basis of four parameters; the values of these four parameters represent the configuration of the system at a given time. These parameters are: the failure probabilities of each reader processed in Reader Results Comparison step, the number of readers, the number of processed groups of tags and a particular probability called *Rational Behavior*. *Rational Behavior* is specific to each reader and

represents the probability that the existence of a failure in the reader will have visible effects on the system. This parameter characterizes especially intermittent faults of each reader where the failure appears and disappears according to the reader configuration. But it should be noted that the proposed probabilistic model takes in consideration all RFID failures; intermittent and permanent faults, hardware and software faults, internal and external faults (*i.e.*, faults due to environment disturbances). More details about Rational Behavior parameter will be given in section VIII.

We call the ability to correctly identify the state of the readers (faulty or fault-free) ***Identifiability*.**

C.1 Identifiability

Identifiability represents the precision of failure detection; *i.e.*, it characterizes the reliability of the diagnosis of *RFID diagAlgo*. It takes into account the runtime conditions such as the number of the identified tags or group of tags, the number of readers, the *Rational Behavior* and the reader failure probabilities computed in RRC. So, the *Identifiability* is specific to each decision that the diagnosis algorithm "*RFID diagAlgo*" makes about the state of each reader/group of tags at a given configuration of the system (*i.e.*, according to the values of the four parameters of the *Identifiability*).

RRC aims to identify:

- a real fault-free reader as fault-free (*Correct Negative*).
- a real faulty reader as faulty. In other words, RRC tries to avoid the case where a faulty reader is declared fault-free (*False Negative*).

So, the *Identifiability* depends on these two cases (correct and false positive) as we will see later.

a) Correct Negative

Let n be the number of readers analyzing the groups of tags, p the probability of failure of a reader (we have considered that each reader has the same failure probability for sake of readability of the formula below) and $C(x, y)$ the binomial coefficient.

A fault-free reader is identified as such in RRC after processing of t groups of tags if:

- At least half of all readers are fault-free and their results are the same as the one of the reader under analysis. This case is represented by the following probability:

$$\sum_{i=\left\lfloor \frac{n}{2} \right\rfloor}^{n-1} C(i, n-1) \times (1-p)^{i} \times p^{n-1-i} \qquad (1)$$

- At least half of all readers are faulty, but for each group of tags, there are at least half of all readers that correctly process this group, and then have the same result than the reader under analysis. This case is represented by the following probability:

$$\sum_{i=\left\lfloor \frac{n}{2} \right\rfloor}^{n-1} \left[C(i, n-1) \times p^{i} \times (1-p)^{n-1-i} \times \left(\sum_{j=i+1-\frac{n}{2}}^{i} C(j, i) \times (1-r)^{j} \times r^{i-j} \right)^{t} \right] \qquad (2)$$

where r is the *Rational Behavior* probability of the reader. We recall that it indicates the probability that a faulty reader will have a faulty behavior and will process the tags incorrectly[5]. We will see later how this parameter is estimated.

If $CN(t, n, p, r)$ is the probability to have a correct negative, then

$$CN(t, n, p, r) = (1) + (2)$$

b) False Negative

A faulty reader will be considered as fault-free after processing of t groups of tags if:

- The reader under analysis processes correctly the t groups of tags. So, it has the behavior of a fault-free reader. The probability of this case is represented by the following formula:

$$CN(t, n, p, r) \times (1 - r)^{t} \qquad (3)$$

- The reader x under analysis processes incorrectly i groups of tags ($1 \leq i \leq t$), but for each one of them, there are at least half of all readers which are faulty and have the same result as the one of x. This case is represented by the following probability:

$$\sum_{i=1}^{t} \left[\left(\sum_{j=\left\lfloor \frac{n}{2} \right\rfloor}^{n-1} C(j, n-1) \times (p \times r)^{j} \times \left((1-p) + p \times (1-r) \right)^{n-1-j} \right)^{i} \times C(i, t) \times r^{i} \times (1-r)^{t-i} \times CN(t-1, n, p, r) \right] \qquad (4)$$

If $FN(t, n, p, r)$ is the probability to have a false negative, then

$$FN(t, n, p, r) = (3) + (4)$$

[5] In some cases, a faulty reader will correctly identify the tags in its field of view (*e.g.*, the reader has a damaged antenna, but not yet used, the path that contains the error in its integrated circuit is not used, *etc.*).

So, the *Identifiability* is processed as follows

$$\boldsymbol{I(t, n, p, r)} = (1 - p) \times \boldsymbol{CN(t, n, p, r)} + p \times (1 - \boldsymbol{FN(t, n, p, r)})$$

From the formula of the *Identifiability*, we can see that to have a better diagnosis precision, we must reduce the *False Negative* parameter and increase the *Correct Negative* one.

If we commonly calculate the *Identifiability* of the failure of the reader R_2 of the previous example of TABLE I, we proceed as follows: the number of groups of tags is $t = 3$, the number of readers is $n = 7$. We recall that for sake of simplicity, the *Identifiability* formula considers the average failure probability of the readers instead of considering the specific failure probability of each reader, so, the average failure probability is $p = (0 + 2/3 + 0 + 0 + 0 + 1 + 1) / 7 \approx 0.38$ and we assume that the *Rational Behavior* is for example $r = 0.9$ (more details about how to compute the *Rational Behavior* are given in section VIII). So, the *Identifiability* will be $I(3, 7, 0.38, 0.9) \approx 0.98$.

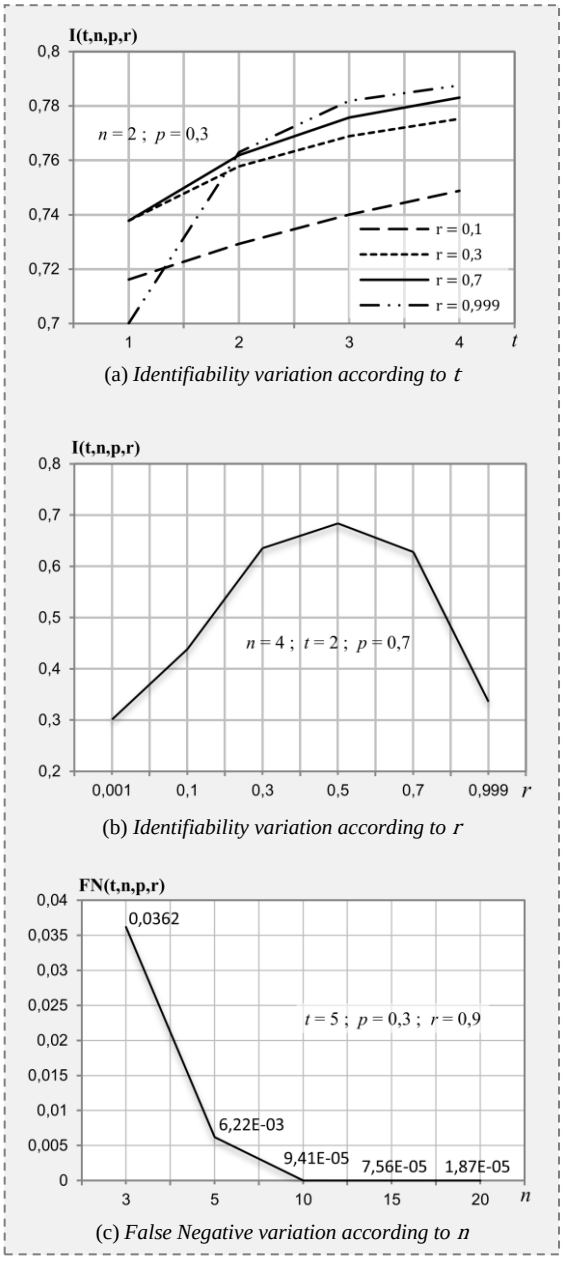
C.2 Evaluation of the Identifiability

We have calculated the probability $I(t, n, p, r)$ in different configurations to see how the *Identifiability* behaves according to each one of its parameters. The test scenario is the following: p and r get their values in {0, 0.001, 0.1, 0.3, 0.5, 0.7, 0.999, 1}; n varies from 2 to 20; and finally, the parameter t gets its values in {1, 2, 3, 4, 5}.

After the calculation of the *Identifiability* with the test scenario defined above, we have obtained 6080 cases. We can summarize the behavior of the *Identifiability* in Fig. 7.

Fig. 7 (a) shows that the precision of the diagnosis (the *Identifiability*) grows up with increasing number of groups of tags *t*. The *Identifiability* has the same behavior with growing values of *r*. This is consistent, because with a high value of *r*, more faulty readers will exhibit their failures, and so they will be detected.

In Fig. 7 (b), $I(t, n, p, r)$ is increasing with r until it reaches its maximum value at $r \cong 0.5$. After this, $I(t, n, p, r)$ decreases with the increasing of r. With a great value of p ($p = 0.7$), most readers will be faulty. When r is small; *i.e.*, the faulty readers do not exhibit their failures; namely, they process the tags correctly and it is hard to *RFID diagAlgo* to detect them ($I(t, n, p, r)$ is small). The precision of *RFID diagAlgo* grows up with the increasing of the value of r, because there are more and more readers that exhibit their failures, and then they are diagnosed as faulty ones. But when r becomes greater than 0.5, the growth of the value of r has the opposite effect on $I(t, n, p, r)$; most of the faulty readers make the wrong decision (because $r > 0.5$), and because of the faulty readers represent the majority, *RFID diagAlgo* will consider the readers that have made the incorrect result as fault-free ones, and the other ones as faulty. It is clear that when r grows up, the number of faulty readers that make the wrong decision grows up and the number of incorrect analysis on the readers made by *RFID diagAlgo* will increase; *i.e.*, $I(t, n, p, r)$ will decrease.



(a) *Identifiability variation according to t*

(b) *Identifiability variation according to r*

(c) *False Negative variation according to n*

Fig. 7. Evaluation of the Identifiability

We can also use this approach to determine how many groups of tags the system has to process for a better precision in detecting failures. TABLE II is an example where $p = 0.1$ and $r = 0.5$ that shows the number of groups of tags required on the basis of the number of readers present in the RFID system. For example, if our system has 5 readers, the precision of the algorithm is at its maximum with 9 groups of tags; (*i.e.*, $I(t, n, p, r) = 0{,}997426$). But we can see that from $t = 4$, the *Identifiability* is approximately the same as the maximum. For purpose of optimization, *RFID diagAlgo* can take a decision just after processing 4 groups of tags, instead of wasting time in waiting for other incoming groups of tags.

TABLE II
NUMBER OF REQUIRED GROUPS OF TAGS ACCORDING TO THE NUMBER OF READERS

t \ n	3	4	5	6
1	0, 943	0, 943475	0, 94889	0, 948958
2	0, 966071	0, 963647	0, 97351	0, 973067
3	0, 978362	**0,9726**	0, 985853	0, 984806
4	0, 984838	**0,976236**	**0,991992**	**0,990405**
5	0, 988196	**0,977424**	**0,995001**	**0,992968**
6	0, 989893	**0,977545**	**0,996438**	**0,994045**
7	**0,990716**	0, 977249	**0,99709**	**0,994405**
8	**0,991086**	0, 976833	**0,997352**	**0,994429**
9	**0,991229**	0, 976423	**0,997426**	0, 994306
10	**0,991262**	0, 976067	0, 997413	0, 994125
11	0, 991246	0, 975775	0, 997361	0, 993932

As said before, one of our objectives is to reduce the FN (*False Negative*) parameter. We effectively minimize its impact on the precision of the diagnosis of *RFID DiagAlgo*. This is made possible with the growing number of readers, and so, the precision of failure detection increases. Fig. 7 (c) shows that while the number of readers is increasing, FN decreases to reach a very small value at $n = 10$. Fig. 7 (c) is another example that shows how we can use this approach in real RFID systems to process the best value of one parameter of the *Identifiability* knowing the three other ones to have a few false negatives; *e.g.*, if a RFID system has the same configuration as in Fig. 7 (c) (*i.e.*, $t = 5$, $p = 0.3$ and $r = 0.9$), we will set the size of each group of readers to analyze together to 10 to have a very accurate diagnosis.

In the remaining sections, a complementary diagnosis approach is presented to allow a precise location and identification of the causes of the identified failures.

VII. MODEL BASED DIAGNOSIS

This approach is based on LLRP, the communication standard between RFID readers and RFID middleware to monitor the behavior of the readers. In this approach, we propose to extend the LLRP protocol to make it able to handle RFID failures. Indeed, LLRP is a complex and complete communication protocol with error notifications, but it is not able to detect misconfiguration errors neither to locate the origin and the causes of the failures. LLRP is very flexible and gives the user the full power in the definition of the Reader Operation and Tag Memory Access specifications. But this usually leads to misconfiguration errors such as "the user has activated the wrong ROSpec", "the AccessSpec is not linked with the right ROSpec", *etc.* Effects of this kind of errors can be "the reader does not identify all tags located in its field of view", "the reader does not retrieve the needed information from the tag memories", *etc.* In the subsequent sections, we will show how to deal with this kind of failures with our proposed LLRP extension.

A. LLRP failures

We refine the RFID failures presented earlier (*cf.* section IV) and classify them according to their causes into two categories as follows:

A.1 Failures due to inconsistent design or misconfiguration

- Masking of useful data or even blocking all data by the middleware.
- Mismatch between the received and the expected data by the middleware or the reader.

A.2 Failures due to runtime conditions

- Slow execution / component overload.
- No data capture.
- Erroneous received data.

B. Failure Processing

As said before, RFID readers are inherently inaccurate. This usually leads to the transmission of erroneous data to the RFID middleware. The proposed solution to deal with the aforementioned failures consists of two mechanisms. The first one uses a state verification process using the existing LLRP model to deal with the failures due to inconsistent design and misconfiguration. The second one extends the LLRP model to deal with the failures not covered by the first mechanism (*i.e.*, failures due to runtime conditions).

B.1 Using the existing LLRP model

Failures due to inconsistent design or misconfiguration are caused by the fact that one of the participants (middleware or reader) in the communication is in the wrong state; *i.e.*, the middleware (or the reader) is in a state where it does not expect the received commands or data. Then, as a consequence, although they are useful data, they will be ignored and an error message may be returned. For example, let us assume that a reader is in a state where the ROSpec is disabled instead of being inactive (the user omits to enable the ROSpec or the reader does not interpret correctly the "Enable" request of the middleware). So, when some tags enter in the field of view of the reader, the motion sensor sends a signal to the reader in order to start the tag inventory. But, this signal is just ignored and the tags are not read.

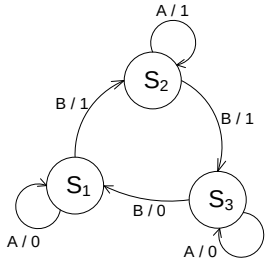


Fig. 8. Arbitrary three-state FSM

This kind of failures is handled without any need to extend the existing LLRP model. All we have done is translating the textual specification of LLRP protocol into a finite state machine (FSM). This LLRP FSM is used with usual test sequence generation techniques ("Distinguishing Sequences") to deal with these failures. A distinguishing sequence is a set of transitions that allow identifying the initial and current state of

the reader or the middleware [27]. This is useful to deduce the exact configuration of the reader (*i.e.*, number of ROSpec and AccessSpec, the state of each one of them, which AccessSpec belongs to which ROSpec, *etc.*) when the failure occurs. This FSM consists in 18 states and more than 200 transitions. For sake of simplicity, we will only show how to exploit the Distinguishing Sequence technique in an example to identify the failure causes. So, let us take a simple FSM such as the one in Fig. 8. To identify the state of the reader or the middleware when the failure occurs, we construct the successor tree of this FSM. A successor tree shows the behavior of an FSM starting from all possible states under all possible input sequences.

Fig. 9 shows an example of a successor tree built up from the FSM of Fig. 8. It shows the transition sequences that distinguish between the different states of the FSM. When a failure occurs, the FSM can be in any state. So, the objective is to identify this state.

In this successor tree, the input *B* separates the state S_1 from S_2 and S_3; *i.e.*, if we have an output "0" (*resp.*, "1") after applying the input *B*, we are sure that the current state of the FSM is S_1 (*resp.*, S_2 or S_3). Then, the input *A* separates the state S_2 from S_3. At the end of this state verification process, if we have a sequence such as {*B*/1, *A*/0}, then, we are sure that the current state of the FSM is S_3 and the **initial state** at the launch of this verification process is S_2, *i.e.*, the unique state from which the sequence {*B*/1, *A*/0} is possible. So, the sequence {*B*/1, *A*/0} is a distinguishing sequence of the state S_2. Other examples of distinguishing sequences are: {*A*/1} for S_2, {*A*/0, *B*/0} for S_3, {*A*/0, *B*/1} or {*B*/1, *B*/1} for S_1, *etc.*

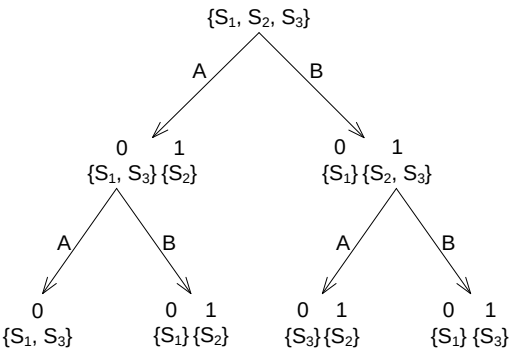


Fig. 9. The successor tree of the arbitrary FSM

Now, let us see how to use this technique in a real case. For instance, returning to the previous example of TABLE I, if the failure is a "tag inventory" problem, this technique can indicate if the problem comes from the following misconfiguration errors: "the user or SafeRFID-MW has activated the wrong ROSpec on R_2", "the ROSpec specifies the wrong reader antenna", *etc.* If the failure is a "tag memory access" problem such as "Masking of useful data by the reader" (*e.g.*, the middleware receives only the tag identifiers but not their memory contents), SafeRFID-MW sends to the reader specific commands using the distinguishing sequence technique to check if this error is due to a wrong internal state of the reader; *i.e.*, the AccessSpec is not executed to access the tag memories. At the end of the diagnosis, we deduce that the reader's initial state was one of the following, which is the cause of the failure.

- *Case 1:* ROSpec is "Active" but not the AccessSpec.
- *Case 2:* ROSpec and AccessSpec are "Active" but the AccessSpec is not in the list of AccessSpec to be triggered by the running ROSpec.
- *Case 3:* ROSpec and AccessSpec are "Active" but the identified tags do not satisfy the triggering conditions of the AccessSpec.

If it turns out that the causes of the failure does not come from misconfiguration errors, the proposed LLRP extension comes in to check the runtime conditions in order to identify the failure causes.

B.2 Extending the LLRP model

In this step, we propose to extend the LLRP model to handle the failures caused by the runtime conditions.

1) Slow execution / Component overload

These failures may be caused by the amount of raw data to process that leads the middleware to slow down. We should know that SafeRFID-MW associates to each new reader an instance of the LLRP FSM to permanently maintain consistent communications with it. So, to check that the source of this problem comes from the huge amount of data that are fed back to it, SafeRFID-MW needs just to read all the instances of LLRP FSM looking for all the readers that are in "communicating state". Then, it only remains to verify that the number of the current communicating readers does not exceed its capacity. If it does, as a possible solution, SafeRFID-MW will ask the readers that have internal memory to stop the data transfer and to temporary store the data in their memories.

2) No data capture / erroneous received data

These failures are also caused by inconsistent design and misconfiguration, but they may also be caused by:

- The reader antenna is damaged or broken.
- The tags are out of the reader field of view or the signal power level of the reader is too weak.
- The tags move too fast. So, the reader has not enough time to identify all tags.
- The tags are not supported by the reader or on the contrary, the selected air protocol is not supported by the tags.
- The presence of external disturbances (radio waves of other devices, water, metal objects, *etc.*).

Fig. 10 emphasizes the LLRP extension. The lozenges in the diagram represent the inputs in the FSM, and the leaves (the circular shapes) are the causes of the failure.

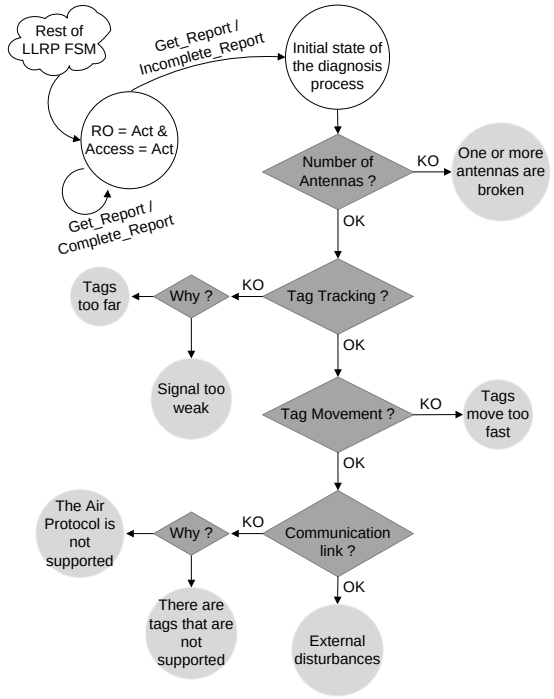


Fig. 10. LLRP failure diagnosis diagram

As we can see in Fig. 10, when the reader is executing a ROSpec with an AccessSpec (ROSpec and AccessSpec are active), SafeRFID-MW may ask for the result of these specifications. If the report is incomplete (*e.g.*, there are tags that are not inventoried correctly, impossible read/write of tag memories, *etc.*), SafeRFID-MW triggers the diagnosis process. The failure causes are hierarchically sorted according to their diagnosis difficulties. So a "broken antenna" is easier to diagnose than a "faulty communication link" as we will see below:

- SafeRFID-MW can verify if an antenna is damaged or broken by asking the reader to give it back the number of "operational" antennas. If the reader is not able to perform this action and knowing that most readers have at least two antennas (one for transmission and one for reception), SafeRFID-MW can transmit a signal by an antenna to receive it by the other one. If this operation succeeds, we assume that the antennas are not broken. To verify if one of them is damaged, SafeRFID-MW will vary the receive sensitivity[6] or the signal power level of the antenna to determine the communication scope of the antenna as a damaged antenna will cover a smaller area. So, it is possible to deduce if the antenna is damaged or not (the distance between the antennas must be known).
- For the second cause of failure (*i.e.*, "the tags are out of the scope of the antenna"), SafeRFID-MW will locate the tags by varying the power level or the receive sensitivity of the antenna as explained above. Fig. 11 shows an area containing four readers, and how to exactly locate a tag to verify if it is out of the scope of a reader. For example, when the tag or group of tags T_2 is not identified correctly by the reader R_1 while it should be, the SafeRFID-MW will exploit the neighbors of R_1 to locate T_2 by reducing the scope of these readers enough to deduce that T_2 is out of R_1 field of view.

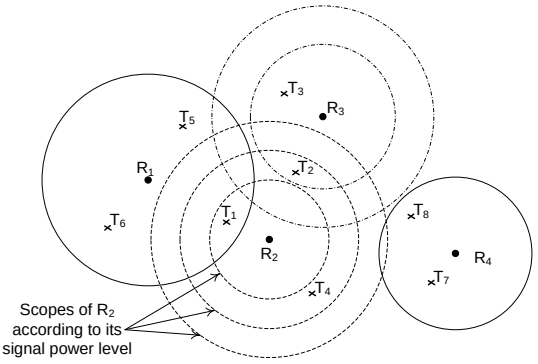


Fig. 11. Tag localization technique

- To deduce that the failure is due to tags that move too fast, the middleware can proceed as follows:
  - If the tags are carried by a conveyor belt, Safe RFID-MW has just to know the speed of the conveyor belt and to compare it with the maximum speed supported by the reader.
  - The second possibility is to calculate the speed of the tagged item using its two previous positions (or three, to calculate the tag speeding), also by using the neighbors of the reader under analysis.
- To detect the failure of the communication link, we proceed as follows:
  - If erroneous data are received by the middleware SafeRFID-MW, then we are sure that the problem comes from the communication link between the reader and SafeRFID-MW; otherwise the erroneous data will be filtered by the reader.
  - To deduce the causes of the problem when it comes from the communication link between the reader and the tags, SafeRFID-MW will ask the neighbors of the concerned reader which air protocol they use to successfully identify the tags. So, SafeRFID-MW can deduce that the reader has used a non-supported protocol. Otherwise it will check if the reader does not support the concerned tags (*e.g.*, UHF reader that try to read HF tags) always by asking its neighbors.
- If none of these causes are found, we assume that the failure is due to external disturbances such as the presence of metal objects between the reader and the tags.

VIII. RFID DIAGALGO IMPLEMENTATIONS

We have already seen that *Identifiability* depends on four parameters. As mentioned before, The *Rational Behavior* is specific to each reader and concerns only its failures (*e.g.*, damaged antenna, errors in the reader software, *etc.*). A possible way for processing the *Rational Behavior* can be by pro-

[6] Receive sensitivity indicates how faint a radio frequency signal can be successfully received by a given receiver. The lower the power level that the receiver can successfully process, the better the receive sensitivity [30].

cessing the number of execution paths (hardware and software paths); *e.g.*, Fig. 12 shows an example of a control flow graph (side *b*) according to the program of side (*a*) that we can use to process the parameter *r*. In this case, we have ($N = 4$) possible execution paths. By using the Halstead measures[7], we can estimate the number (A) of errors in the program that will have visible effects on the system [28]. So, the *Rational Behavior* in this example will be $r = A/N$.

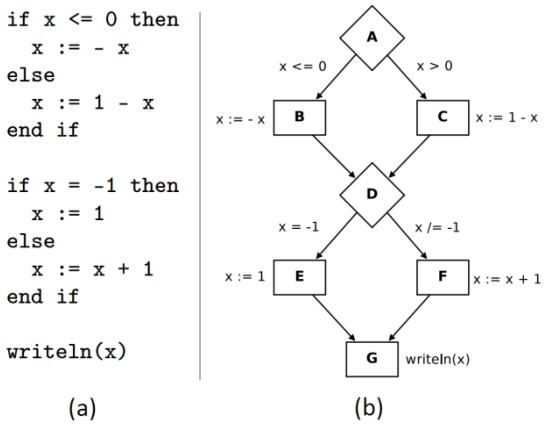


Fig. 12. Control flow graph

The same approach can be used to compute the *Rational Behavior* of a reader. This case requires taking into consideration the reliability of the hardware components. So, we need the following elements to estimate the reader's *Rational Behavior*:

- The number of software failures that have visible effects on the system (Halstead measures).
- The number of hardware failures that have visible effects on the system; it can be computed by performing a Failure mode and effect analysis (FMEA) on the reader such as the one proposed in [29].
- The number of software execution paths; it can be computed by using the source code of the software.
- The number of Hardware execution paths; it can be determined by using the electronic circuit diagram of the reader.

In a real world deployment, RFID readers are usually black boxes (*i.e.*, we do not have access to their architecture details); so the processing of the Rational Behavior parameter is impossible. In such cases, we will take in consideration only the failures that have observable effects on the system (*i.e.*, no silent errors). This means that *r* will have a great value such as 0,999 or 1; this means that when a failure occurs, its effect will be observable on the system.

RFID diagAlgo works as follows. After the reader partitioning phase, it proceeds to the verification of the state of the reader under analysis. Then, it verifies the state of the current group of tags once the monitored reader is declared fault-free. This algorithm has a complexity of $O(t \times n^2)$ and is presented in the annex A. We have deliberately chosen this representation without optimization for sake of simplicity. For information, the execution time of *RFID diagAlgo* once it gets all reader results does not exceed 20 *milliseconds*[8].

We have implemented *RFID diagAlgo* in several ways. Each one will be discussed to show its advantages and disadvantages. The main implementation is to analyze all reader results together at one time. But because of slowness of this approach, we looked for other alternative implementations. So, the second proposed implementation is to analyze the readers in disjoint subgroups of 5 readers. The last implementation is to analyze the readers as the reader results arrive. This approach can be seen as a moving window that moves along the set of the readers.

A. Calculation done at the end

In this implementation, the middleware starts the process of analyzing the RFID data once it has all the reader results. This is interesting regarding the precision of the diagnosis but it takes much time (indeed the middleware waits until all tags are read by all readers in the path of these tags). This waiting period is in the range of seconds or even minutes while the execution time of the diagnosis process when the reader results are available is in the order of milliseconds. For this reason, we propose two alternative implementations that are much faster and we will discuss later how reliable (in terms of diagnosis precision) they are compared to the main implementation (*i.e.*, Calculation done at the end).

B. Alternative implementations

B.1 Calculation done five by five

In this implementation, the middleware starts the diagnosis process every time it has the results of 5 readers. When the five readers are analyzed by the middleware, it moves to the five next readers, and so on. We have set the size of the reader group under analysis to five, because, with five readers, the middleware has enough data to perform mostly a correct diagnosis and does not take much time such as in the first implementation.

B.2 Moving window implementation

In this implementation, the middleware analyzes 5 readers at a time as in the second implementation. The difference between these implementations lies in the selection of the five readers to be analyzed. Indeed, in this approach, the progress speed of the Moving Window varies according to the analyzed readers at each step. When the readers are analyzed by the middleware, the faulty ones are ejected from the window, and the same number of readers is added (from the remaining readers in the path) to the moving window. If all readers of the window are fault-free, only the first reader in the window is ejected as shown in Fig. 13.

[7] The Halstead measures introduced by Maurice Howard Halstead in 1977 are functions of the number of operators and operands in the program to estimate the number of errors in an implementation.

[8] *RFID diagAlgo* is tested on an Intel® Core™ i5 processor-based machine.

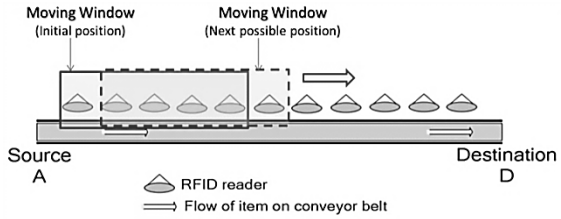


Fig. 13. Moving window approach

C. Discussion

Both alternative implementations are much faster than the original one (*i.e.*, Calculation done at the end). The speed of the original implementation depends on the number of groups of tags (tagged items), the number of readers in the system and the time that the tagged items take to reach each reader. The processing time of each implementation is negligible compared to the required time to have the results of all readers (*i.e.*, few milliseconds against several minutes). So, the main challenge here is to show that the alternative implementations are reliable enough (in addition to their speed).

To evaluate these alternative implementations, we have compared their accuracy and their detection rate with those of the original implementation in three different scenarios. Each scenario has 44 test cases for a total of 923 readers to diagnose. The number of the participating readers in each test case varies from 3 to 60 readers and the number of groups of tags varies from 2 to 6 groups. A test case is implemented as a matrix (readers, groups of tags) as shown on TABLE I.

Each line of the test matrix represents the results of one reader. We recall that when $R_i(g_j) = 1$ (*resp.*, $R_i(g_j) = 0$), the couple (R_i , g_j) matches its associated profile; *i.e.*, the reader R_i does not exceed the limit of the predetermined profile parameter according to the group g_j (*resp.*, does not match its profile and this can be caused by a faulty reader or a faulty group of tags).

A reader is considered faulty if it exceeds the limit of the performance parameter at least one time. The process of verifying if a reader exceeds or not the performance parameter limit is performed by another program that is not presented here.

TABLE III
SAMPLE DIAGNOSIS RESULT

```
Running TEST (7 readers / 3 groups of tags):
 Moving window:        Calc. at the end:     Calc. 5 by 5:
  | R1:  OK             | R1:  OK             | R1:  OK
  | R2:  0,9993592773   | R2:  0,9881458421   | R2:  0,9993592773
  | R3:  OK             | R3:  OK             | R3:  OK
  | R4:  OK             | R4:  OK             | R4:  OK
  | R5:  OK             | R5:  OK             | R5:  OK
  | R6:  0,9999199097   | R6:  0,9881458421   | R6:  OK
  | R7:  0,9999199097   | R7:  0,9881458421   | R7:  OK
Time needed (all methods): 0.047425581 seconds.
Time needed by:
   Moving window method:      0.017562847 seconds.
   Calc at the end method:    0.028806175 seconds.
   Calc 5 by 5 method:        4.19886E-4 seconds.
-----------------------------------------------------------------
```

TABLE III shows a sample diagnosis result performed by all three implementations on the example of TABLE I as a three-column table. This table contains the failure probabilities of the faulty readers according to test matrix of TABLE I. TABLE III shows also the processing time of each proposed implementation.

C.1 Scenario 1

This case represents a simulation of a little disturbed environment (with 44 test matrices) that has a majority of fault-free readers (665 fault-free readers against 258 faulty ones). The result of this simulation is shown in Fig. 14.

- **Correct Negative:** Fault-free reader is considered as such.
- **Correct Positive:** Faulty reader is detected.
- **False Negative:** Faulty reader is not detected
- **False Positive:** Fault-free reader is considered as faulty one.

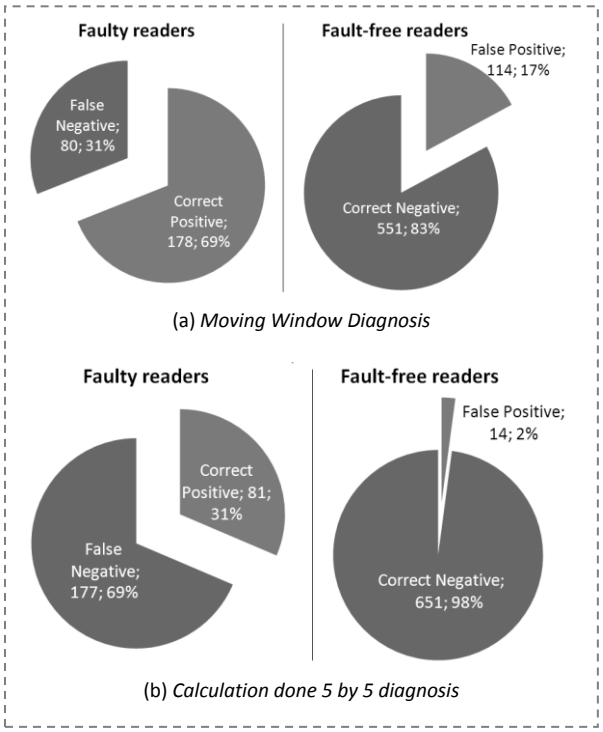


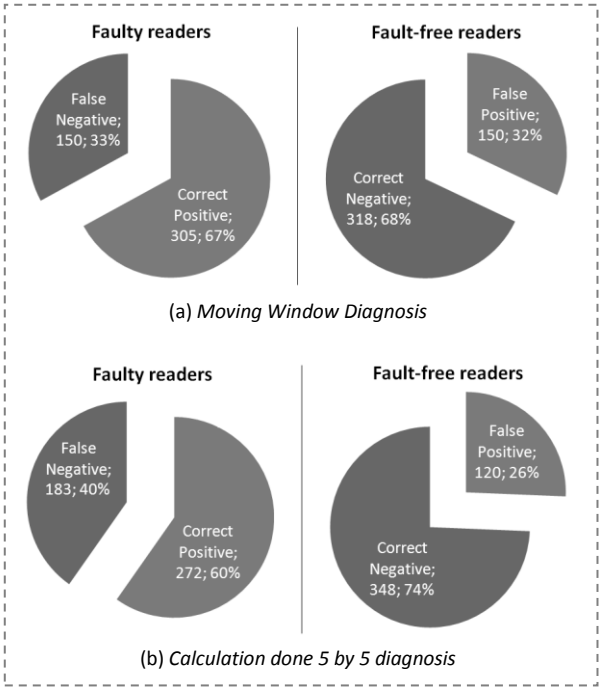
Fig. 14. Scenario 1

C.2 Scenario 2

This case represents a simulation of an intermediate disturbed environment that has nearly the same number of fault-free readers (468 fault-free readers) as faulty ones (455 faulty readers). The result is shown in Fig. 15.

C.3 Scenario 3

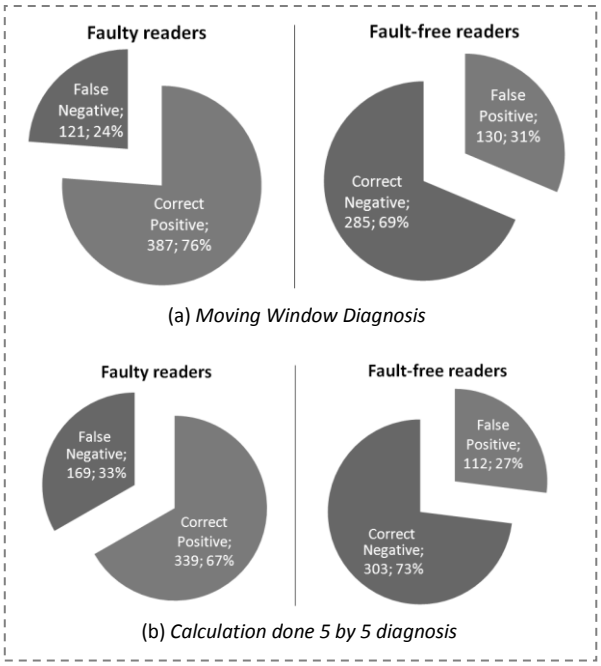
This case represents a simulation of a highly disturbed environment that has a majority of faulty readers (508 faulty readers against 415 fault-free ones). The result of this simulation is shown in Fig. 16.



(a) *Moving Window Diagnosis*

(b) *Calculation done 5 by 5 diagnosis*

Fig. 15. Scenario 2



(a) *Moving Window Diagnosis*

(b) *Calculation done 5 by 5 diagnosis*

Fig. 16. Scenario 3

All these figures show the same trend. The moving window approach detects more failures than the *calculation done 5 by 5* approach regardless the execution environment. But also it generates more false alarms (*i.e.*, false positive). In addition, the *Moving Window* approach is faster than the *calculation done five by five* approach because after the first position of the window, the middleware has just to wait for the results of usually one next reader to include it in the next window rather than five next readers for the second implementation.

The only cases where the *Moving Window* approach is inefficient are when there are at least three faulty readers in the same window at the beginning of the diagnosis process. This will distort all the diagnosis. Indeed, the faulty readers constitute the majority. So, the real fault-free readers will be considered as faulty each time and ejected from the moving window.

So, the choice between these implementations depends on the dependability policy adopted by the end-user; if he wants to detect the RFID components failure as soon as possible, with a maximum failure detection rate, the *Moving Window* implementation is the best, if he wants a diagnosis mechanism that gives fewer false alarms and that works in a reasonable time, the *Calculation done 5 by 5* approach is better, and finally, if he wants to have a more reliable diagnosis even if it takes much more time, then *Calculation done at the end* approach is the best.

IX. Conclusion

We have presented in this paper a probabilistic diagnosis approach based on a statistical analysis to monitor the readers and tags in a RFID system. It consists of three steps. The first one is the calculation of the neighbors of each reader according to the analyzed tags. The second step is the comparison of all results of all readers, and then in the third step, once a faulty reader or tag is identified, the proposed approach will associate to this decision a probability called *Identifiability* that takes in consideration the runtime conditions. After that, we have presented a state verification mechanism based on an extension of the communication standard between RFID readers and middleware to refine the diagnosis. Finally, we have presented three different implementations of the probabilistic diagnosis, and we have shown which one is better according to the running environment and the objectives of the end user.

References

[1] P. Krishna and D. Husak, "RFID INFRASTRUCTURE," *IEEE Communications Magazine*, vol. 45, no. 9, pp. 4-10, September 2007.

[2] S. R. Jeffery, M. Garofalakis and M. J. Franklin, "Adaptive cleaning for RFID data streams," in *Proceedings of the 32nd international conference on Very large data bases*, Seoul, Korea, 2006.

[3] R. Derakhshan, M. E. Orlowska and X. Li, "RFID Data Management: Challenges and Opportunities," *IEEE International Conference on RFID*, pp. 175-182, 2007.

[4] L. Catarinucci, R. Colella, M. De Blasi, L. Patrono and L. Tarricone, "Experimental Performance Evaluation of Passive UHF RFID Tags in Electromagnetically Critical Supply Chains," *Journal of Communications Software and Systems*, vol. 7, no. 2, pp. 59-70, 2011.

[5] L. Catarinucci, R. Colella and L. Patrono, "On the use of passive UHF RFID tags in the pharmaceutical supply chain: a novel enhanced tag versus hi-performance commercial tags," *International Journal of Radio Frequency Identification Technology and Applications (IJRFITA)*, vol. 4, no. 2, 2013.

[6] L. Yang, J. Cao, W. Zhu and S. Tang, "A Hybrid Method for achieving High Accuracy and Efficiency in Object Tracking using Passive RFID," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Lugano, pp. 109-115, March 2012.

[7] P. Sanghera, F. Thornton, B. Haines, F. Kung Man Fung, J. Kleinschmidt, A. M. Das, H. Bhargava and A. Campbell, How to Cheat at Deploying and Securing RFID, Massachusetts, USA: Syngress Publishing, Inc., Elsevier, Inc., 2007.

[8] R. Kheddam, O.-E.-K. Aktouf and I. Parissis, "Online monitoring and diagnosis of RFID readers and tags," in *20th IEEE International Conference on Software, Telecommunications and Computer Networks*, Split, Croatia, 2012.

[9] T. Hassan and S. Chatterjee, "A Taxonomy for RFID," IEEE 39th International Conference on System Sciences, Hawaii, 2006.

[10] UCLA WINMEC, "RFID@WINMEC - RFID Research," 2005. [Online]. Available: http://www.winmec.ucla.edu/rfid/winrfid.

[11] B. S. Prabhu, X. Su, H. Ramamurthy, h.-C. Chu and R. Gadh, "WinRFID – A Middleware for the enablement of Radio Frequency Identification (RFID) based Applications," *in Mobile, Wireless and Sensor Networks: Technology, Applications and Future*, pp. 331-336, 2005.

[12] N. Ahmed, R. Kuma, R. S. French and U. Ramachandran, "RF²ID: A Reliable Middleware Framework for RFID Deployment," in *International IEEE Parallel and Distributed Processing Symposium*, Long Beach, CA, March 2007.

[13] N. Ahmed, "Reliable Framework for Unreliable RFID Devices," 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Mannheim, 2010.

[14] C. Floerkemeier, M. Lampe and C. Roduner, "Facilitating RFID Development with the Accada Prototyping Platform," in *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops. PerCom Workshops '07.*, Zurich, Switzerland, 2007.

[15] C. Floerkemeier, C. Roduner and M. Lampe, "RFID Application Development With the Accada Middleware Platform," *IEEE Systems Journal*, pp. 1-13, 2007.

[16] ASPIRE Project, "ASPIRE - The EU funded project that brings RFID to SMEs," 2009. [Online]. Available: http://www.fp7-aspire.eu.

[17] OW2 Consortium, "AspireRFID Architecture," 2009. [Online]. Available: http://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation/AspireRfidArchitecture.

[18] J. Soldatos, "AspireRFID Can Lower Deployment Costs," 16 Mars 2009. [Online]. Available: http://www.rfidjournal.com/article/view/4661.

[19] A. Gupta and M. Srivastava, "Developing Auto-ID Solutions using Sun Java System RFID Software," October 2004. [Online]. Available: http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html.

[20] M. E. Ajana, H. Harroud, M. Boulmalf and H. Hamam, "FlexRFID: A Flexible Middleware for RFID Applications Development," IEEE Press, IFIP International Conference on Wireless and Optical Communications Networks, WOCN '09., Cairo, Egypt, 2009.

[21] A. Sengupta and S. Z. Schiller, "FlexRFID: A design, development and deployment framework for RFID-based business applications," *Information Systems Frontiers*, vol. 12, no. 5, pp. 551-562, November 2010.

[22] A. P. Anagnostopoulos, J. K. Soldatos and S. G. Michalakos, "REFiLL: A lightweight programmable middleware platform for cost effective RFID application development," *Pervasive and Mobile Computing*, vol. 5, no. 1, pp. 49-63, February 2009.

[23] G. Fritz, V. Beroulle, O.-E.-K. Aktouf, M.-D. Nguyen and D. Hély, "RFID system on-line testing based on the evaluation of the tags Read-Error-Rate," *IEEE, Mixed-Signals, Sensors and Systems Test Workshop*, pp. 1-10, 2010.

[24] G. Fritz, B. Maaloul, V. Beroulle, O.-E.-K. Aktouf and D. Hély, "Read rate profile monitoring for defect detection in RFID Systems," in *IEEE RFID-Technologies and Applications (RFID-TA)*, 2011.

[25] S. Rangarajan and D. Fussell, "A Probabilistic Method for Fault Diagnosis of Multiprocessor Systems," in *IEEE 18th International Symposium on Fault-Tolerant Computing*, Tokyo, Japan, 1988.

[26] D. Fussell and S. Rangarajan, "Probabilistic diagnosis of multiprocessor systems with arbitrary connectivity," in *IEEE 19th International Symposium on Fault-Tolerant Computing*,

FTCS-19. Digest of Papers., Chicago, IL, pp. 560-565, 1989.

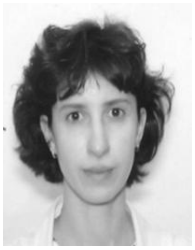
[27] D. Lee and M. Yannakakis, "Principles and Methods of Testing Finite State Machines - A Survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090-1123, 1996.

[28] J. P. Kearney, R. L. Sedlmeyer, W. B. Thompson and M. A. Gray, "Software complexity measurement," *journal of Commun. ACM*, vol. 29, no. 11, pp. 1044-1050, Nov. 1986.

[29] G. Fritz, V. Beroulle, O.-E.-K. Aktouf, M. D. Nguyen and D. Hély, "RFID System On-line Testing Based on the Evaluation of the Tags Read-Error-Rate," *Journal of Electronic Testing, SpringerLink*, June 2011.

[30] D. A. Westott, D. D. Coleman, P. Mackenzie and B. Miller, CWAP - Certified Wireless Analysis Professional Official Study Guide: Exam PW0-270, Indianapolis, Indiana: Wiley publishing Inc., 2011.

Rafik Kheddam obtained his engineering degree (in "*Advanced Information Systems and Software Engineering*") in 2009 from UMMTO university in Algeria and a Master degree of "*Safety and Security Software*" in 2010 from Faculty of Science and Technology of Besançon, France. Since December 2010, he is a PhD student at the MSTII Doctoral School and CTSYS team member of LCIS laboratory / teacher at the Grenoble Institute of Technology, Esisar. His research interests in LCIS laboratory concern the definition of software fault-tolerance and on-line testing mechanisms for RFID systems.



Oum-El-Kheir Aktouf has been as an Assistant Professor with Grenoble Institute of Technology, Esisar Engineering School and LCIS laboratory, since 1999. Her research interests concern dependability of embedded systems, especially sensor-based systems and RFID systems, using on line tests and diagnosis approaches. Her teaching activities concern operating systems, real time systems, distributed computing and computing systems' dependability. She obtained her Phd degree from Grenoble Institute of Technology on June 1997.

Ioannis Parissis is Full Professor at Grenoble INP and member of the LCIS laboratory since 2008. He was an Assistant Professor at Université Joseph Fourier, Grenoble, France, from 1999 to 2008. His research interests are related to software engineering and, more specifically, to validation and verification issues. His main research interests are automated test generation from formal specifications, testing techniques for embedded synchronous programs, test coverage criteria for dataflow programs as well as test modeling and test generation for interactive applications.

ANNEX A Probabilistic Diagnosis Algorithm (***RFID diagAlgo***)

Inputs: Read results of each reader.
Outputs: Failure probabilities of each faulty reader and / or groups of tags.

```
    // R is a set of readers to be analyzed by the algorithm.
    // G is a set of groups of tags processed by R.
1   begin
2       n = |R|; t = |G|;
3       FT ≔ ∅; // The set of Faulty Tags or groups of tags.
4       FR ≔ ∅; // The set of Faulty Readers.
5       T[n] ≔ {0, 0, …, 0}; // T[u]: number of groups of tags read by reader u.
6       r ≔ 0.99; // Most failures of the faulty readers have visible effects on the system.
7       nbFailR[n] ≔ {0, 0, …, 0}; // Failure counters for each reader, (array of n counters).
8       nbFailT[t] ≔ {0, 0, …, 0}; // Failure counters for each group of tags, (array of t counters).
9       failureProba[n] = {0, 0, …, 0}; // Failure probabilities of all readers.
10      x ≔ 0; // number of groups of tags already processed.
11
12      for (g ∈ G){
13          x++;
14          for (u ∈ R_g) { // R_g is the set of readers that process the group g with R_g ⊂ R.
15              computeProfile(u,g); // The profile approach is the chosen performance parameter,
16                                   // (see section III for more information).
17              T[u] ≔ T[u] + 1; // Counting the number of groups of tags processed by u.
18              if (D_u(g) = 0) {
19              // D_u(g)=0 means "g does not match its original profile according to the reader u".
20                  nbFailT[g] ≔ nbFailT[g] + 1; // number of times g is declared faulty.
21              }
22          }
23          compute(S(g)) ; // S(g) contains the faulty components (faulty readers and/or tags).
24          for (u ∈ R) {
25              if (u ∈ S(g)) {// u is faulty on g.
26                  nbFailR[u] ≔ nbFailR[u] + 1;
27                  failureProba[u] ≔ nbFailR[u] / T[u] ;
28              }
29          }
30          if (g ∈ S(g)) { // g is a faulty group.
31              FT ≔ FT ∪ {g};
32              print("g is faulty with a probability", nbFailT[g] / (n × I(n, x, failureProba, r)));
33          }
34      }
35      compute(SF); // SF contains the real faulty readers.
36      for (u ∈ SF) {
37          FR ≔ FR ∪ {u};
38          print("u is faulty with a probability", I(n, t, failureProba, r));
39      }
40  end
```