Approches logicielles de sûreté de fonctionnement pour les systèmes RFID

Rafik Kheddam

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

« Rafik KHEDDAM »

Thèse dirigée par **« Ioannis PARISSIS »**
codirigée par **« Oum-El-kheir AKTOUF »**

préparée au sein du **Laboratoire de Conception et d'intégration des Systèmes (LCIS)**
dans **l'École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatiques (MSTII)**

Approches logicielles de sûreté de fonctionnement pour les systèmes RFID

Thèse soutenue publiquement le **09 avril 2014,**
devant le jury composé de :

M. Pierre-Cyrille HEAM
Professeur à l'université de Franche-Comté, Examinateur
Mme. Nathalie MITTON
Chargée de recherche à l'INRIA de Lille, Rapporteur
M. Yves LE TRAON
Professeur à l'université Luxembourg, Rapporteur
M. Ioannis PARISSIS
Professeur à Grenoble INP, Directeur de thèse
Mme. Oum-El-Kheir AKTOUF
Maitre de conférences à Grenoble INP, Co-encadrant

Université Joseph Fourier / Université Pierre Mendès France /
Université Stendhal / Université de Savoie / Grenoble INP

Remerciements

Je remercie en premier lieu mes encadrants : M. Ioannis PARISSIS, Professeur à Grenoble INP et Mme Oum-El-Kheir AKTOUF, Maître de Conférences à Grenoble INP. Je les remercie chaleureusement pour leur disponibilité, pour tout le temps qu'ils m'ont consacré, pour leur accompagnement et tous les conseils qu'ils n'ont cessés de me donner tout au long de cette thèse.

Je suis très reconnaissant à Mme. Nathalie MITTON, chargée de recherche à l'INRIA de Lille et M. Yves LE TRAON, professeur à l'université Luxembourg ; je les remercie vivement d'avoir accepté d'être les rapporteurs de ma thèse.

Je remercie aussi M. Pierre-Cyrille HEAM, Professeur à l'université de Franche-Comté, d'avoir accepté de présider le jury de cette thèse de doctorat.

Je remercie également toutes les personnes participant au projet SAFERFID spécialement M. Vincent BEROULLE, chef du projet et Maître de Conférences à Grenoble INP, M. Christophe MEDINA, ingénieur au RFTLab et M. Christophe CHANTEPY, responsable du RFTLab, pour leur collaboration et pour le temps et leurs conseils.

Mes remerciements s'adressent également tout le personnel du LCIS et de l'Esisar et à toutes les personnes qui m'ont aidé, de prêt ou loin, à la réalisation de cette thèse.

Enfin, je remercie ma famille et ma fiancée Lynda CHOUALI, pour leur aide, leur patience et leur soutien inconditionnel durant toutes ces années d'études. Merci !

Table des matières

Chapitre 5 - Implémentation et simulation de l'algorithme RFID diagAlgo

Conclusion générale

Bibliographie

Annexes

Liste des Figures

Liste des Tables

Introduction générale

I. Problématique

L'identification des objets est utilisée depuis très longtemps. Autrefois, nous nous servions de la forme des objets, de leur toucher ou de leur odeur pour les identifier ou identifier la famille à laquelle ils appartiennent. Avec les avancées technologiques, plusieurs techniques d'identification ont vu le jour, notamment l'identification par code-barres qui est encore utilisée de nos jours et qui représente la technologie d'identification la plus répandue au monde. Mais cette dernière présente l'inconvénient d'être lente (un seul objet identifié à la fois) et de nécessiter une vision directe des codes-barres. Une autre technologie d'identification similaire, appelée RFID (Radio Frequency IDentification) était utilisée dans le domaine militaire. La RFID a été utilisée durant la deuxième guerre mondiale sous le nom de **IFF** (**I**dentify **F**riend or **F**oe) [1] [2] pour différencier les avions des alliés de ceux des ennemis. Par la suite, cette technologie s'est répandue dans le domaine public et a été utilisée pour la traçabilité des objets importants (de grande valeur) ; ce qui justifiait de tels investissements [3].

La technologie RFID présente l'avantage d'identifier plusieurs objets en même temps "sans contact, ni vision directe" et de manière unique grâce notamment à l'identification par EPC (Electronic Product Code) qui fournit pour chaque objet un code unique et universel. Depuis quelques années, la RFID est devenue omniprésente dans notre quotidien. Les innovations techniques ont permis aux systèmes RFID de s'affranchir des sources d'énergie embarquées (nécessaires à la communication), ce qui a rendu les tags moins chers, plus petits et donc plus accessibles et mieux adaptés à divers domaines comme la logistique. La RFID est utilisée notamment pour l'amélioration des chaînes d'approvisionnement et le suivi des marchandises, pour la lutte contre le vol dans les supermarchés, pour le contrôle d'accès, *etc.* Elle est aussi utilisée dans l'internet des objets (Internet of Things), un nouveau réseau dans lequel, tous les objets sont connectés et en mesure de communiquer entre eux (informatique ambiante) afin d'offrir de nouveaux services pour améliorer le confort des utilisateurs, en rendant la consommation d'énergie plus intelligente [4] et l'environnement plus sécurisé.

De nos jours, la RFID est déployée sur plusieurs sites géographiquement éloignés avec souvent des besoins en sûreté de fonctionnement, notamment lorsque cette technologie est utilisée dans les domaines critiques tels que le domaine médical, l'aéronautique, le ferroviaire, *etc*. Pour répondre à ce besoin, un nouveau composant RFID a fait son apparition. Il porte le nom de « *middleware RFID* ». Il a le rôle de gérer les nombreuses sources de données RFID, de collecter, contrôler, traiter et distribuer les données vers les applications concernées. Malgré les avancées technologiques dans le domaine de la radiofréquence, la RFID reste non fiable et sensible à son environnement d'exécution. Par exemple, le taux de lecture des lecteurs RFID actuels dans un environnement réel est d'environ 70% [5] [6] [7] [8] [9] ; *i.e.*, pour 100 tentatives de lecture, seules 70 d'entre elles réussissent à identifier des tags RFID. Ainsi, il devient nécessaire d'imaginer et de concevoir des solutions de tolérance aux fautes à intégrer dans les middlewares RFID afin d'améliorer la fiabilité de tout le système et assurer une continuité des services fournis. Cela est d'autant plus important lorsque cette technologie est utilisée dans des domaines où la sûreté de fonctionnement est primordiale. La tolérance aux fautes dans les systèmes RFID n'est pas encore arrivée à maturité ; Il existe peu de travaux de recherche dans ce domaine en comparaison avec d'autres aspects comme la sécurité et la confidentialité des données RFID. Ajouté à cela, la plupart des solutions de tolérance aux fautes existantes se focalisent sur la surveillance des composants RFID séparément. Elles ne sont donc pas adaptées aux environnements distribués avec plusieurs sources de

données connectées. En effet, ne pas considérer la totalité du système dans l'approche de monitoring conduit souvent à la non détection des défaillances ou à un diagnostic pauvre comme l'impossibilité de localiser la défaillance ou d'identifier sa cause.

II. Projet SAFERFID

Dans le cadre de la thématique de tolérance aux fautes dans les systèmes RFID, le projet SAFERFID supporté par l'Agence Nationale de Recherche ANR[1] depuis octobre 2010 a vu le jour au sein de l'équipe CTSYS (Conception et Test de SYStèmes embarqués) du Laboratoire de Conception et d'Intégration des Systèmes (LCIS). L'objectif à travers ce projet est d'accroître la robustesse globale des systèmes RFID à différents niveaux (tag, lecteur et middleware) en proposant des méthodes de durcissement logicielles et matérielles comme le montre la figure 1.

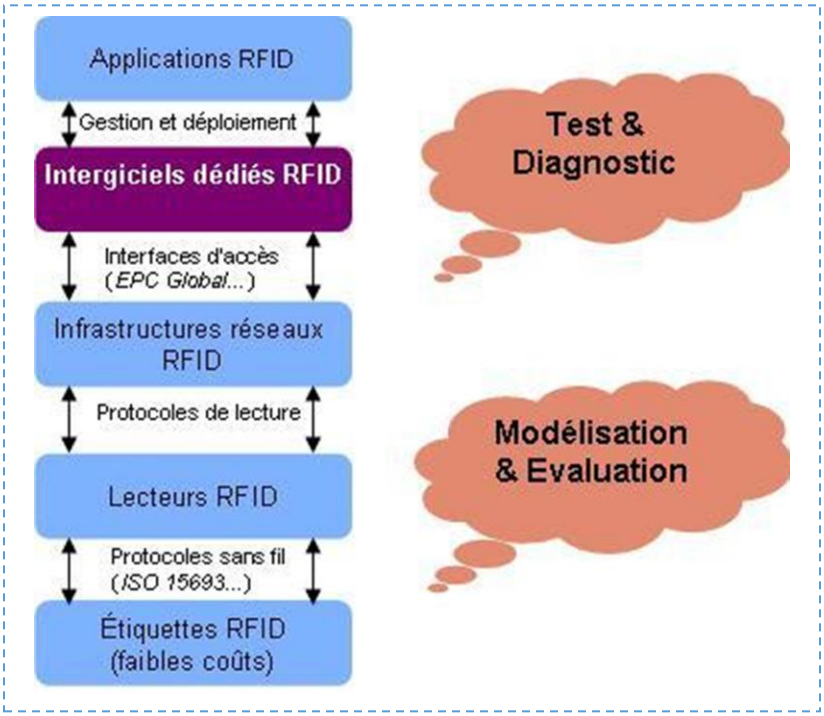


Figure 1. Portée du projet SAFERFID

Trois thèses sont issues de ce projet de recherche et correspondent aux trois niveaux évoqués précédemment ; à savoir, (i) le tag[2], (ii) le lecteur[3] et (iii) le middleware. Ce mémoire présente les travaux de recherche effectués pour améliorer la sûreté de fonctionnement des systèmes RFID au niveau de la couche middleware. L'intérêt de focaliser nos recherches sur la couche « Middleware » est que cette dernière représente la colonne vertébrale des systèmes RFID actuels, les systèmes disposant de plusieurs sources de données (lecteurs RFID, détecteurs de mouvement ou de température, *etc.*) éventuellement distribuées sur plusieurs sites géographiquement éloignés. Le middleware RFID dispose d'une vision globale de tout le système. De ce fait, placer nos solutions de monitoring à ce niveau permettra d'avoir un maximum d'informations pour effectuer un diagnostic précis du système RFID. Ainsi, les approches que nous proposons sont réparties sur la couche « Middleware » et sur l'interface de communication entre le middleware et les sources de données (par où toutes les données transitent). Nous détaillerons

[1] http://www.agence-nationale-recherche.fr
[2] Thèse de M. Omar Abdelmalek sur la conception et le prototypage d'une architecture de tag UHF robuste.
[3] Thèse de M. Gilles Fritz sur la simulation de fautes pour l'évaluation du test en ligne des systèmes RFID.

les solutions proposées dans la section suivante.

III. Organisation du mémoire

Ce document est organisé en quatre parties :

- **Introduction** : cette partie permet d'introduire la problématique de la sûreté de fonctionnement dans les systèmes RFID et le projet de recherche duquel cette thèse est issue.
- **Etat de l'art** : composée des chapitres 1 et 2, cette partie permet d'introduire respectivement les systèmes RFID, les protocoles de communication utilisés, les middlewares RFID et enfin, les mécanismes de tolérance aux fautes existants.
- **Contribution** : cette partie présente les différents mécanismes logiciels développés dans nos travaux pour augmenter la fiabilité des systèmes RFID. Elle se présente sous deux axes comme le montre la figure 2. Le premier axe, traité au chapitre 3, regroupe les solutions de tolérance aux fautes au niveau de la couche middleware du système RFID. Le deuxième axe, traité au chapitre 4, regroupe les mécanismes de tolérance aux fautes au niveau du protocole de communication entre le middleware et le lecteur RFID.

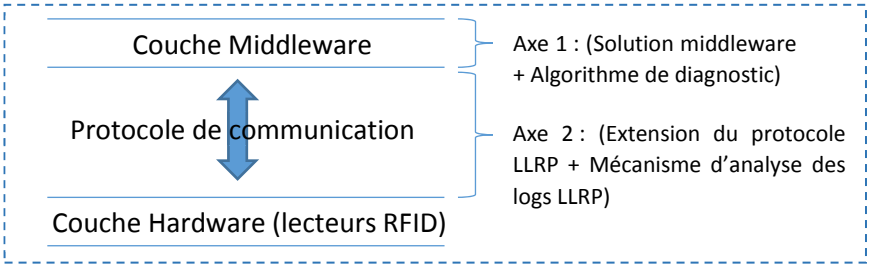


Figure 2. Localisation des solutions proposées dans un système RFID

Pour résumer, la contribution de ce mémoire se présente en un middleware RFID tolérant aux fautes accueillant trois mécanismes de tolérance aux fautes ; (i) un algorithme de diagnostic probabiliste [10] [11], (ii) une extension du protocole de communication pour prendre en compte le manque de précision des lecteurs RFID notamment dans les environnements hostiles [12] et enfin, (iii) un analyseur de fichiers log dont le but est d'extraire la suite d'événements qui a mené le système vers la défaillance et ainsi déduire la cause de cette défaillance [13].
L'algorithme de diagnostic proposé est repris dans un cinquième chapitre pour l'évaluer et discuter ses performances.

- **Conclusion et Perspectives** : cette partie revient sur ce qui a été présenté dans la partie contribution, en apportant un regard critique sur les différents mécanismes proposés et présente quelques perspectives d'amélioration et des pistes de recherche pour approfondir la sûreté de fonctionnement des systèmes RFID.

IV. Liste des publications

- Rafik KHEDDAM, Oum-El-Kheir AKTOUF, Ioannis PARISSIS et Sanaa BOUGHAZI, "Monitoring of RFID Failures Resulting from LLRP Misconfigurations", *21th IEEE International Conference on Software, Telecommunications and Computer Networks* (*SoftCOM*). Split, Croatie, 2013, pp. 1-6.

- Rafik KHEDDAM, Oum-El-Kheir AKTOUF et Ioannis PARISSIS, "SafeRFID-MW: a RFID Middleware with runtime fault diagnosis", *Journal of Communications Software and Systems (JCOMSS), Special issue on RFID Technologies and Internet of Things*. Vol. 9, No. 1, Mars 2013. pp. 57-73.

- Rafik KHEDDAM, Oum-El-Kheir AKTOUF et Ioannis PARISSIS, "Online monitoring and diagnosis of RFID readers and tags", *20th IEEE International Conference on Software, Telecommunications and Computer Networks* (*SoftCOM*). Split, Croatie, 2012, pp. 1-9.

- Rafik KHEDDAM, Oum-El-Kheir AKTOUF et Ioannis PARISSIS, "Algorithme de diagnostic probabiliste pour la détection des défaillances dans les systèmes RFID", *9ème Conférence Internationale Jeunes Chercheurs* (*MajecSTIC*), Lille, France, 2012, pp. 1-8.

- Rafik KHEDDAM, Oum-El-Kheir AKTOUF et Ioannis PARISSIS, "An extended LLRP model for RFID system test and diagnosis", *5th IEEE International Conference on Software Testing, Verification and Validation*. Montréal, Canada, 2012, pp. 529-538.

- Frédéric DADEAU, Pierre-Cyrille HÉAM et Rafik KHEDDAM, "Mutation-Based Test Generation from Security Protocols in HLPSL", *4th IEEE International Conference on Software Testing, Verification and Validation.* Berlin, Allemagne, 2011, pp. 240-248.

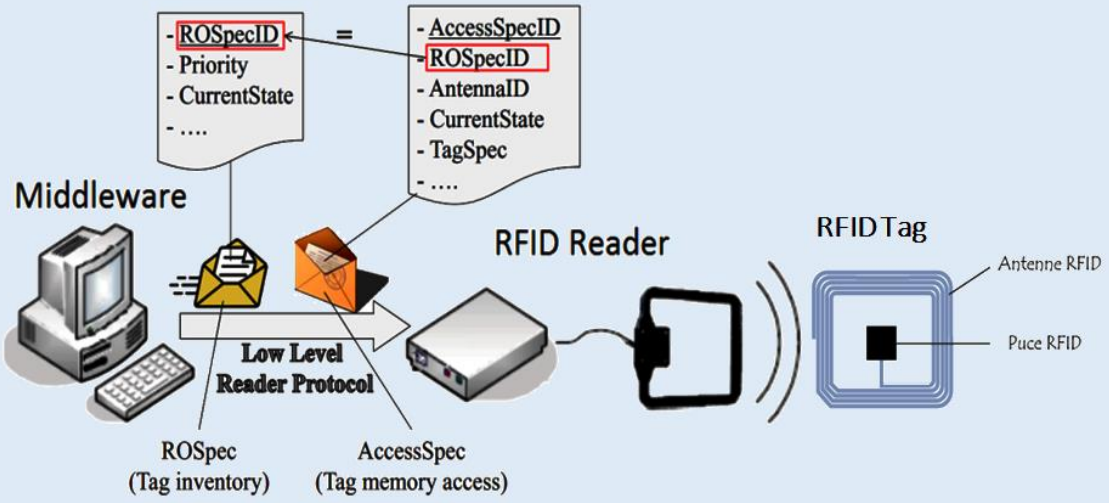
Etat de l'art

Chapitre 1 : Etude des systèmes RFID

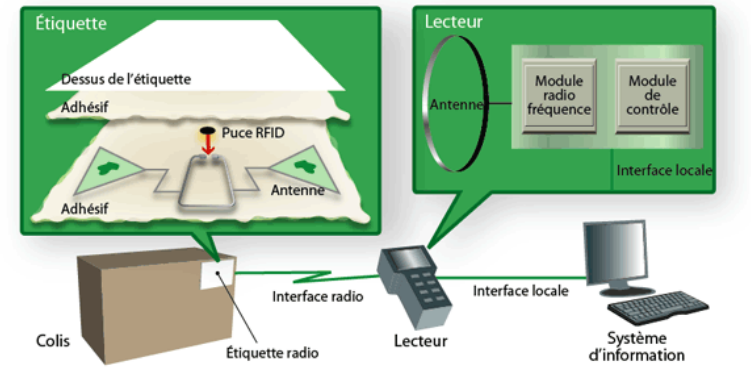
- Présentation générale de la RFID
- Présentation du protocole LLRP

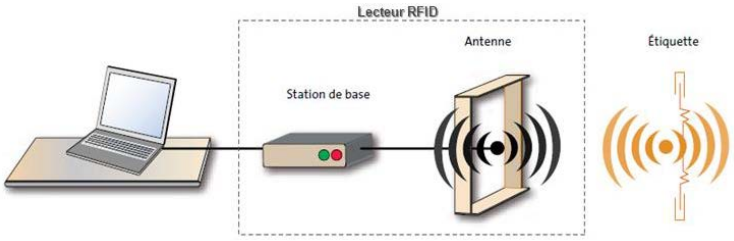
Chapitre 2 : Etude des middlewares RFID

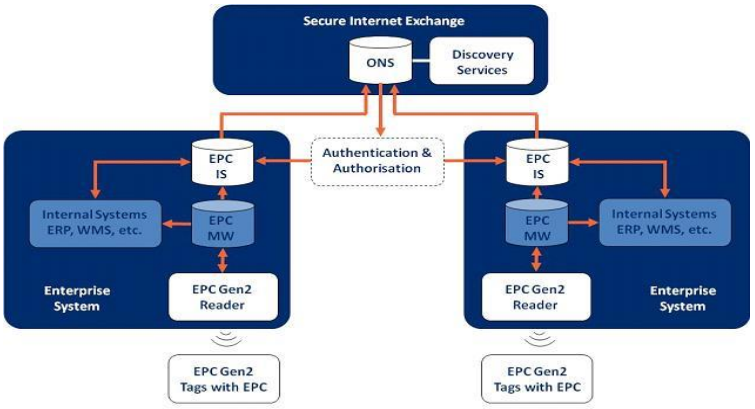
- Middlewares RFID existants
- Techniques de tolérance aux fautes

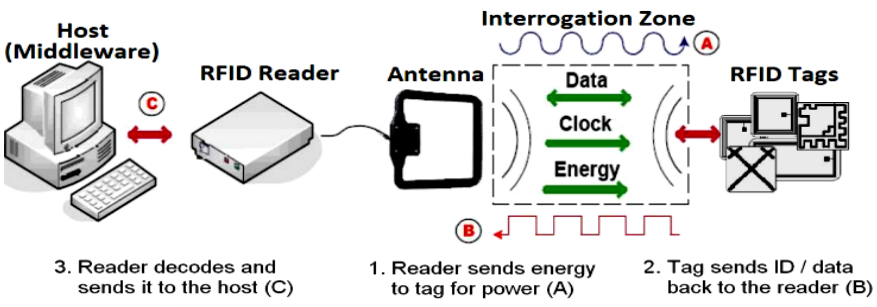


Chapitre 1
Etude des systèmes RFID









Sommaire

Mots Clés

Radio Frequency Identification
Système RFID
Middleware
Tag / étiquette RFID
Lecteur
Station de base
EPCglobal
Electronic Product Code
LLRP
EPC C1 Gen2
Radiofréquence
ROSpec
AccessSpec
RFSurvey

CHAPITRE 1
ETUDE DES SYSTEMES RFID

I. Introduction

Depuis plusieurs années, l'identification par radio fréquence, plus connue sous l'acronyme RFID (Radio Frequency IDentification), s'est répandue dans la vie quotidienne. Autrefois, cette technologie était utilisée exclusivement dans le domaine militaire. Sa première utilisation remonte à la deuxième guerre mondiale [1] [2]. Au début des années 1940s, la « British Royal Air Force » a équipé les avions avec des transpondeurs radio qui répondent quand ils sont interrogés afin de différencier les avions des alliés de ceux des ennemis. Ce système d'identification s'appelait IFF (Identfy Friend or Foe) [14]. La technologie RFID est restée confidentielle [15] et à usage militaire jusqu'à la fin des années 70s. Dans les années 80s, le tag passif (étiquette RFID ou transpondeur) a fait son apparition et a permis d'élargir le champ d'application de la RFID vers des domaines grand public, notamment grâce à son prix de plus en plus bas. La RFID a été commercialisée pour la première fois en Europe et utilisée pour l'identification du bétail [16]. A la fin de l'année 2003, l'organisation EPCglobal[4] voit le jour, pour promouvoir la technologie RFID et les codes EPC (Electronic Product Code) qui sont destinés à identifier de manière unique et universelle différents objets grâce aux étiquettes RFID.

II. Architecture d'un système RFID

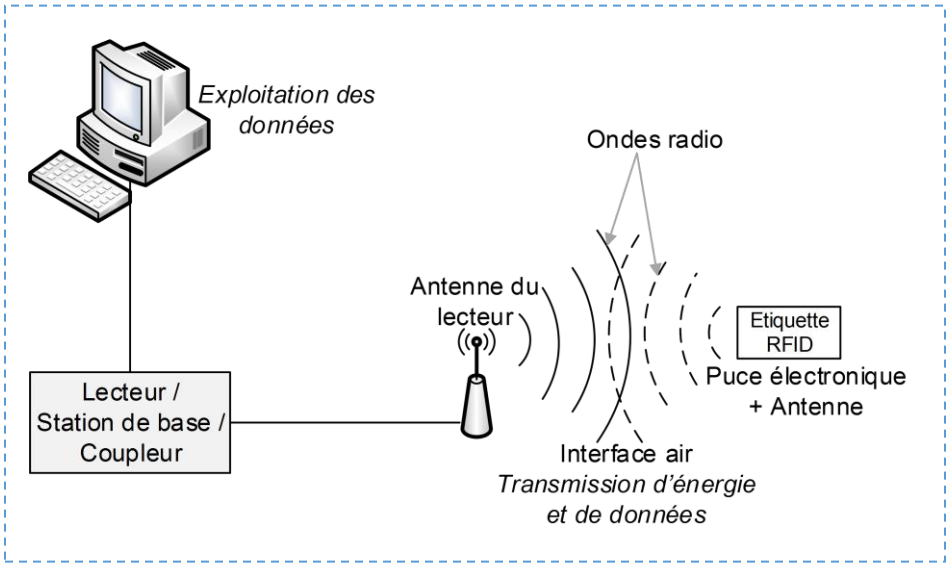


Figure 3. Architecture d'un système RFID

La technologie RFID permet d'identifier plusieurs objets en même temps "sans contact ni vision directe". Elle représente une technologie d'identification semblable à celle du code-barres. Elle est composée de *lecteurs* ou *interrogateurs RFID*, d'*étiquettes* ou *tags* et d'*applications de gestion des données* échangées entre lecteurs et tags grâce à des ondes radio. Récemment, vu l'emploi de la RFID dans des systèmes de plus en plus complexes et distribués, un nouveau composant appelé « middleware

[4] EPCglobal *Inc.* est une compagnie à but non lucratif spécialisée dans le développement de standards de communication pour promouvoir la technologie RFID [www.epcglobalinc.org].

RFID » a fait son apparition pour faire face aux quantités énormes de données qui sont émises par les différentes sources RFID. La figure 3 montre les différents composants d'un système RFID simple.

1. Lecteur/Interrogateur/Station de base RFID

Un lecteur RFID est un dispositif actif permettant la communication avec les étiquettes (tags) RFID grâce à des ondes radio de fréquences spécifiques émises et reçues par ses antennes. Ce dispositif peut être fixe ou mobile avec une ou plusieurs antennes internes (intégrées) ou externes. Il permet d'activer (si besoin) et d'interroger un tag qui passe à proximité de son champ radiofréquence (RF) en lui fournissant éventuellement l'énergie dont il a besoin pour fonctionner (cas des tags passifs et semi passifs). Le tag répond par un signal radiofréquence contenant une information qui identifie l'élément auquel ce dernier est rattaché [17]. Un dialogue s'installe alors, entre ces deux entités suivant des règles bien définies que nous appelons « protocole de communication » ou « interface air » [18].

Un lecteur RFID permet :

- La gestion de la communication avec le tag (activation, ouverture de session, lecture, écriture, autorisation...).
- La gestion du transport des données (fréquence, vitesse de transfert, modulation, puissance d'émission...).
- Le codage, le décodage, le contrôle et le stockage des données.

Les fonctions d'un lecteur RFID peuvent aussi être regroupées en deux tâches principales [17] :

- **Inventaire :** est l'identification de tous les tags présents dans le champ RF du lecteur en utilisant une séquence de commandes d'un protocole de communication spécifique comme le protocole UHF EPC Class 1 Gen 2 [19]. Plusieurs tags peuvent répondre en même temps. L'algorithme de singularisation (algorithme d'anticollision) permet au lecteur d'isoler les tags un à un, afin de pouvoir décoder leurs réponses.
- **Accès mémoire :** le lecteur doit avoir la possibilité d'accéder à la mémoire des tags, de récupérer les données stockées dedans, de les interpréter, ainsi que de les mettre à jour (modification, ajout et suppression d'informations).

2. Etiquette/Tag/Transpondeur RFID

Un tag RFID sert à identifier de manière unique l'élément auquel il est rattaché. Il dispose d'une petite mémoire dans laquelle sont stockés l'identifiant ainsi que le firmware du tag nécessaire à son bon fonctionnement. Il peut aussi disposer d'une mémoire additionnelle pour stocker d'autres informations, mais aussi de capteurs (de température, d'humidité, des détecteurs d'accélération) afin de fournir de nouveaux services à l'utilisateur. Les tags se divisent en trois types [14] :

- **Tag passif :** il s'agit de la forme la plus simple des tags RFID. Un tag passif contient une petite antenne connectée à une puce. Ce tag est passif dans le sens où il ne dispose pas de source d'énergie embarquée [17]. Il récolte l'énergie électromagnétique émise par le lecteur et la convertit en courant continu pour alimenter sa puce. Il utilise la rétrodiffusion du signal (backscattering) pour communiquer avec le lecteur.
- **Tag semi passif :** il dispose d'une batterie pour faire fonctionner sa puce [17], mais il utilise le même principe que le tag passif pour la communication (*i.e.*, backscattering).
- **Tag actif :** il fonctionne grâce à une batterie intégrée qui l'alimente en énergie. Ainsi, il est capable de répondre aux requêtes des lecteurs avec son propre signal RF au lieu de la réflexion du signal du lecteur comme pour les tags précédents. Cette caractéristique fournit de nouvelles fonctionnalités comme la communication tag-à-tag, la possibilité d'intégrer des capteurs pour

la mise à jour dynamique des informations contenues dans la mémoire du tag (*e.g.,* un thermomètre associé à un tag actif pour suivre la chaîne du froid d'un aliment).

3. Applications de gestion des données

Au plus haut niveau de l'architecture d'un système RFID, nous trouvons les applications utilisateur. Ces applications sont chargées de l'exploitation des données RFID suivant la stratégie de l'entreprise. Nous retrouvons parmi ses applications, les ERP (Enterprise Resource Planning), les applications de gestion des chaînes d'approvisionnement (Supply Chain Management), les WMS (Warehouse Management Systems), et d'autres applications à caractère critique comme les applications médicales et de transport.

4. Middleware RFID

Les anciens systèmes n'étaient pas distribués, c'est-à-dire que les données étaient récupérées par une seule application. Mais avec la croissance des besoins des entreprises, il est devenu vital de partager les données de celles-ci avec leurs partenaires et notamment leurs fournisseurs. Afin de répondre à ce nouveau besoin, l'architecture des systèmes RFID a été revue pour y inclure un nouveau composant logiciel ; un middleware ou intergiciel RFID qui exploite l'architecture distribuée de cette technologie et qui permet la coopération entre applications hétérogènes. La figure 4 nous montre la composition en couches de la nouvelle architecture.

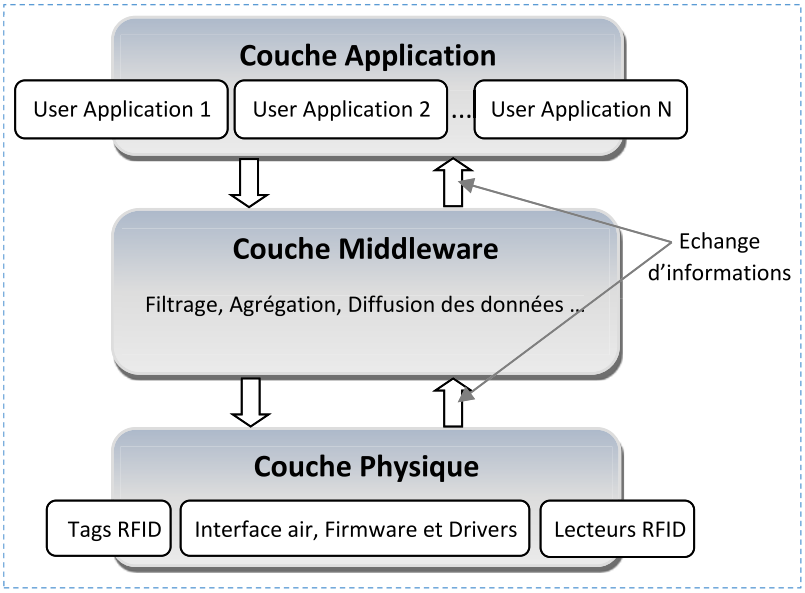


Figure 4. Architecture en couches d'un système RFID

III. Fonctionnement d'un système RFID

1. Fonctionnement général

Le fonctionnement d'un système RFID est donné dans la figure 5. Cette dernière présente un système RFID passif ou semi-passif qui communique en exploitant le principe de la rétrodiffusion [20].

A. Le lecteur RFID interroge le tag et lui fournit éventuellement l'énergie dont il a besoin pour fonctionner par le biais d'ondes radio sur une fréquence déterminée.

B. Le tag convertit une partie du signal reçu en énergie et l'utilise pour répondre au lecteur[5] (cas d'un tag passif ou semi-passif). Un dialogue s'installe alors entre le lecteur et le tag selon un protocole de communication bien défini (*exp.*, EPC C1 Gen2 pour la communication en UHF [19]).

C. Les données lues sont transmises vers l'application utilisateur pour exploitation.

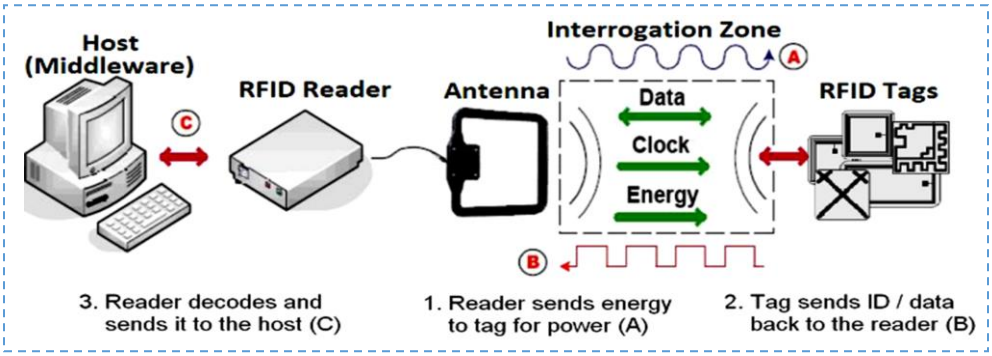


Figure 5. Fonctionnement d'un système RFID

2. Bandes de fréquences des systèmes RFID

Le signal radiofréquence se situe entre le signal électrique et le signal infrarouge. Il couvre un large spectre allant de 3 KHz à 300 GHz (figure 6). Le signal de fonctionnement et de communication entre lecteurs et tags RFID peut avoir différentes fréquences allant de 125 KHz à 5,8 KHz [14] [21]. La table 1 montre les bandes de fréquences de la RFID utilisées suivant les différentes régions du monde. La table 1 montre aussi le mode de couplage utilisé (inductif ou par propagation) que nous détaillerons dans la section III.3.

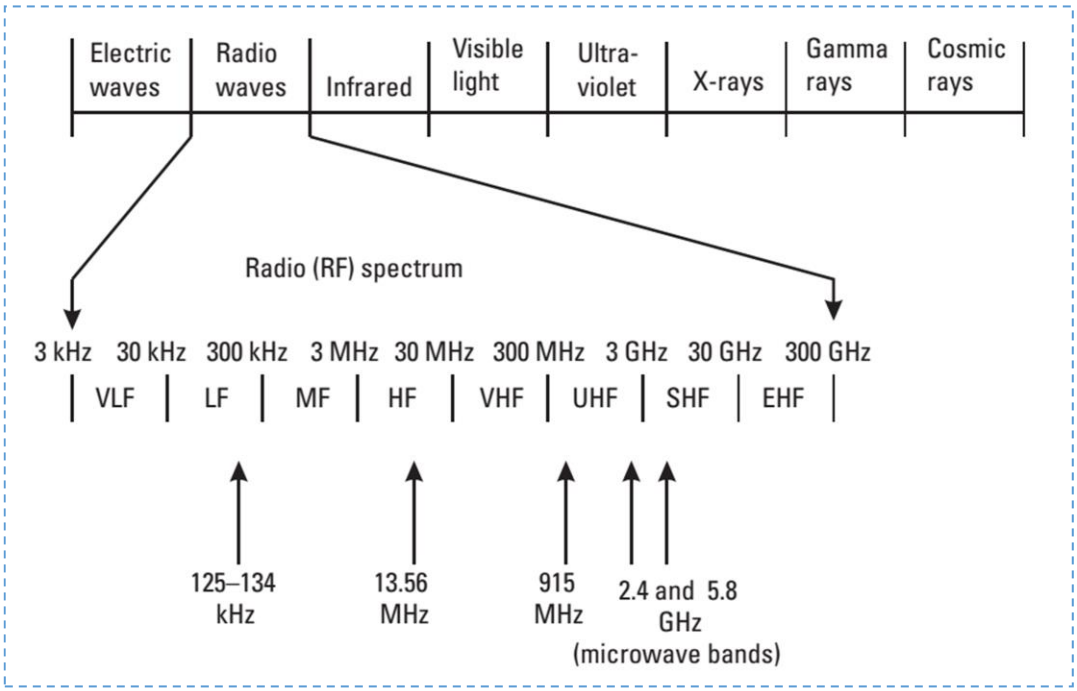


Figure 6. Aperçu des fréquences de la RFID dans le spectre électromagnétique [14]

[5] Les tags actifs utilisent leur propre énergie pour communiquer avec les lecteurs RFID.

- **Basses fréquences** (LF: Low Frequency) : 125 - 134 KHz, Distance de lecture de quelques centimètres.
- **Hautes fréquences** (HF : High Frequency) : 13,56 KHz, Distance de lecture entre 1 cm à 1,5 m.
- **Très hautes fréquences** (VHF : Very-High Frequency) : 433.05 – 434.79 MHz, distance de lecture jusqu'à 3 mètres.
- **Ultra hautes fréquences** (UHF : Ultra High Frequency) : 860 - 960 MHz, ces fréquences supportent de grandes distances de lecture, avec des débits de communication plus élevés, mais sont par contre plus sensibles aux interférences que les basses fréquences.
- **Micro-ondes** ou **Super hautes fréquences** (SHF) : 2,4 - 5,8 GHz, l'avantage de ces fréquences est leur résistance aux perturbations électromagnétiques. Par contre elles nécessitent beaucoup d'énergie (Tags actifs).

NB : le choix de la bande de fréquences dépend essentiellement de l'environnement d'utilisation du système, ainsi que des performances recherchées. Avec l'ultra haute fréquence (UHF), les débits et les distances sont plus élevés. Avec les basses et moyennes fréquences (LF, HF), la sensibilité aux obstacles (le métal ou le liquide) est moins importante.

Table 1. Bandes de fréquences opérationnelles de la RFID

Band	Frequency	System	Regions/Countries
Low frequency (LF)	125 – 134 kHz	Inductive	United States, Canada, Japan, and Europe
High frequency (HF)	13.56 MHz	Inductive	United States, Canada, Japan, and Europe
Very-high frequency (VHF)	433.05 – 434.79 MHz	Propagation	Europe, United States (active tags at certain locations)
Ultrahigh frequency (UHF)	865 – 868 MHz	Propagation	Europe, Middle East, Singapore, Northern Africa
	866 – 869 and 923 – 925 MHz	Propagation	South Korea, Japan, New Zealand
	902 – 928 MHz	Propagation	United States, Canada, South America, Mexico, Taiwan, China, Australia, Southern Africa
	952–954 MHz	Propagation	Japan (for passive tags)
Microwave	2.4 – 2.5 and 5.725 – 5.875 GHz	Propagation	United States, Canada, Europe, Japan

3. Mode de couplage

Le couplage entre le lecteur et le tag peut être [14] [22] :

- Magnétique, appelé aussi couplage inductif. Il concerne généralement les tags passifs. La distance qui peut séparer le lecteur du tag est de quelques cm à 1,5 mètre et cela concerne les systèmes à fréquences basses (LF) ou à hautes fréquences (HF).
- Radiatif, aussi appelé couplage par propagation. Il concerne aussi bien les tags passifs que les tags actifs. Les distances de communication peuvent atteindre 10 mètres et cela concerne les plages de fréquences UHF et micro-ondes.

4. Interférences dans un système RFID

Les performances d'un système RFID peuvent être affectées par plusieurs types d'interférences : interférences *lecteur/tag*, *lecteur/lecteur* [23] [17] ou bien *tag/tag* [24].

- **Interférences lecteur/tag** : elles se produisent lorsque plusieurs lecteurs essaient de lire (donc alimentent) le même tag, ce qui l'empêche d'être lu. En effet, lorsque deux lecteurs ou plus tentent d'identifier le même tag, si la différence entre l'intensité de leurs signaux dépasse le seuil toléré par le tag, alors la lecture de ce dernier ne peut pas avoir lieu. Un tag peut cependant répondre à un des signaux en collision grâce à l'utilisation d'un filtre qui va rejeter certaines interférences. Une différence d'intensité de 6 à 15 dB (marge de tolérance) est nécessaire pour qu'un tag arrive à différencier deux signaux en collision et répondre à un seul lecteur [17].
- **Interférences lecteur/lecteur** : ce genre d'interférences se produit quand un lecteur A se trouve en phase de communication avec un tag, et en même temps capte le signal d'un autre lecteur B qui utilise une fréquence très proche, voire égale à celle de A. si le signal de B est plus fort que celui dudit tag alors, le lecteur A peut ne pas être en mesure de décoder le signal provenant du tag. Ce problème est accentué du fait que la force du signal d'un tag passif vers un lecteur diminue (suivant la distance) plus rapidement que celle d'un lecteur vers lecteur ; *i.e.*, la force du signal lors d'une communication tag/lecteur diminue en fonction de la distance d par «$1/d^4$» contre «$1/d^2$» pour la force du signal lecteur/lecteur [17].
- **Interférences tag/tag** : elles se produisent lorsque plusieurs tags répondent simultanément au signal d'un lecteur. Leurs signaux s'interfèrent (*i.e.*, entrent en collision) et la transmission échoue [24]. Afin de résoudre ces problèmes et de permettre la communication dans un environnement peuplé par plusieurs tags, des méthodes ou des algorithmes d'anticollision (aussi appelés algorithmes de singularisation [17]), sont utilisés par les lecteurs afin d'isoler les tags un par un et ainsi pouvoir lire correctement le contenu de leurs mémoires.

IV. Normes de communication

Dans la technologie RFID passive UHF, il existe deux familles de protocoles de communication « **Reader-Talks-First »** et « **Tag-Talks-First »** [25].

- **RTF** (Reader-Talks-First) ou **ITF** (Interrogator-Talks-First) : le lecteur diffuse un signal d'alimentation, mais les tags présents dans son champ restent inactifs, jusqu'à ce que le lecteur leur envoie une requête d'identification. Le tag concerné répond au lecteur s'il est à la portée de ce dernier. Le lecteur est capable de retrouver les tags avec des identifiants spécifiques en interrogeant tous les tags dont les identifiants commencent par 0 ou par 1. Si plusieurs tags répondent, il demande à ceux dont l'identifiant commence par 01 de répondre et ainsi de suite. Cette recherche est un parcours de l'arbre binaire représentant les tags. Elle est connue sous le nom d'algorithme de singularisation (anticollision). Le protocole UHF Class 1 Gen 2 par exemple fait partie de cette famille.
- **TTF** (Tag-Talks-First) : un tag RFID lorsqu'il entre immédiatement dans le champ du lecteur, lui signale sa présence en réfléchissant le signal reçu et en lui envoyant son identifiant. Le lecteur répond par une brève modulation du signal d'alimentation et le transfert des données peut commencer [26]. Cette technique est très utile pour connaître tous les tags qui passent à proximité du lecteur, notamment les objets transportés par un tapis roulant dans un entrepôt ou un aéroport.

1. Normes ISO

Les normes ISO relatives aux protocoles de communication des différentes bandes de fréquences sont [14]:

- ISO 18000-1 : définit l'architecture de référence et les paramètres à normaliser.
- ISO 18000-2 : paramètres de communication pour les fréquences inférieures à 135 KHz.
- ISO 18000-3 : définit les paramètres de communication pour la fréquence 13,56 MHz.
- ISO 18000-4 : concerne la fréquence 2,45 GHz.
- ISO 18000-5 : concerne la fréquence 5.8 GHz.
- ISO 18000-6 : pour les fréquences situées entre 860 et 930 MHz.
- ISO 18000-7 : pour un fonctionnement en 433 MHz.

2. Norme EPCglobal

EPCglobal *Inc*. est une organisation à but non-lucratif née suite à un accord en juillet 2003, entre l'Auto-ID Center du MIT (*Massachusetts Institute of Technology*) et GS1[6] (connu aussi sous le nom de EAN[7] en Europe et Uniform Code Council pour le GS1 US) [27] [28] [29]. Cette organisation a pour mission le développement et le déploiement des standards EPC (Electronic Product Code) et EPC Network, ainsi que la promotion de la technologie RFID à travers le monde.

2.1. Electronic Product Code (EPC)

Le code EPC est le nouveau moyen d'identification des objets après les codes-barres. Il permet l'identification d'un objet à travers le monde de manière unique. Sa structure est définie par EPCglobal de telle façon qu'il peut être utilisé pour déterminer plusieurs attributs de l'objet portant le code en question. La taille d'un tel code peut être de 64 ou 96 bits (qui sont les plus répandus) ou bien de 128 ou 256 bits. Les attributs d'un code EPC de 96 bits (qui permet théoriquement d'identifier 2^{96} objets) sont représentés sur la figure 7.

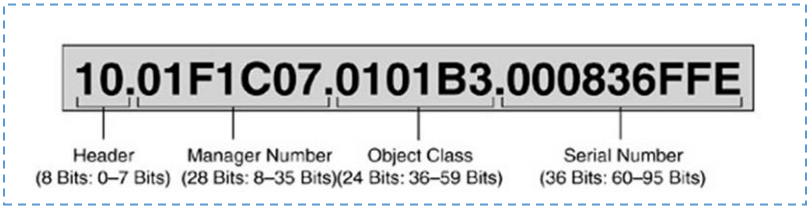


Figure 7. Format d'un code EPC de 96 bits

- **Entête (Header) :** est constitué des 8 bits de poids les plus forts (bit 0 au bit 7) et permet de définir la longueur, le type de clé d'identification, la structure et la version du code EPC.
- **Numéro de gestion EPC (Manager Number) :** (28 bits) identifie le domaine ou l'entreprise (code du fabricant du produit) qui est responsable d'attribuer les données des champs « classe de l'objets » et « numéro de série ».
- **Classe de l'objet (Object Class) :** (24 bits) identifie la classe ou le type des objets (type du produit).
- **Numéro de série (Serial Number) :** (36 bits) identifie l'objet (le produit) au sein d'un même type de produit de manière unique.

2.2. Classification des tags EPC

Avec la norme EPC, les étiquettes RFID sont classées en quatre types [30] :

[6] GS1 est un organisme mondial actif dans le domaine de la normalisation des méthodes de codage utilisées dans la chaîne logistique.
[7] European Article Numbering

- **EPC Class 0 et Class 1** : ces classes concernent les tags passifs qui peuvent stocker des données EPC mais aussi des donnée utilisateurs (en option). Les tags classe 0 sont en lecture seule alors que les tags EPC de classe 1 sont de type « Write Once / Read Many » (une seule écriture, plusieurs lectures). Le protocole **UHF class 1 GEN 2** permet d'accroître les performances de communication (plusieurs centaines de lectures simultanément) mais aussi d'améliorer la sécurité des données grâce à des fonctionnalités de blocage (restriction de l'accès aux données par mot de passe) et de désactivation de l'étiquette avec la fonction « KILL ».
- **EPC Class 2** : cette classe concerne les tags passifs avec une mémoire réinscriptible (read/write memory) qui peut stocker le code EPC ainsi que les données utilisateurs.
- **EPC Class 3** : elle concerne les tags semi-passifs, elle comprend la caractéristique de la classe 2, avec la possibilité d'inclure des capteurs (de température, d'humidité,…) et le log des données.
- **EPC Class 4** : concerne les tags actifs et donne, en plus des caractéristiques de la classe 3, la possibilité de communiquer entre les tags (communication ad-hoc).

2.3. EPC Network

Le réseau EPC est une collection de technologies capable de fournir une identification automatique en temps réel d'objets ainsi que le partage intelligent de données au sein de l'entreprise ou bien vers l'extérieur (ses partenaires) [29]. Ce réseau est conçu dans l'objectif de relier tous les objets dans le monde entier à travers Internet [31]. La figure 8 montre l'architecture de ce réseau.

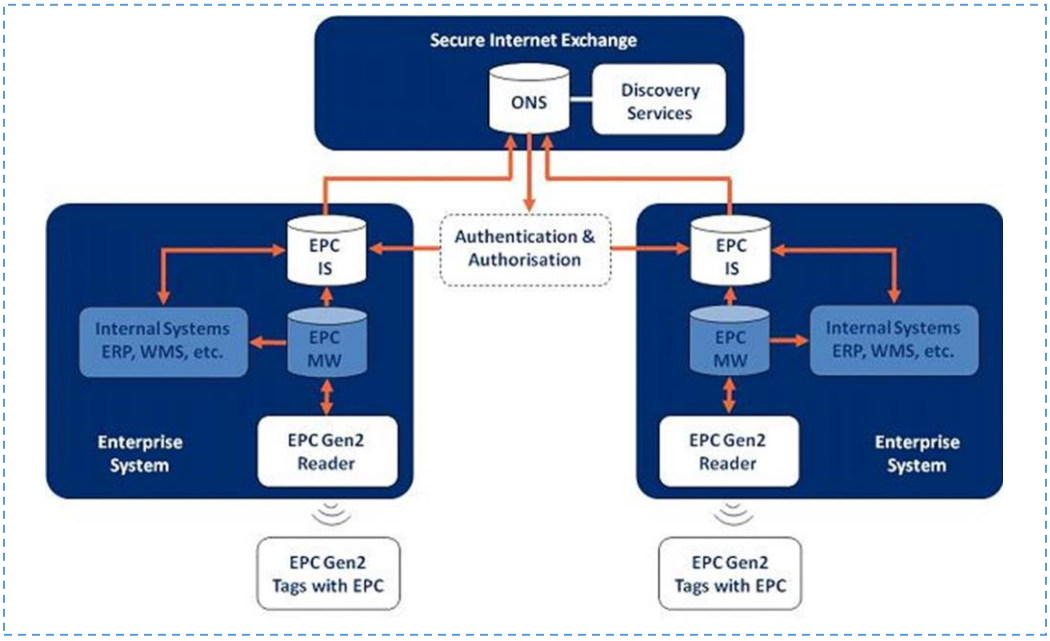


Figure 8. Architecture du réseau EPCNetwork [33]

Les composants d'un réseau EPC coopèrent entre eux pour fournir plusieurs services et répondre aux besoins des applications.

- **Lecteur et tag EPC** : le principe des tags EPC est que seul le code EPC sera stocké dans la mémoire du tag. Celui-ci donne accès aux autres données enregistrées ailleurs dans une base de données donnant plus d'informations sur l'objet identifié.
- **EPC Middleware (EPC MW)** : appelé aussi « Savant » en anglais [31], il a pour rôle de faire des prétraitements sur les données collectées comme le filtrage et la sélection des données susceptibles d'intéresser les entreprises (voir Chapitre 2 pour des exemples de middlewares RFID).

- **EPC Information Services (EPCIS)** : ce composant est à la base des échanges de données pour les entreprises. Il emmagasine toutes les données qui ont un « contexte métier » dans une base de données qui est mise à la disposition du composant « Discovery Services ».
- **Discovery Services (DS)** : il fournit l'accès aux données relatives à un code EPC (avec l'aide de l'ONS décrit ci-dessous) et qui sont stockées dans un EPCIS donné [32]. Dans une chaîne d'approvisionnement, où il y a plusieurs partenaires commerciaux[8], le composant « Discovery Services » peut être considéré comme étant un moteur de recherche [33] responsable de retrouver les données qui concernent un code EPC dans les différents EPCIS des entreprises partenaires. La figure 9 montre un client qui interroge le DS à propos d'un objet dont le code EPC est 112. Le DS lui renvoie alors la liste des serveurs EPCIS dans lesquels l'objet est répertorié (*i.e.*, EPCIS 1 et EPCIS 2).
- **Object Name Service (ONS)** : ce composant est un service public qui fait partie de « Discovery Services ». Il fournit en quelque sorte une table de correspondance entre les codes EPC et les EPCIS correspondants. Il est utilisé pour retrouver le ou les serveurs EPCIS qui gèrent les données d'un code EPC [29] [31]. Ce composant fonctionne comme le système de noms de domaine DNS (Domain Name System) qui établit une correspondance entre une adresse IP et un nom de domaine (comme le domaine « .fr »). L'ONS traduit le code EPC en une adresse internet (URL) qui référence l'EPCIS où se trouvent davantage d'informations concernant l'objet portant le tag EPC.

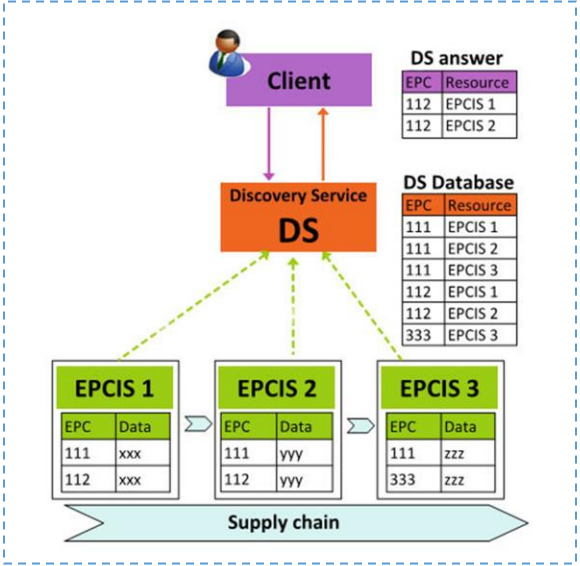


Figure 9. Fonctionnement du Discovery Service

2.4. Protocole EPC Class 1 Gen 2

Le protocole EPC Class 1 Gen 2 est le standard de communication entre les lecteurs et les tags RFID UHF [19]. En 2006, le protocole UHF C1 G2 est reconnu comme un standard ISO/IEC 18000, et porte le nom de ISO/IEC 18000-6 Type C [34]. Ceci a accéléré son adoption par les industriels. Ce protocole se base sur un algorithme de gestion des collisions ou de singularisation appelé « random-slotted anti-collision algorithm » [19] [35].

Lorsqu'un lecteur fait un inventaire des tags, plusieurs de ces derniers peuvent répondre en même temps empêchant le lecteur de décoder correctement le signal reçu. Ce phénomène s'appelle

[8] Chacun des partenaires commerciaux garde des informations relatives aux codes EPC de ses produits dans son EPCIS local.

« collision ». EPC C1 Gen 2 définit un paramètre Q utilisé pour la gestion des collisions. Ce paramètre est envoyé aux tags par le lecteur afin qu'ils choisissent aléatoirement une valeur entre 0 et 2^Q-1 qui s'appelle « Slot Counter » (SC). Seuls les tags qui ont choisi SC=0 répondent au lecteur.

1. Si un seul tag répond au lecteur, le tag est inventorié ; *i.e.*, le lecteur a lu correctement l'identifiant du tag. Alors, le lecteur peut le sélectionner pour récupérer d'autres informations ou il le met en veille afin de ne pas gêner son dialogue avec les autres tags.
2. Si plusieurs tags répondent en même temps, il y a « collision » et le lecteur demande à tous les tags non inventoriés de choisir de nouveau, un autre SC et le cycle recommence.
3. Si aucun tag ne répond, le lecteur demande à tous les tags non inventoriés de décrémenter de 1 leurs SC jusqu'à ce que le SC d'un (voir cas 1) ou de plusieurs tags (voir cas 2) soit égal à zéro.

Ces opérations sont répétées par le lecteur jusqu'à ce que tous les tags soient inventoriés. Il est clair que si la valeur du paramètre Q est trop petite, il y aura beaucoup de collisions. Si Q est grand, il n'y aura pas de collisions ou il y en aura peu, mais en contrepartie, l'identification (l'inventaire) de tous les tags prendra beaucoup plus de temps.

2.5. Protocole LLRP

Afin de supporter et d'accélérer l'adoption de la technologie RFID, et après avoir standardisé le dialogue entre les lecteurs et les tags RFID UHF, EPCglobal a proposé en 2007 le standard LLRP (Low Level Reader Protocol) afin d'harmoniser l'interfaçage des lecteurs avec les systèmes d'information.

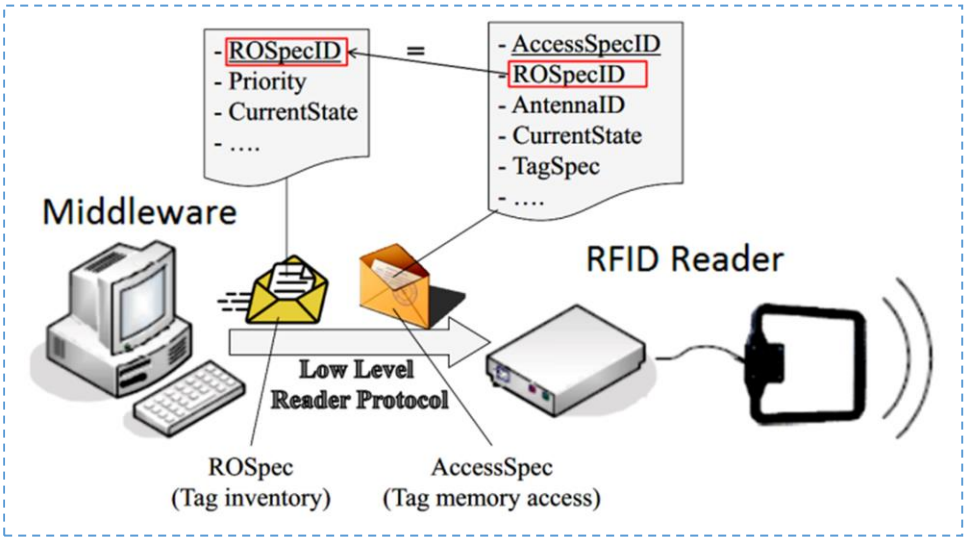


Figure 10. Low Level Reader Protocol

Le protocole LLRP se base exclusivement sur l'échange de messages au format XML (eXtensible Markup Language) [36]. La figure 10 montre les échanges de données entre un client (un middleware) et un lecteur RFID. Cette figure montre l'envoi de deux messages du middleware vers le lecteur. Ces deux messages nommés ROSpec (Reader Operation Specification) et AccessSpec (Tag Memory Specification) représentent les deux principales spécifications du protocole. Elles sont responsables de fournir au lecteur tous les paramètres nécessaires pour faire correctement « l'inventaire des tags[9] » et « l'accès en lecture/écriture aux différentes mémoires des tags sélectionnés ».

[9] Un inventaire des tags est l'identification de tous les tags localisés dans le champ des antennes du lecteur.

2.5.1. Fonctionnement du protocole LLRP

La communication entre les tags et les lecteurs se fait en deux phases. La première est l'inventaire des tags. La deuxième est l'accès aux mémoires des tags sélectionnés. Ce principe de fonctionnement est repris dans le protocole LLRP. LLRP fonctionne en deux parties : la négociation de version du protocole à utiliser dans la communication (LLRP version negotiation) et l'échange de messages (runtime messages). Cette dernière regroupe les deux phases précédemment évoquées sur la communication tag/lecteur à travers les spécifications ROSpec pour l'inventaire et la sélection des tags et AccessSpec pour l'accès aux mémoires des tags sélectionnés. La figure 11 met l'accent sur les deux phases qui composent le fonctionnement du protocole LLRP.

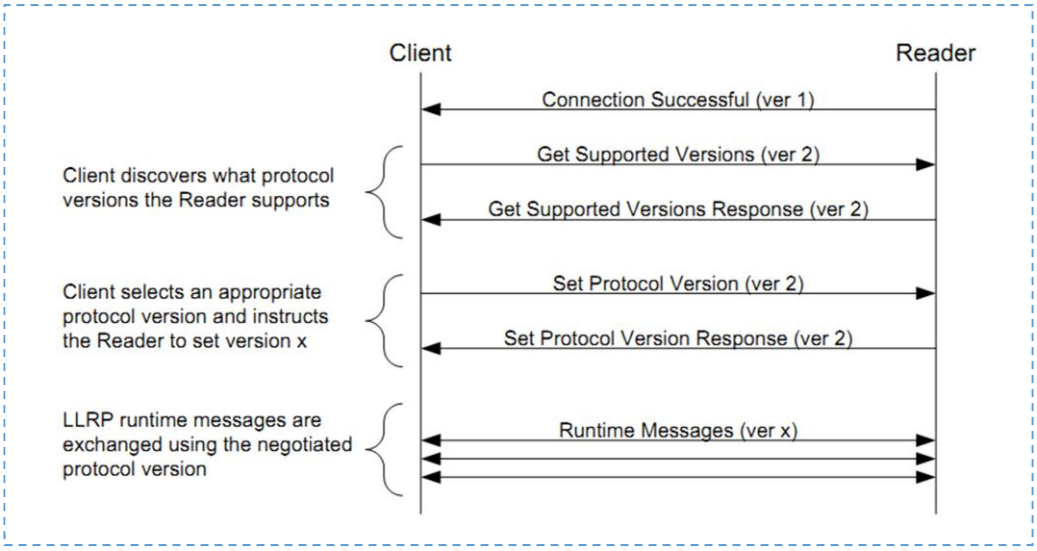


Figure 11. Mise en œuvre du protocole LLRP

La connexion LLRP peut être initiée aussi bien par le lecteur que par le client (le système d'information responsable de la gestion des données collectées). L'établissement de cette connexion entre un lecteur et un client se fait toujours dans la version 1 du protocole LLRP. Ensuite, un message en version 2 du protocole LLRP est envoyé par le client pour demander la liste des versions supportées par le lecteur. Le lecteur répond au client soit en lui fournissant la liste demandée ou bien par un message d'erreur. Dans ce dernier cas, le client présume que le lecteur ne prend pas en charge la version 2 du protocole et que pour toute la durée de la session, l'échange de messages se fera en version 1 du protocole. Dans le cas où une liste est fournie, le client choisit la version la plus récente de la liste que lui prend en charge.

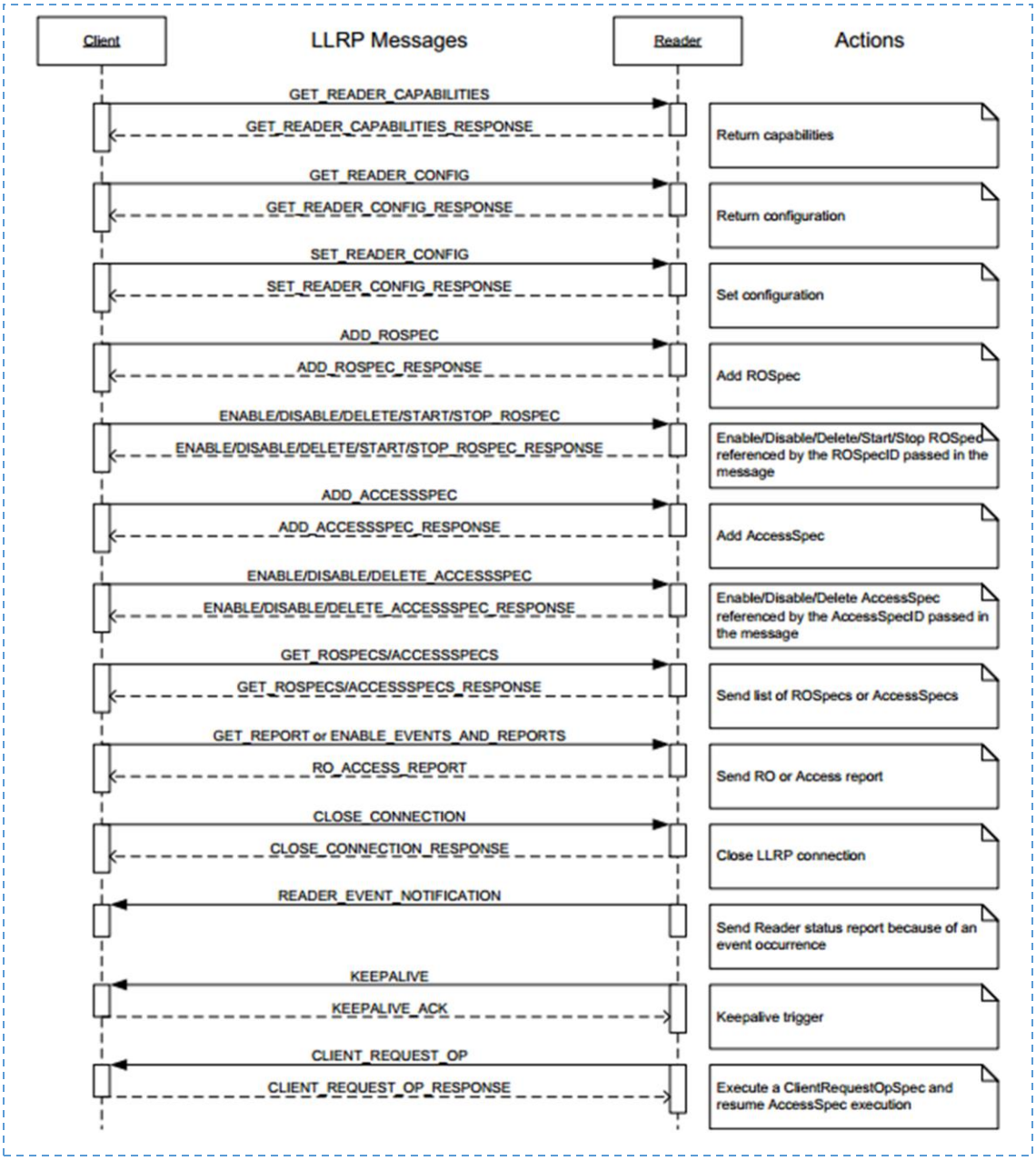


Figure 12. Messages LLRP

Une fois que le lecteur et le client se sont mis d'accord sur la version du protocole à utiliser pour la session en cours, la phase « Runtime messages » vue en la figure 11 peut commencer. Cette phase consiste en plusieurs types de messages échangés entre un client et un lecteur. La figure 12 résume l'essentiel de cet échange. Elle montre un client qui récupère les capacités du lecteur RFID (nombres d'antennes dont il dispose, fréquences supportées, *etc.*) et sa configuration actuelle avant de commencer à le configurer suivant les besoins du client. A chaque message émis par le client correspond un accusé de réception qui l'informe du succès ou de l'échec de l'interprétation du message. Un client

peut ajouter (définir) plusieurs[10] ROSpec et AccessSpec. Chaque ROSpec peut être reliée à une ou plusieurs ROSpec. Une fois que les spécifications LLRP (ROSpec, AccessSpec) sont envoyées aux lecteurs, le client peut émettre les différentes requêtes de manipulation de l'état des spécifications. La figure 13 donne un exemple de requête START_ROSPEC pour lancer l'exécution de la ROSpec dont l'identifiant est 102503.

```
<?xml version="1.0" encoding="UTF-8"?>
<llrp:START_ROSPEC xmlns:llrp="http://www.llrp.org/ltk/schema/core/encoding/xml/1.0"
Version="1" MessageID="0">
  <llrp:ROSpecID>102503</llrp:ROSpecID>
</llrp:START_ROSPEC>
```

Figure 13. Requête de lancement d'une ROSpec

La figure 14 montre le schéma détaillé de l'exécution d'une ROSpec. Nous rappelons qu'une ROSpec est constituée essentiellement d'un ensemble de AISpec (opérations d'inventaire des tags) et de RFSurveySpec (opérations de mesure des niveaux de puissance d'une antenne). A chaque sous-spécification (AISpec ou RFSurvey) est associé un indice appelé « SpecIndex » qui détermine son ordre d'exécution. Une sous spécification spéciale appelée « LoopSpec » peut être définie par le client à la fin des listes des AISpec et RFSurvey. Ce paramètre spécial et optionnel permet de définir le nombre de fois (LoopCount) où la liste des sous spécifications sera exécutée. La figure 14 montre qu'une fois la ROSpec active, le paramètre SpecIndex est initialisé à 1 pour exécuter la première sous spécification. Il peut s'agir d'une AISpec ou d'une RFSurvey. Cette sous-spécification continue de s'exécuter jusqu'à ce que sa condition d'arrêt soit vérifiée ; au quel cas, le SpecIndex est incrémenté pour sélectionner la sous spécification suivante. S'il ne reste plus de sous spécifications, alors la ROSpec se termine et passe dans l'état inactive. Si la sous spécification suivante est une LoopSpec, alors le compteur de boucle LoopCount est décrémenté et le SpecIndex réinitialisé à 1 afin de re-exécuter toute la liste des sous spécifications jusqu'à arriver une nouvelle fois à LoopSpec. Cette boucle est répétée jusqu'à ce que le LoopCount soit égal à 0, et la ROSpec se termine. Lors de l'exécution d'une AISpec, une ou plusieurs AccessSpec peuvent être déclenchées si les tags identifiés passent les filtres (satisfont les conditions « TagSpec ») des AccessSpec. A titre d'exemple, la figure 15 montre un tag RFID avec ses quatre blocs mémoires indexés de 0 à 3. Le bout de code en XML est une partie d'une AccessSpec qui permet de modifier un code EPC de 128 bits organisé en 8 mots mémoires de 16 bits chacun. Cette AccessSpec écrit dans le mot n° 6 du bloc mémoire n° 1 du tag, le code hexadécimal « aabb ».

[10] Le nombre maximal de spécifications qu'un client peut définir est dépendant du lecteur et peut être récupéré avec la requête GET_READER_CAPABILITIES.

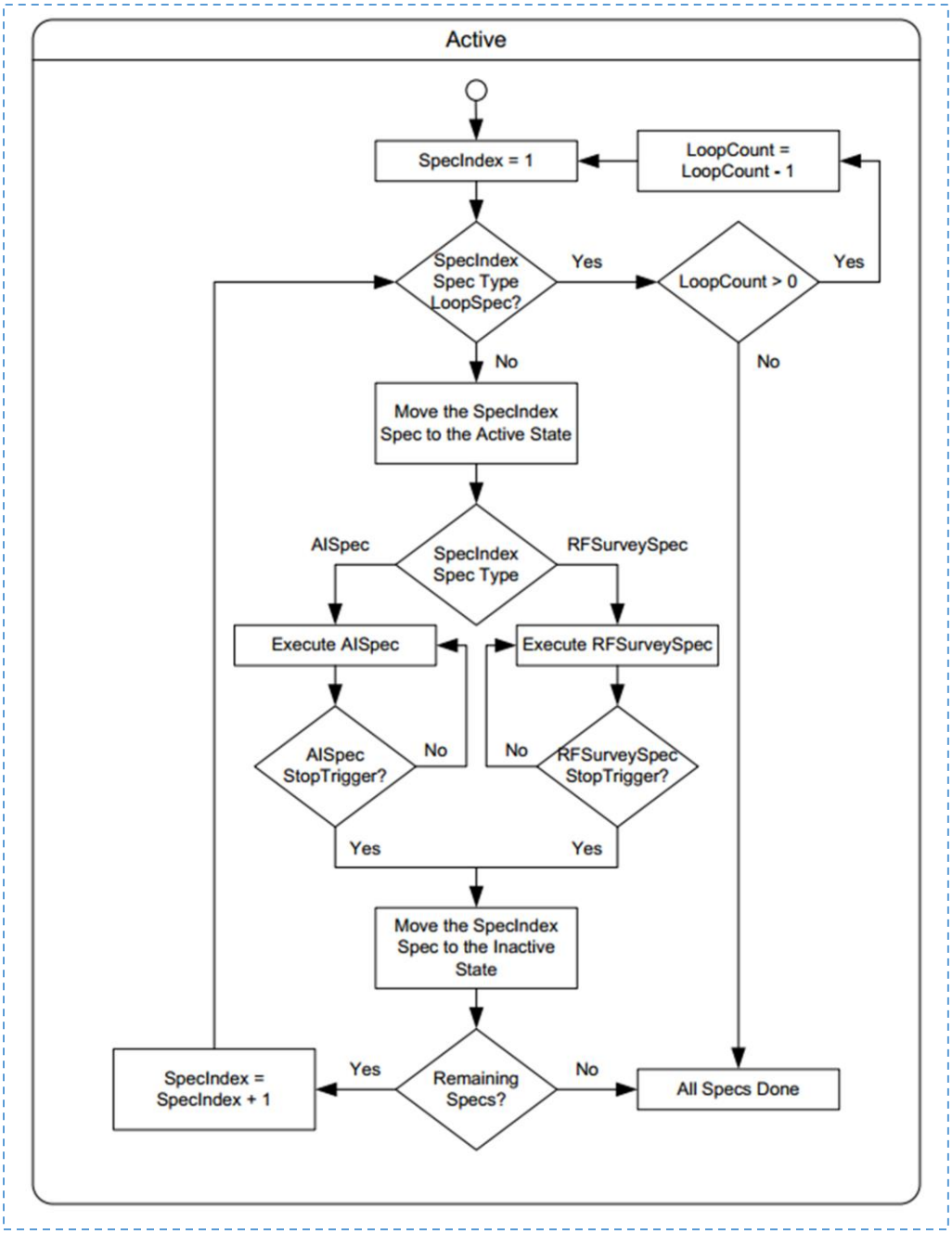


Figure 14. Schéma d'exécution d'une ROSpec

Bout de code d'une AccessSpec

```
<llrp:C1G2Write>
   <llrp:OpSpecID>1234</llrp:OpSpecID>
   <llrp:AccessPassword>0</llrp:AccessPassword>
   <llrp:MB>1</llrp:MB>
   <llrp:WordPointer>6</llrp:WordPointer>
   <llrp:WriteData>aabb</llrp:WriteData>
</llrp:C1G2Write>
```

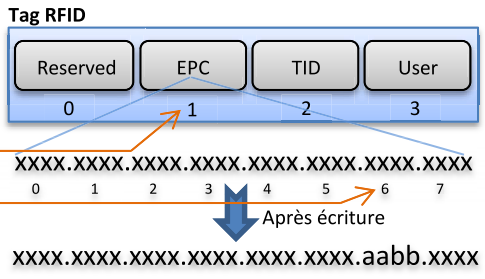


Figure 15. Modification du code EPC d'un tag avec LLRP

Pour clore la description du fonctionnement du protocole LLRP, les points ci-dessous en résument les éléments principaux:

- Une ROSpec regroupe toutes les commandes et paramètres dont un lecteur RFID a besoin pour l'identification et la sélection des tags, ainsi qu'une liste de AccessSpecs à activer une fois les tags sélectionnés (les tags ne sont sélectionnés que s'ils remplissent les conditions d'au moins une AccessSpec).

- Une ROSpec dispose de trois états possibles {Disabled, Inactive, Active}.

- Une AccessSpec dispose de deux états possibles {Disabled, Active}.

- ROSpec active => ROSpec en cours d'exécution.

- AccessSpec active => AccessSpec prête à être exécutée ou en cours d'exécution.

- Une AccessSpec ne s'exécute jamais tant qu'elle n'est pas dans l'état « Active » et qu'elle n'a pas été sélectionnée par la ROSpec en cours d'exécution.

2.5.2. Messages LLRP

La figure 16 montre les échanges de données dans les différents niveaux du système RFID. Dans le premier niveau, l'échange se passe entre les tags RFID et les lecteurs RFID à travers divers protocoles notamment le standard EPC Class 1 Gen 2 pour les communications UHF que nous avons vu précédemment. L'échange de données dans le second niveau concerne les communications entre les lecteurs et le middleware RFID qui se passent à travers le standard LLRP [36]. Ce dernier se base sur des messages XML pour instaurer et maintenir le dialogue du middleware avec et les lecteurs RFID.

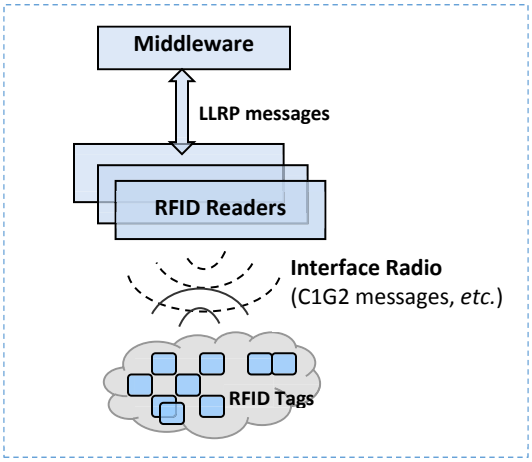


Figure 16. Echange de données dans un système RFID

a) Messages du middleware vers les lecteurs

Cette catégorie regroupe les différentes requêtes du protocole LLRP pour la configuration des lecteurs, l'identification, la sélection et l'accès aux mémoires des tags, ainsi que les *RFSurvey*[11]. Ces messages regroupent *ReaderConfiguration*, *readerCapabilities*, *ROSpec*, *AccessSpec* et tous les différents

[11] RFSurvey est un ensemble d'opérations effectuées par le lecteur pour scanner et mesurer les niveaux de puissance d'une antenne à travers un ensemble de fréquences [79].

messages de manipulations des *ROSpec* et des *AccessSpec* (*e.g.*, *Enable*, *Disable*, *Start*, *Stop Delete*, *etc.*) [36].

- ReaderConfiguration : permet de configurer et d'obtenir les réglages du lecteur (*e.g.*, fréquence de transmission et de réception, types de notifications activées, *etc.*).
- ReaderCapabilities : pour obtenir les capacités du lecteur (*e.g.*, fréquences supportées, interfaces air ou protocoles pris en charge, *etc.*).
- ROSpec : permet de configurer le lecteur pour faire l'inventaire des tags (*e.g.*, conditions de lancement et d'arrêt de l'inventaire, sélection des antennes à utiliser, contenu du rapport…). La figure 17 (a) montre les différents paramètres qui constituent une ROSpec.
- AccessSpec : permet de configurer le lecteur en vue de l'accès aux mémoires des tags (*e.g.*, type d'informations à récupérer, conditions que les tags doivent remplir pour être sélectionnés, *etc.*). La figure 17 (b) montre les différents paramètres qui constituent une AccessSpec.

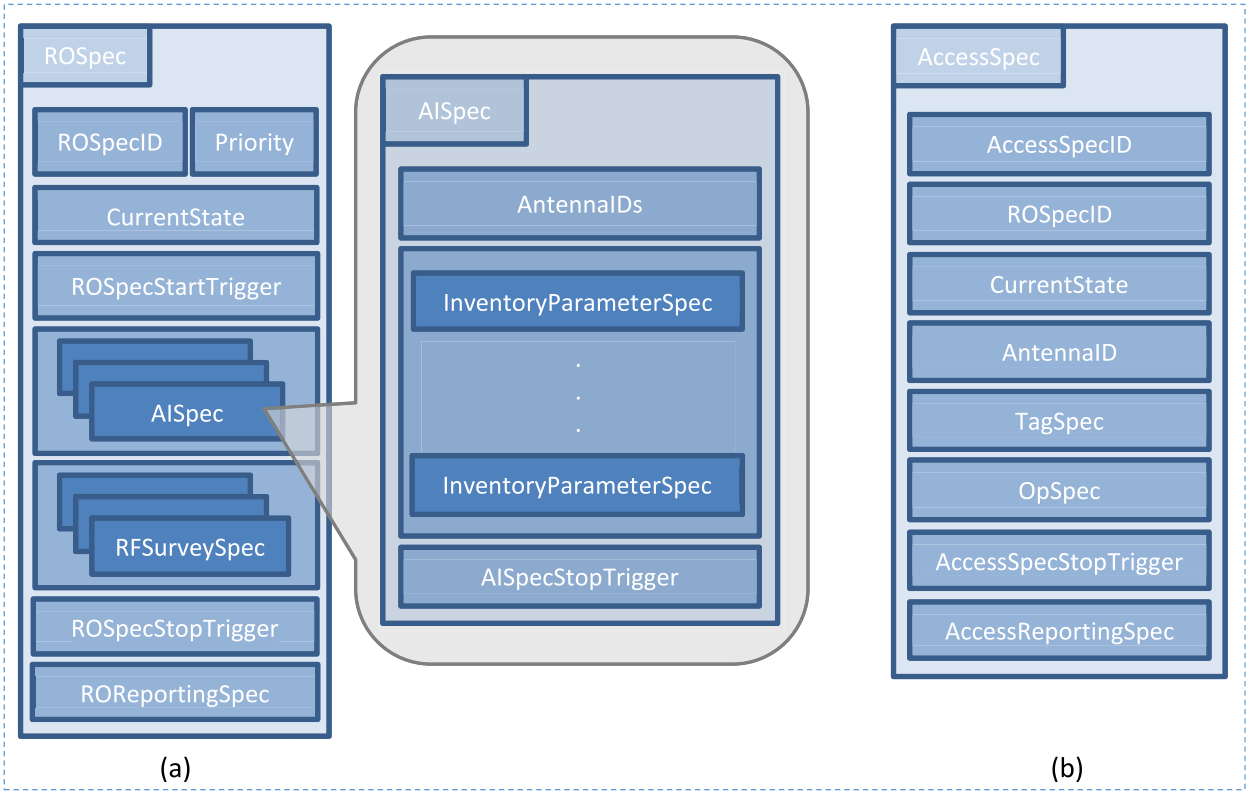


Figure 17. Contenu des spécifications ROSpec et AccessSpec

Une ROSpec est composée principalement des paramètres suivants :

- **ROSpecID** : identifiant de la ROSpec. Ce paramètre permet de manipuler la ROSpec dans le lecteur ; *i.e.*, l'exécuter, l'arrêter, la désactiver, *etc.*
- **Priority** : priorité de la ROSpec qui définit son ordre d'exécution. L'exécution des ROSpec s'inspire beaucoup de la gestion des processus dans un système d'exploitation. Chaque ROSpec dispose d'une priorité et passe par différents états pour s'exécuter avec possibilité de préemption, tout comme les processus dans un système d'exploitation.

- **CurrentState** : état courant de la spécification. Il peut être « Disabled », « Inactive » et « Active ». Chaque nouvelle ROSpec est toujours dans l'état « Disabled ». Elle devient active lorsqu'elle est sélectionnée pour être exécutée par le lecteur.
- **ROSpecStartTrigger** : condition de lancement de la ROSpec. La ROSpec ne sera exécutée que si la condition spécifiée est présente. Ce paramètre peut prendre les valeurs suivantes :
    i. *Null* : La spécification ne commence qu'à la réception de la requête « START » par le lecteur.
    ii. *Immediate* : l'exécution de la spécification commence dès qu'elle devient active.
    iii. *Periodic* : ce paramètre indique une date périodique à laquelle la ROSpec sera exécutée ; *e.g.,* « lancer la ROSpec chaque jour à 8h00 ».
    iv. *GPI* (General Purpose Input) : ce paramètre permet notamment de spécifier un évènement externe qui déclenche l'exécution de la ROSpec ; *e.g.*, sur réception d'un signal à partir d'un détecteur de mouvement.
- **AISpec** (Antenna Inventory Specification): une ROSpec peut contenir une ou plusieurs AISpecs. Une AISpec constitue l'élément principal de la ROSpec. Elle regroupe tous les paramètres nécessaires au lecteur pour effectuer un inventaire des tags. Toutes les AISpecs d'une même ROSpec sont dans l'état « Inactive» et seule la AISpec en cours d'exécution est dans l'état « Active ». Les paramètres principaux d'une AISpec sont :
    - AntennaIDs : identifiants des antennes sur lesquelles l'AISpec va s'exécuter.
    - InventoryParameterSpec : ce paramètre permet de spécifier le protocole air à utiliser[12], la sensibilité du récepteur (Receiver Sensitivity[13]) la puissance du signal, le canal de transmission à utiliser, *etc.*
    - AISpecStropTrigger : condition d'arrêt de la AISpec. Il peut s'agir d'une durée, d'un nombre fixe de tags à identifier, d'un nombre de tentatives de lecture ou tout simplement à l'arrêt de la ROSpec. Nous reviendrons sur ce paramètre dans le chapitre 4 pour montrer que s'il est mal réglé, il peut altérer le comportement normal du lecteur.
- **RFSurveySpec** : ce paramètre constitue une opération dans laquelle le lecteur scanne et mesure les niveaux de puissance du signal supportés par une antenne à travers un ensemble de fréquences. Ses principaux paramètres sont :
    - RFSurveySpecID : identifiant de la RFSurvey.
    - AntennaIDs : une liste des antennes sur lesquelles s'exécute la RFSurvey.
    - StartFrequency : valeur en KHz de la fréquence minimale à utiliser.
    - EndFrequency : valeur en KHz de la fréquence maximale à utiliser.
    - RFSurveySpecStopTrigger : condition d'arrêt de la RFSurvey. Elle peut être une certaine durée ou un nombre d'itérations sur la bande de fréquences spécifiée.
- **ROSpecStopTrigger** : ce paramètre spécifie la condition d'arrêt de la ROSpec. Cette condition peut être :
    i. *Nulle* ; dans ce cas, l'exécution de la ROSpec s'arrête quand toutes les AISpecs et les RFSurveys sont terminées.
    ii. *Durée* ; une durée en millisecondes au bout de laquelle, la ROSpec sera arrêtée.
    iii. *Requête du client* ; à la réception de la requête « STOP_ROSPEC » du client (de l'utilisateur).

[12] Dans la version actuelle du protocole LLRP (LLRP version 2), seul le protocole UHF EPC class 1 Gen 2 est pris en charge.

[13] Receiver Sensitivity (sensibilité du récepteur) indique quel est le niveau minimal du signal qui peut être reçu correctement par un récepteur (un lecteur) [80]. Une forte sensibilité implique une meilleure réception des signaux faibles.

- **ROReportingSpec** : ce paramètre définit le contenu du rapport et à quel moment il faut l'envoyer.

Une AccessSpec est composée principalement des paramètres suivants :

- **AccessSpecID** : identifiant de la AccessSpec.
- **ROSpecID** : identifiant de la ROSpec responsable de déclencher cette AccessSpec. Si cet identifiant est égal à zéro, alors Cette AccessSpec est associée à toutes les ROSpecs du lecteur. Cela veut dire que toutes les ROSpecs sont susceptibles de déclencher cette AccessSpec.
- **CurrentState** : état courant de la AccessSpec. Une AccessSpec peut être soit « Disabled » ou « Active ». Lorsqu'une AccessSpec est active, on dit qu'elle est prête à s'exécuter, à l'inverse de la ROSpec qui commence son exécution lorsqu'elle devient active. La figure 18 montre les différentes transitions d'une AccessSpec.

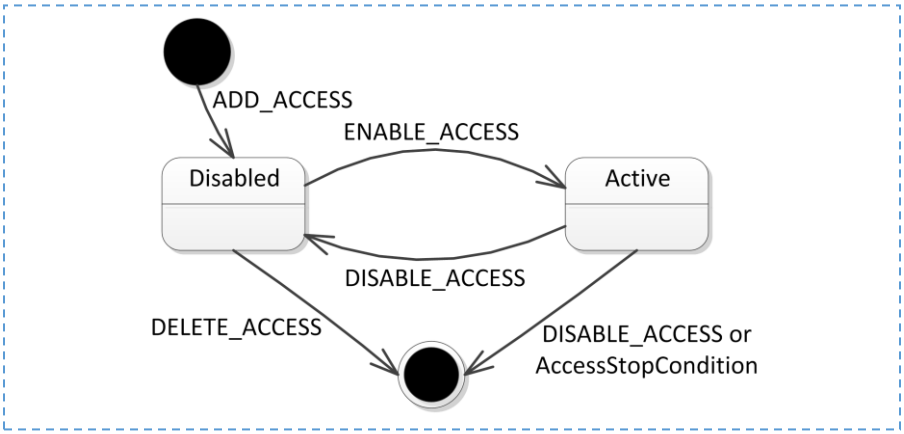


Figure 18. Diagramme d'états-transitions d'une AccessSpec

- **AntennaID** : liste des antennes sur lesquelles la AccessSpec sera exécutée.
- **TagSpec** : conditions que les tags identifiés doivent remplir pour que cette AccessSpec soit exécutée sur eux. Ce paramètre spécifie dans quelle mémoire {mémoire EPC, mémoire TagID, mémoire utilisateur, mémoire réservée} extraire les données et le pattern (TagMask) auquel les données seront comparées. Si les données d'un tag coïncident avec le pattern, ce dernier est sélectionné pour que l'AccessSpec soit exécutée sur lui.
- **OpSpec** : spécifie l'ensemble des données et des requêtes du protocole utilisé dans l'interface air pour effectuer correctement les opérations de lecture, écriture, verrouillage, *etc.*
- **AccessSpecStopTrigger** : condition à laquelle la AccessSpec sera automatiquement supprimée. Ce paramètre peut avoir soit la valeur nulle (la AccessSpec ne sera pas supprimée) ou bien une valeur qui spécifie le nombre de fois où la AccessSpec sera exécutée avant d'être supprimée.
- **AccessReportingSpec** [optionnel] : ce paramètre définit à quel moment il faut envoyer le rapport. Il peut être déclenché soit avec l'envoi du rapport de la ROSpec ou bien à la fin de l'exécution de la AccessSpec.

b) Messages du lecteur vers le middleware

Cette catégorie regroupe essentiellement les réponses aux requêtes du middleware. Il peut s'agir de l'état du lecteur, des accusés de réception, des messages d'erreurs et de notification, ainsi que des résultats d'inventaire ou d'accès aux mémoires des tags. La figure 19 montre un exemple de message envoyé par le lecteur vers le client suite à une tentative de connexion LLRP par le client. Ce message horodaté informe le client que sa tentative de connexion est un succès.

```
<?xml version="1.0" encoding="UTF-8"?>
<llrp:READER_EVENT_NOTIFICATION xmlns:llrp="http://www.llrp.org/ltk/.../encoding/xml/1.0"
Version="1" MessageID="0">
  <llrp:ReaderEventNotificationData>
    <llrp:UTCTimestamp>
      <llrp:Microseconds>2013-05-13T16:37:31.200+02:00</llrp:Microseconds>
    </llrp:UTCTimestamp>
    <llrp:ConnectionAttemptEvent>
      <llrp:Status>Success</llrp:Status>
    </llrp:ConnectionAttemptEvent>
  </llrp:ReaderEventNotificationData>
</llrp:READER_EVENT_NOTIFICATION>
```

Figure 19. Message de notification du lecteur

2.5.3. Modélisation du protocole LLRP

Nous avons traduit la description textuelle du protocole LLRP en une machine à états finis de 17 états et plus de 200 transitions (Voir annexe F). Cet automate nous sera utile pour vérifier si la solution middleware que nous proposons et qui sera présentée dans les prochains chapitres est conforme aux spécifications de EPCglobal *Inc*. Cette machine à états finis sera exploitée pour proposer une extension du protocole LLRP dans le but d'améliorer sa fiabilité ; En d'autres termes, Cette modélisation sera utilisée pour le diagnostic et le test à base de modèles que nous verrons par la suite.

Soit *G* la machine à états finis de LLRP. $G = (S, I, O, \delta, \lambda)$; où *I*, *O* et *S* sont respectivement un ensemble fini de symboles d'entrée, un ensemble fini de symboles de sortie et un ensemble fini d'états.

- $\delta : S \times I \to S$ est la fonction de transition d'états.
- $\lambda : S \times I \to O$ est la fonction de sortie.

Quand un lecteur RFID ou un middleware est dans un état *s* de *S*, et reçoit une entrée *a* de *I*, il produit une sortie spécifiée par $\lambda(s,a)$ et passe vers un état spécifié par $\delta(s,a)$.

Les labels des transitions dans l'automate LLRP sont sous la forme « X/Y », où X représente les messages (généralement des requêtes) du middleware vers le lecteur, et Y représente généralement la réponse du lecteur au message X du middleware.

Chaque session de communication du protocole LLRP se fait toujours en deux phases comme mentionné précédemment.

a) LLRP version negotiation

Le middleware et le lecteur négocient la version du protocole LLRP à utiliser pour la session courante. Cette négociation commence une fois que la connexion[14] entre le client (le middleware) et le lecteur a été établie. Il faut savoir qu'un lecteur ou un middleware RFID peut ne pas implémenter toutes les versions du protocole LLRP. Alors, il devient nécessaire que le lecteur et le client négocient et se mettent d'accord sur une version commune du protocole LLRP. La figure 20 montre la machine à états finis représentant la phase de négociation et les tables 2 et 3 montrent respectivement les différents états et les transitions qu'un lecteur RFID empreinte durant cette phase de négociation.

[14] Cette connexion peut être initiée aussi bien par le middleware que par le lecteur RFID.

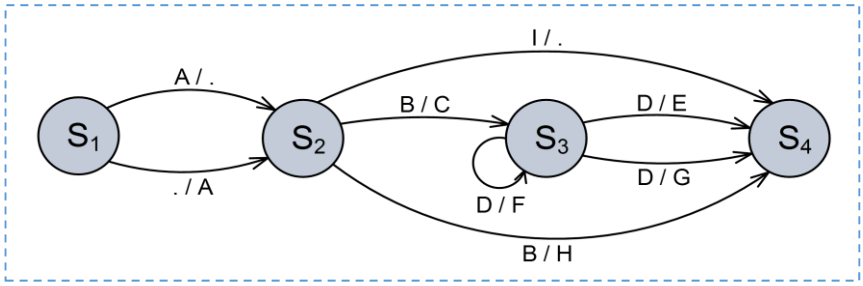


Figure 20. Négociation de la version de LLRP à utiliser

Table 2. Ensemble des états possibles du lecteur lors de la négociation de la version de LLRP

State	*Meaning*
S_1	The reader is disconnected from the middleware.
S_2	The connection is established between the middleware and the reader.
S_3	The middleware received the list of LLRP versions supported by the readers, (the state of the reader is still unchanged).
S_4	The middleware and the reader agree on the version of LLRP to use.

Table 3. Ensemble des transitions possibles lors de la négociation de la version de LLRP

Transition triggering condition	*Meaning*
A	LLRP connection (always with version 1).
B	Getting protocol supported versions.
C	Lists of supported LLRP versions.
D	Setting protocol version (the middleware chooses the most recent version in the list given by the reader, unless it does not support it, and informs the reader).
E	Reader response about the chosen version.
F	Unsupported message by the reader.
G	Unexpected message by the reader (the reader has received a message other than the expected one, and then informs the middleware).
H	Error message that indicates the received message is in an unsupported version.
I	This is not a request or a command but just an indication that the middleware implements only the first version of LLRP, so there is no version negotiation.
.	This means “no request or response”.

b) Runtime messages

Cette phase représente la principale phase du protocole LLRP dans laquelle le middleware demande au lecteur d'identifier, de sélectionner, de lire et d'écrire dans les mémoires des tags. A titre d'exemple, la figure 21 montre le comportement d'un lecteur qui dispose d'une seule ROSpec. L'automate complet de LLRP est donné en annexe F.

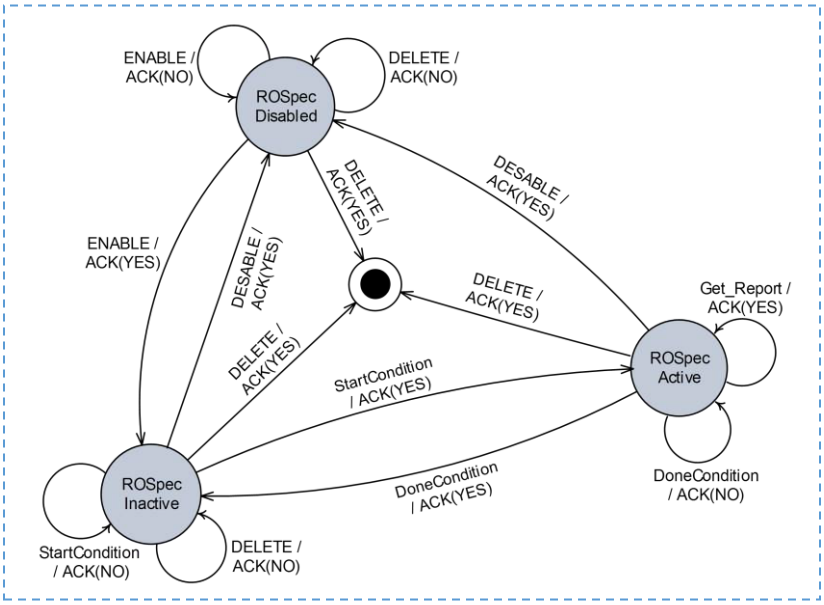


Figure 21. Comportement d'un lecteur avec une seule ROSpec

Une nouvelle spécification (dans notre cas, il s'agit d'une ROSpec) est toujours dans l'état « Disabled » et devient « Inactive » quand le lecteur reçoit correctement la commande « ENABLE » du middleware. La ROSpec reste inactive attendant son tour ou bien la réalisation de sa condition d'activation. Cette condition peut être : « la priorité de la ROSpec est plus haute que celles des autres ROSpec », « un événement extérieur (comme un événement d'un détecteur de mouvement) », *etc.* Le lecteur peut renvoyer les résultats de lecture une fois que la ROSpec est active. Une ROSpec retourne vers l'état « Inactive » quand l'inventaire est terminé ou dans le cas où une autre ROSpec avec une plus haute priorité se présente (préemption). La ROSpec peut aussi être désactivée ou supprimée à la demande du middleware.

V. Avantages et inconvénients de la RFID

1. Avantages de la RFID à travers des exemples d'application [37] :

Parmi les avantages de la RFID, nous pouvons citer

- Opération de prêt/retour facilitées : l'apposition de tags RFID sur des livres ou du matériel à louer accélère la procédure de location et permet de s'assurer que l'objet restitué est bien l'objet loué.
- Contrôle de la chaîne du froid : se fait grâce aux tags actifs et aux capteurs de température intégrés qui permettent d'avoir la température d'un produit de sa production à sa

Figure 22. Amélioration de la sécurité des employés grâce à la RFID

consommation afin de vérifier qu'il a bien été soumis aux conditions de température prescrites.

- Sécurité des individus : pour la localisation des enfants dans un parc d'attraction par exemple, ou bien pour la sécurité des ouvriers dans les chantiers (Figure 22).
- Élevage et suivi vétérinaire des animaux : la RFID peut être utilisée à des fins de marquage.
- Protection des produits contre le vol : grâce à la propriété de l'identification à distance et sans vision directe, les tags RFID vont remplacer les codes-barres, afin de réduire les vols ou encore accélérer la procédure de paiement en caisse (Figure 23).
- Transport et péage routiers : l'utilisation de la RFID dans les cartes de transport facilite les transactions dans les points de contrôle et fluidifie la circulation.

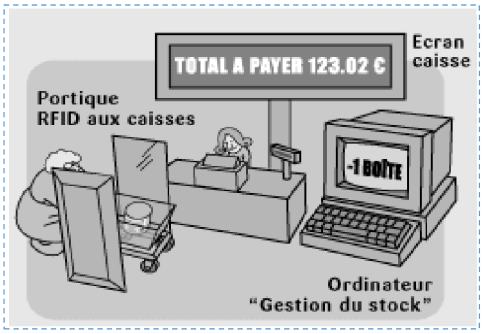


Figure 23. La RFID dans les supermarchés [37]

- Amélioration de la chaîne d'approvisionnement ;
  - Chaque produit se voit associé un code EPC unique, ce qui permet une gestion des stocks plus précise, fiable et rapide : plusieurs produits peuvent être inventoriés simultanément, à distance et sans vision directe du tag.
  - Récupération des informations en temps réel : dès que le produit change de statut (en cours de transport, arrivé à l'entrepôt, vendu au client…), l'information peut être mise à jour dans la chaîne de suivi d'approvisionnement.
  - Lutte contre la contrefaçon : les produits avec étiquettes RFID sont plus difficiles à contrefaire que les codes bar.
  - Réduction de l'intervention humaine (réduction des erreurs humaines, de temps et des coûts associés au suivi des produits)
- Contrôle d'accès : régularisation des accès dans les grandes entreprises ou laboratoires suivant les droits d'accès de chaque employé.
- Amélioration de la fiabilité des systèmes de manutention de bagages dans les aéroports. La figure 24 montre l'utilisation de la technologie RFID pour la gestion et l'aiguillage des bagages dans l'aéroport International HKIA de Hong Kong. La table 2 montre l'amélioration de la fiabilité et la précision des systèmes de manutention de bagages équipés en RFID. Cette précision monte à 97% avec de la RFID contre 80% pour les systèmes à codes-barres [38] et cela en utilisant 4 antennes RFID sur chaque point de contrôle comme montré sur la figure 24.

Table 4. Comparaison de l'efficacité de la RFID et des codes-barres dans l'aéroport HKIA

	Barcode-only Baggage Tag	Integrated RFID Baggage Tag
Read Rate	**80% in average**	**97%**

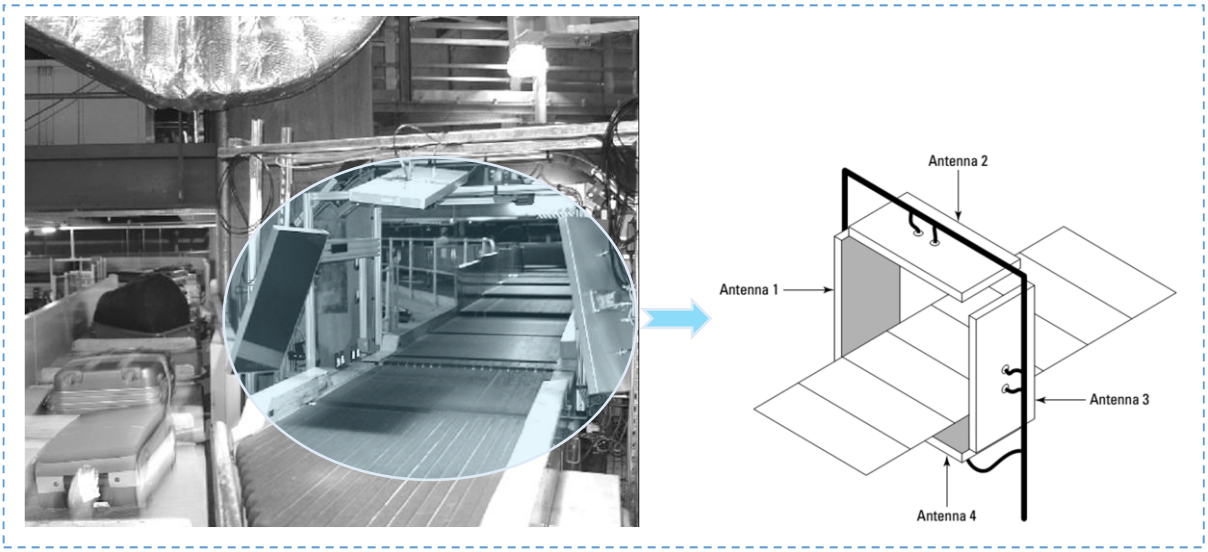


Figure 24. Système de manutention de bagages équipé en RFID à l'aéroport HKIA

2. Inconvénients de la RFID :

Parmi les inconvénients de la RFID, nous pouvons citer

- Coût plus élevé par rapport aux codes-barres.
- Norme globale inexistante (chaque pays utilise ses propres fréquences).
- Perturbations de la transmission en présence de liquide ou de métal ; ce qui peut fausser ou empêcher la lecture.
- Interférences Lecteur/Tag, Lecteur/Lecteur ou Tag/Tag qui peuvent rendre plus difficile ou même empêcher la lecture des tags.
- Problème d'atteinte à la vie privée : la RFID peut être utilisée à des fins néfastes pour les utilisateurs (localisation des individus à leur insu, analyse de leurs habitudes à des fins commerciales).
- Risque lié à la santé (risque de cancer à cause de l'exposition fréquente aux ondes radio).

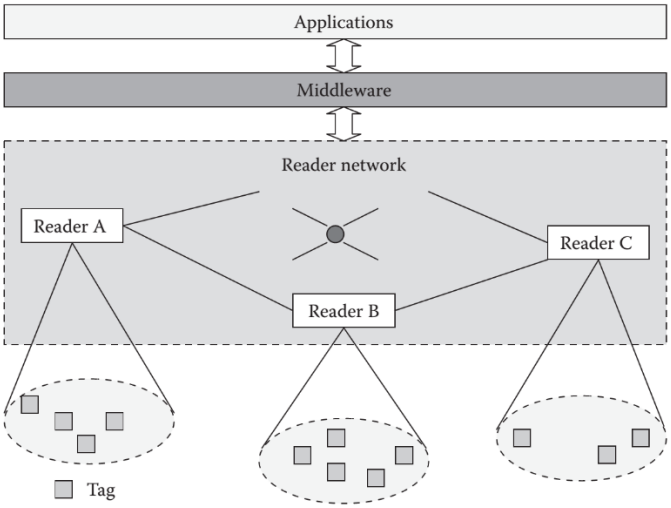
VI. Conclusion

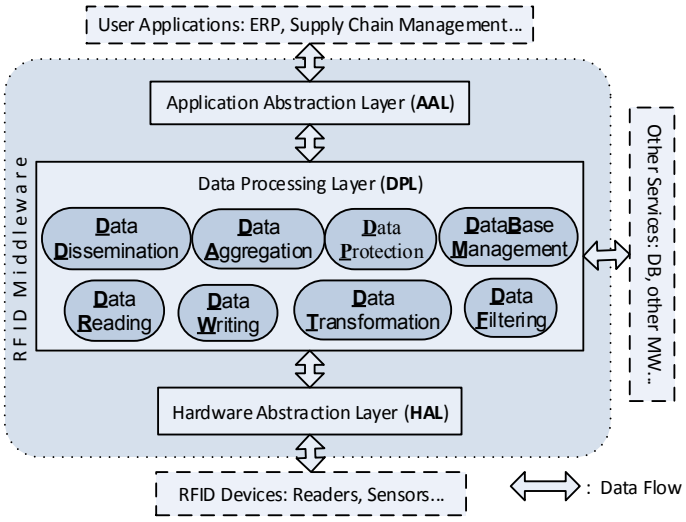
Nous avons vu dans ce chapitre l'architecture des systèmes RFID puis leur fonctionnement général. Ensuite, nous avons présenté les normes utilisées dans les systèmes RFID, notamment le standard LLRP utilisé pour la communication entre les middlewares et les lecteurs RFID. Enfin, nous avons présenté quelques avantages et inconvénients de la technologie RFID.

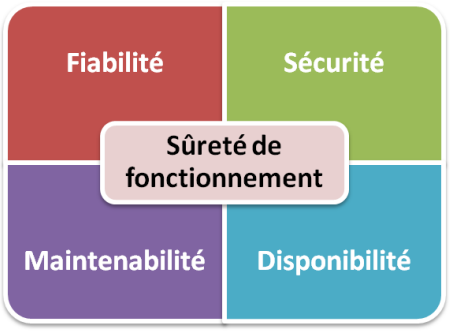
Dans le prochain chapitre, nous entamerons l'étude des middlewares RFID. Nous commencerons par présenter l'architecture en couches d'un middleware RFID et quelques middlewares RFID existants. Puis, nous présenterons les techniques de tolérance aux fautes dans les systèmes RFID. Enfin, nous conclurons par un comparatif des middlewares étudiés.

Chapitre 2

Etude des middlewares RFID







Sommaire

Mots Clés

Système RFID
Middleware
Filtrage
Agrégation des données
Diffusion des données
LLRP
EPCglobal Network
Sûreté de fonctionnement
Monitoring
Performance d'un lecteur RFID

CHAPITRE 2
ETUDE DES MIDDLEWARES RFID

I. Introduction

La technologie RFID s'est développée très rapidement cette dernière décennie pour se répandre dans beaucoup de domaines et envahir notre vie quotidienne à travers une variété d'applications. Ceci nécessite un déploiement fiable de ce type de systèmes afin de répondre aux nouvelles exigences des utilisateurs en lien avec le partage des données et la distribution du système d'information sur plusieurs sites géographiquement éloignés. Malgré les avancées technologiques, ce déploiement est un challenge important qui peut empêcher l'adoption complète de cette technologie [39]. Avec les applications RFID et les sources de données qui se multiplient, la masse d'information générée est plus difficile à traiter ; ce qui impose plusieurs défis aux concepteurs de systèmes RFID. Parmi ces défis, notons :

- La quantité de données (reçues à partir des différentes sources) qui nécessite d'être organisée et classifiée est très grande.
- Les flux de données reçus doivent être fusionnés en fonction du temps en raison de la nature souvent temps réel des applications RFID [40].
- L'ensemble du système doit pouvoir gérer des applications de plus en plus complexes d'une manière évolutive et extensible pour répondre efficacement aux requêtes.

En outre, les problèmes liés au déploiement de cette technologie s'accentuent lorsqu'il s'agit de systèmes qui sont sujets à erreurs, spécialement les systèmes RFID passif qui sont connus pour leur comportement peu fiable ; *i.e.*, un taux de détection de tags situé entre 80 et 90 %. Ce taux est aggravé par des facteurs environnementaux (*e.g.,* présence de métal, de liquide, présence de plusieurs lecteurs ou plusieurs tags, *etc.*). D'ailleurs, le taux de lecture précédent tombe à 70% en présence de plus de 5 tags [7] [41].

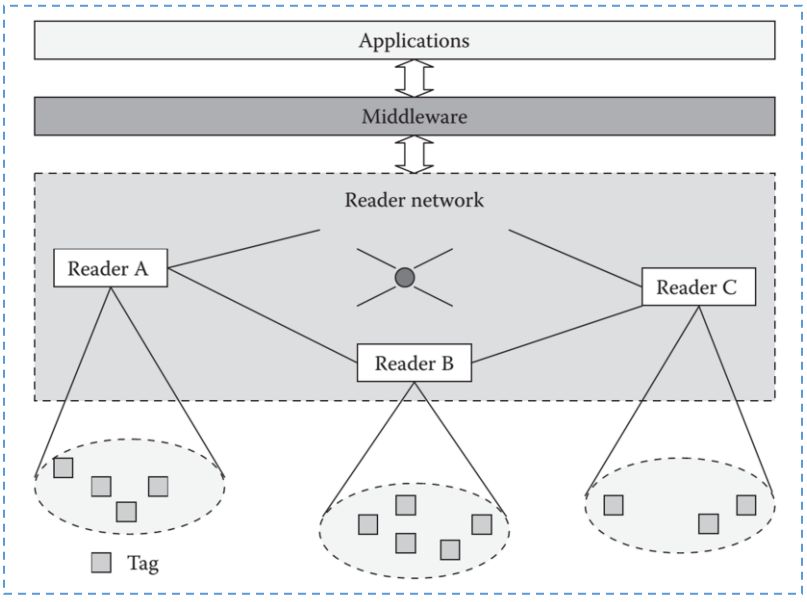


Figure 25. Architecture d'un système RFID distribué

Cela fait apparaître le besoin et l'utilité d'un autre composant qui vient s'insérer entre les appareils qui collectent les données et les applications qui les exploitent (Figure 25). Ce composant s'appelle « **middleware RFID** ». Il a pour tâche de faciliter le travail des applications utilisateurs, de gérer les différents lecteurs et rendre les systèmes RFID plus fiables. Ainsi, le middleware sera responsable des fonctions principales suivantes :

- La *diffusion des données reçues* : toutes les applications qui sont intéressées par un type de données doivent les recevoir.
- L'*agrégation des données* : toutes les données du même type ou bien destinées à la même application seront groupées afin de faciliter leur acheminement.
- Le *filtrage des données* : les données redondantes ou bien inutiles pour une certaine application seront filtrées.
- L'*interprétation des données* : le middleware doit savoir interpréter les données pour une meilleure gestion.
- La *gestion des lecteurs* (initiation de la connexion, lecture, écriture, *etc.*).

II. Architecture d'un middleware RFID

La figure 26 montre l'architecture d'un middleware RFID sous la forme d'un modèle à trois couches avec des fonctionnalités spécifiques pour chacune d'elles. Ce modèle résume les différents composants et fonctionnalités qu'un middleware RFID doit avoir, mais sans être exhaustif, étant donné que l'architecture interne d'un middleware est spécifique à chaque produit.

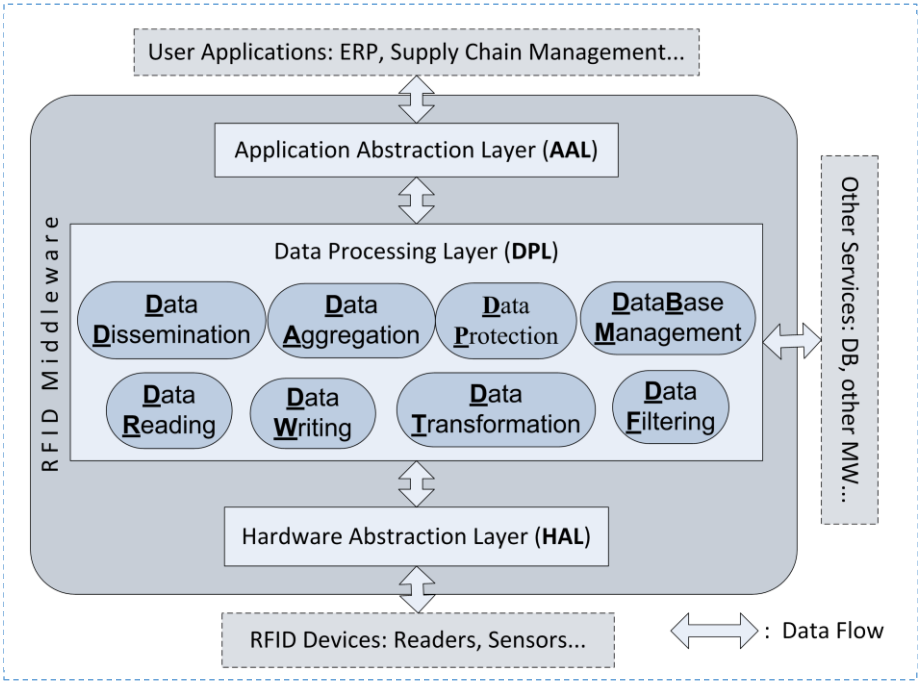


Figure 26. Architecture d'un middleware RFID

1. Couche d'Abstraction Matérielle (Hardware Abstraction Layer)

Cette couche est responsable de l'interfaçage avec le matériel. Elle permet ainsi d'unifier les interactions du middleware avec les différents composants hardware du système RFID, d'assurer des fonctions de filtrage à base d'identifiant du tag ou du lecteur, de contrôler et de transmettre les données vers les couches supérieures du middleware.

2. Couche de Traitement des Données (Data Processing Layer)

Cette couche représente le cœur du middleware. Elle est responsable de tous les services qu'un middleware RFID est supposé fournir comme le filtrage, l'agrégation et la protection des données.

Les principaux composants de cette couche sont les suivants :

2.1. Composant de Diffusion des Données (Data Dissemination)

CDD est responsable de la diffusion des données vers les applications clientes suivant des règles prédéfinies.

2.2. Composant d'Agrégation des Données (Data Aggregation)

CAD est responsable de regrouper les données suivant des critères bien spécifiques comme « regrouper les données qui viennent du même lecteur » ou encore « regrouper les données qui sont destinées à la même application », *etc.*

2.3. Composant de Protection des Données (Data Protection)

CPD est responsable de la protection des données et la gestion des droits d'accès des applications aux différents services fournis par le middleware.

2.4. Composant de Gestion de la Base de Données (Database Management)

CGBD est responsable de l'organisation, du partage, de la sauvegarde et de la suppression des données de la base de données.

2.5. Composant de Lecture des Données (Data Reading)

CLD est responsable de la collecte des données des lecteurs et des tags RFID à travers un ensemble de commandes (requêtes) qu'il envoie aux lecteurs RFID.

2.6. Composant d'Ecriture des Données (Data Writing)

CED est responsable de la mise à jour des données stockées dans les différentes mémoires des étiquettes RFID.

2.7. Composant de Transformation des Données (Data Transformation)

CTD est responsable du traitement et du formatage des données sous différents formats. Ainsi, il permet de transformer les données brutes reçues en « événements métiers » exploitables par les applications clientes (*e.g.*, tel produit se trouvait à telle place à telle heure).

2.8. Composant de Filtrage des Données (Data Filtering)

CFD est responsable de la suppression des données inutiles notamment les doublons, ainsi que de l'extraction des données utiles (demandées) à partir de celles reçues (*e.g.,* extraire le type du produit à partir du code EPC reçu en lui appliquant un masque).

3. Couche d'Abstraction Applicative (Application Abstraction Layer)

Cette couche est responsable de l'interfaçage des applications clientes hétérogènes avec le middleware en leur donnant accès aux différents services fournis par le middleware.

III. Middlewares existants

Il existe plusieurs middlewares RFID avec des architectures et des fonctionnalités plus ou moins différentes. Parmi ces middlewares, nous pouvons citer :

1. Middleware WinRFID

Le programme de recherche sur la RFID du consortium **WINMEC**[15] (Wireless INternet for Mobile Enterprise Consortium) à l'université de Californie à Los Angeles avait pour but de promouvoir, de guider et de démontrer le potentiel de cette technologie. Un middleware appelé WinRFID, a été développé dans le laboratoire WINMEC RFID Lab. WinRFID est un middleware développé en utilisant le Framework .NET de Microsoft. WinRFID démontre la facilité d'intégration de la technologie RFID dans les infrastructures informatiques existantes [42]. Il se présente comme une infrastructure évolutive et extensible (facilité d'intégration avec les applications de l'entreprise et celles de ses partenaires, grâce notamment aux nombreuses API[16] dont il dispose) [42] [43]. Il fournit un environnement distribué et une gestion des données sûre et intelligente (transformation des données que quand c'est nécessaire, acheminement des données traitées suivant les technologies appropriées, collection des données à partir d'événements reçus d'une variété de capteurs [43], *etc.*)

WinRFID se présente sous la forme d'une architecture en cinq couches. La figure 27 illustre les différentes couches ainsi que leurs fonctions.

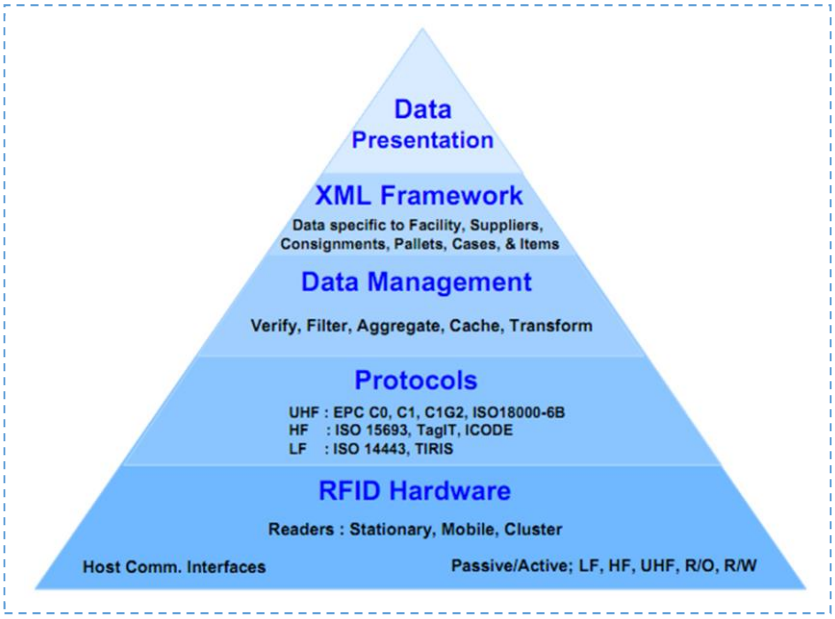


Figure 27. Architecture multicouche de WinRFID [42]

1.1. La couche physique (RFID Hardware)

Elle est responsable de l'interfaçage avec le matériel (lecteurs, tags et autres capteurs). Elle représente l'abstraction des parties physiques du système afin de rendre plus simples la gestion et l'ajout de nouveaux composants [44] (comme des lecteurs, des tags, de nouveaux modules d'entrées/sorties afin d'étendre les fonctionnalités du middleware). Elle fournit l'interface pour la prise en compte des différentes commandes pour chaque type de matériel, des modules d'E/S pour la lecture, l'écriture et le support des différents types de fréquences (LF, HF, UHF), ainsi qu'une interface commune pour tous les tags [44] (qui sont en lecture seule, en lecture/écriture, passifs, actifs,…).

[15] www.winmec.ucla.edu
[16] Application Programming Interfaces

1.2. La couche des protocoles (Protocols)

Dans un bon middleware RFID, le support des multiples protocoles de communication et la capacité d'en ajouter d'autres sont indispensables pour continuer à répondre aux nouvelles exigences des entreprises dans le domaine de la RFID [42] [44]. Cette couche permet l'encapsulation des commandes reçues depuis les couches supérieures suivant le protocole de communication utilisé (*e.g.,* ISO 15693, ISO 18000-6B, ICode, EPC Class 0, *etc.*), l'analyse et l'extraction des données des tags en tenant compte du standard utilisé [44]. Chaque lecteur RFID informe cette couche du protocole à utiliser lors de l'initiation de la communication avec un tag.

1.3. La couche gestion des données (Data Management)

Lors d'une opération de lecture, des erreurs peuvent survenir notamment à cause de la présence de plusieurs tags, la présence d'obstacles, *etc.* Ce qui nécessite plusieurs lectures des mêmes tags afin de combler ces incohérences (erreurs de lecture, tag non lu, tag lu plusieurs fois,…). Ces dernières sont prises en charge dans cette couche [42], en définissant des règles de gestion pour la vérification des données lues, l'agrégation et le filtrage des données redondantes. Cette couche dispose d'un moyen de stockage local temporaire des données sous forme d'une file d'attente [44]. Cela permet le traitement asynchrone du flux de données qui parvient des couches inférieures, ce qui donne assez de temps pour l'application de toutes les règles de gestion prédéfinies sur les données en question.

1.4. La plateforme XML (XML Framework)

Les données reçues sont vérifiées et filtrées par la couche inférieure (Data Management). Elles sont formatées en données XML pour une meilleure interopérabilité des systèmes [43]. La fonction principale de cette couche est de rendre les données plus souples pour une manipulation plus facile par les applications des entreprises (par exemple, la recherche d'informations est plus facile grâce à ce format).

1.5. La couche présentation des données (Data Presentation)

Cette couche est responsable de récupérer les données à partir de la plateforme XML pour la visualisation (sous forme de tableaux, de graphes, *etc.*) et la prise de décision. Un portail permet aux utilisateurs de s'abonner aux informations qui leur sont utiles et d'y accéder de manière sécurisée à travers un mécanisme d'authentification. Les données sont fournies à l'utilisateur dans le format par défaut ou bien sous un autre format (cette couche offre la possibilité d'activer des plug-ins fournis à travers des librairies pour la transformation des données [42]).

2. Middleware RF²ID (Reliable Framework for Radio Frequency IDentification)

Le middleware RF²ID est conçu de manière à prendre en compte la vulnérabilité de la technologie notamment la technologie passive. Il a comme objectifs les éléments suivants :

- Rendre le système évolutif et plus fiable (grâce aux lecteurs virtuels).
- Equilibrage de charge entre les composants de traitement lors de la manipulation d'une grande quantité de données.
- Fluidité et rapidité de circulation des données (haut débit).
- Organisation des données (besoin de trier et d'organiser proprement les données lorsqu'il s'agit d'un grand volume de données, pour une meilleure réactivité aux requêtes).

Le middleware RF²ID repose sur deux notions principales, à savoir les lecteurs virtuels et les chemins virtuels. La figure 28 montre l'architecture détaillée de la plateforme RF²ID.

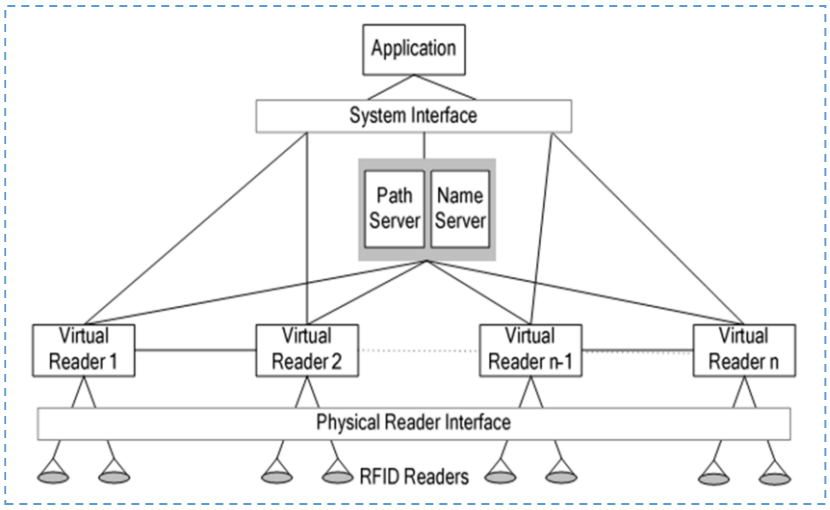


Figure 28. Architecture du système RF²ID [39]

1.1. Lecteurs virtuels (virtual readers)

Un lecteur virtuel (VR) est un module chargé de gérer un ensemble de lecteurs physiques (PR) se trouvant dans le même voisinage. Cette notion de VR est aussi utilisée dans d'autres études notamment dans les travaux de Fagui Liu [45].
Un lecteur RFID est par nature peu fiable (taux de lecture = 70% [7] [8] [9]) ; par conséquent, l'association de plusieurs lecteurs physiques (PR) est pleinement justifiée pour la minimisation du taux d'erreurs de lecture.

Dans un lieu où la disposition des lecteurs ne change pas, l'affectation des lecteurs physiques à un lecteur virtuel donné est réalisée durant la phase d'initialisation. Pour le bon fonctionnement du système, chaque VR gère en tout cinq listes qui sont comme suit :

- **observedTagList** : représente la liste de tous les tags détectés par tous les PR associés à ce VR (existence de données dupliquées).
- **receivedTagList** : liste des tags détectés après filtrage (receivedTagList = observedTagList + Filtrage).
- **expectedTagList** : est la liste des tags attendue par le VR. Elle est envoyée par le VR qui le précède dans le chemin virtuel ou le Vpath (cette notion de Vpath sera traitée dans le deuxième point de cette section). Notons que le premier VR dans le chemin ne peut pas gérer cette liste (expectedTagList = $\text{receivedTagList}_{\text{VR précédent}}$).
- **missingTagList** : représente l'ensemble des tags attendus mais non détectés par les PR qui lui sont associés. On admettra que le VR va recevoir la liste « expectedTagList » de son voisin avant la constitution de la liste « receivedTagList » afin de pouvoir calculer « missingTagList » (missingTagList = expectedTagList − receivedTagList).
- **spuriousTagList** : désigne l'ensemble des tags détectés par les PR mais non attendus (spuriousTagList = receivedTagList − expectedTagList).

NB : la liste « receivedTagList » du premier VR dans un Vpath deviendra la liste « expectedTagList » pour tous les autres VR qui appartiennent au même Vpath.

1.2. Chemins virtuels (Virtual Paths)

Ce sont des canaux de communication logiques qu'on appelle « Chemins virtuels (Vpaths) ». Ils sont constitués de plusieurs VR[17] qui suivent les flux de données afin d'améliorer la fiabilité du système. En effet, Ahmed Nova [39] [40] a constaté que la nature des flux de données peut aider dans l'organisation des données. Il s'est donc basé, sur la nature de ces flux pour la création des Vpaths [46].

Le fonctionnement de la plateforme RF²ID est illustré à travers un exemple d'application dans la figure 29. Il concerne la gestion et l'acheminement des produits dans un entrepôt.

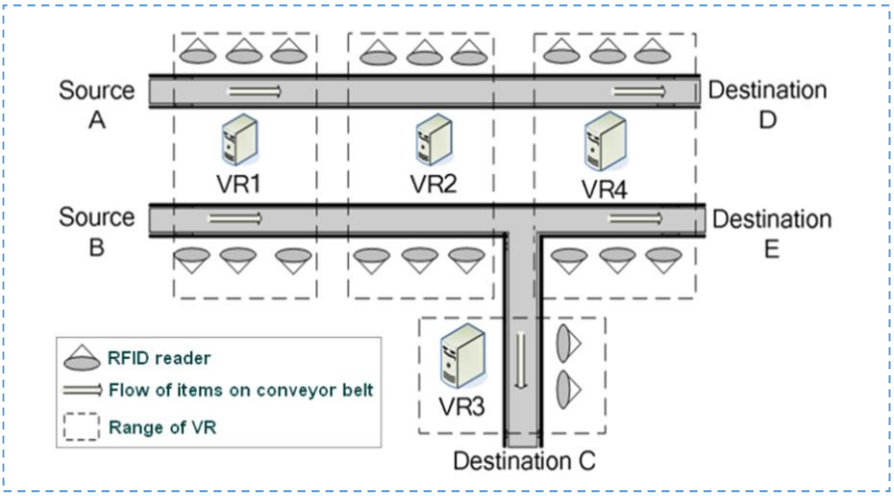


Figure 29. Exemple d'application de RF²ID dans un entrepôt [39]

L'entrepôt est divisé en plusieurs zones géographiques. Chacune de ces zones est associée à un lecteur virtuel (VR) et dispose d'un ensemble de lecteurs PR. Chaque lecteur virtuel couvre une zone qui s'étale sur plusieurs tapis roulants. Ainsi le VR1 concernera tous les produits qui transitent dans l'entrepôt (sur tous les tapis) ; par contre le VR3 ne concernera que les produits qui arrivent par la source B vers la destination C.

La création des chemins virtuels est gérée par les deux serveurs « Path Server » et « Name Server ». « Path Sever » dispose de tous les chemins virtuels existants. Il sera interrogé par un VR (qui a reçu une requête de l'application utilisateur[18]) qui veut créer un Vpath afin de savoir si ce chemin n'existe pas déjà. Dans le cas où le chemin à créer n'existe pas, Name Server sera interrogé à son tour. Ce dernier dispose de l'emplacement (cartographie) de tous les VR dans l'entrepôt. Il permet ainsi de retourner une liste des VR à partir de deux données en paramètres à savoir la source du Vpath et sa destination.

Les lecteurs VR auront aussi à gérer deux paramètres en plus des listes déjà citées. Ces deux paramètres sont :

- $conn_{in}$: représente le nombre maximum de messages entrants gérés par le lecteur VR par unité de temps.
- $conn_{out}$: désigne le nombre maximum de messages sortants gérés par le VR par unité de temps.

Pour qu'un lecteur VR accepte de participer dans un Vpath, il commence par calculer sa charge suivant tous les Vpaths auxquels il appartient, et une estimation de la nouvelle charge avec le nouveau Vpath est calculée. Cette nouvelle charge ne doit pas dépasser $conn_{in}$ et $conn_{out}$, sinon le VR ne pourra pas participer à ce nouveau chemin.

[17] Un Vpath est constitué de plusieurs lecteurs virtuels (VR) et un VR peut appartenir à plusieurs Vpath.

[18] Chaque VR est capable de répondre aux requêtes des applications ou bien de les aiguiller grâce au Vpath vers le VR le plus apte à répondre.

Pour garantir la cohérence des systèmes, les listes créées et échangées entre les VR sont datées. Cela évite à un VR de comparer sa liste receivedTagList avec une ancienne expectedTagList ou bien avec la suivante dans le cas où la première a été perdue lors de la transmission ; en d'autres termes, cela permet d'éviter beaucoup d'erreurs.
Cette architecture est très adaptée pour le suivi des objets, mais aussi pour leur localisation grâce aux chemins virtuels et à l'utilisation desdites listes par les VR. Ainsi, si un objet se perd lors de son acheminement, il suffit d'interroger le dernier VR qui a pu le lire, et en utilisant l'atténuation des signaux des PR[19], l'objet perdu pourra être localisé de manière assez précise.

2. Middleware Fosstrak

Fosstrak[20] (Free and Open Source Software for TRAce and tracK) est une plateforme open source (anciennement appelée « **Accada platform** ») qui est conçue pour répondre aux besoins des applications de traçage et de suivi (track&trace applications). Cette plateforme implémente les spécifications d'EPCglobal Inc., (notamment le standard EPC Network) sous forme de trois modules séparés [47] [48] : un module lecteur, un middleware et le service EPCIS. La figure 30 montre l'architecture en couches de cette plateforme.

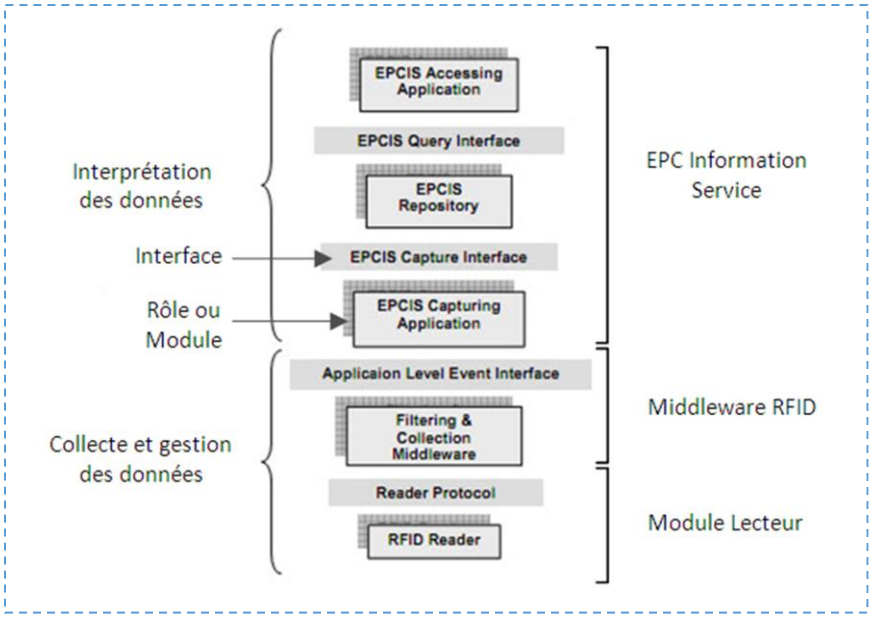


Figure 30. Architecture de la plateforme Fosstrak [47]

2.1. EPC Information Services

Le module EPCIS reçoit les données du middleware, les transforme éventuellement dans un format plus adapté pour les entreprises, les stocke et les rend disponibles pour ces dernières (voir Section IV.2.3 du chapitre 1 pour plus d'informations sur le réseau EPC).

2.2. Module lecteur

Ce module implémente la spécification d'EPCglobal sur le protocole lecteur. Il fournit ainsi plusieurs fonctionnalités comme le filtrage à partir de l'ID du tag ou de l'antenne du lecteur, ou bien l'agrégation en fonction du temps ou de l'espace (par exemple, plusieurs antennes d'un lecteur peuvent être groupées virtuellement en une seule source). Il prend en charge un grand nombre de lecteurs physiques

[19] Afin de rendre la recherche plus efficace, la désactivation de certains PR peut accélérer la recherche, et notamment grâce à la méthode dichotomique qui est très adaptée à ce genre de recherche.
[20] http://www.fosstrak.org

ainsi qu'un mode de simulation en cas d'absence de lecteurs. Il supporte aussi l'écriture sur les tags et l'activation par une source externe comme les capteurs de mouvement [47]. Les données capturées peuvent être diffusées aux applications suivant un mécanisme d'abonnement « publish-subscribe » (qui sera traité ci-après). L'implémentation de ce module est donnée par la figure 31.

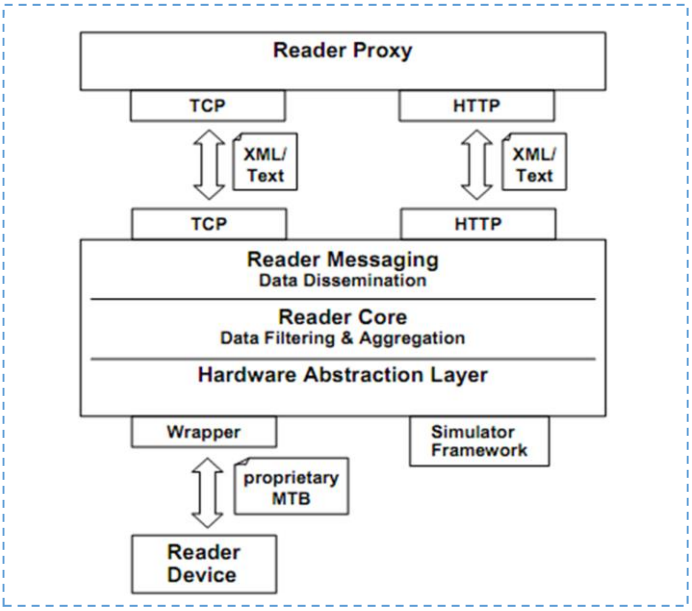


Figure 31. Implémentation du module Lecteur [47]

2.3. Le middleware

En plus du filtrage, de l'agrégation et de la diffusion des données, ce composant permet aux applications de définir une souscription ou un abonnement dans lequel chaque application définit les lecteurs à utiliser, le type de données qui l'intéressent, leurs formats, *etc.* Cet abonnement est utilisé afin de configurer les lecteurs correspondants. La figure 32 montre ce mécanisme d'abonnement.

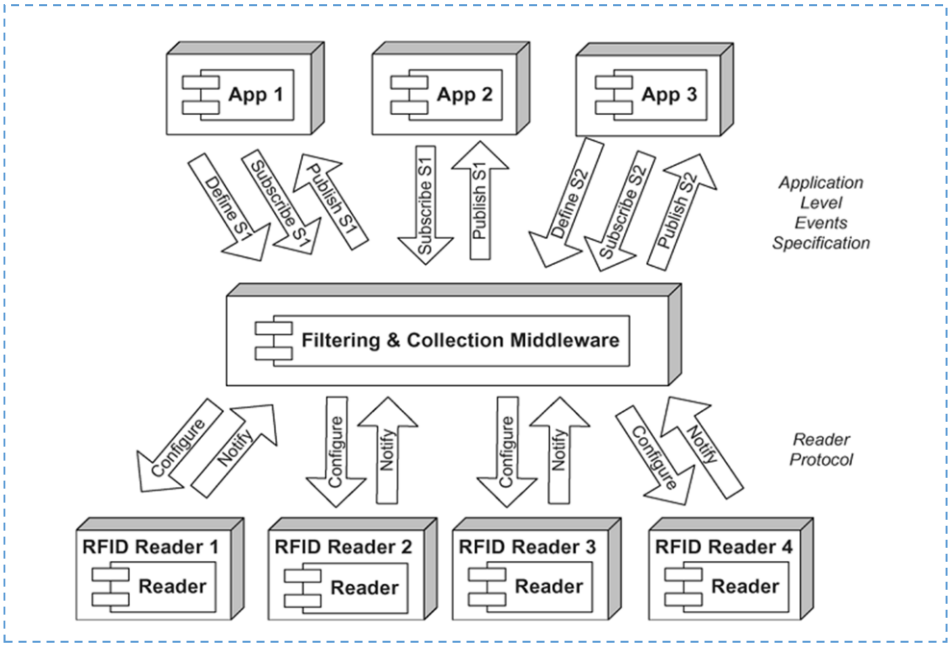


Figure 32. Fonctionnement du middleware Fosstrak [47]

3. Middleware AspireRFID

AspireRFID (Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications) est un projet open-source Européen, lancé dans la seconde moitié de l'année 2008 par le consortium OW2[21] pour le développement et la promotion d'un outil open-source fiable dont le rôle est de faciliter le déploiement (avec un coût minime) et la gestion des applications RFID [49] [50]. Il est le fruit de la collaboration de plusieurs organisations à travers l'Europe comme l'INRIA et l'Université Joseph Fourier de Grenoble en France, Athens Information Technology en Grèce, Melexis Technologies en Suisse, Open Source Innovation en Angleterre, *etc.*

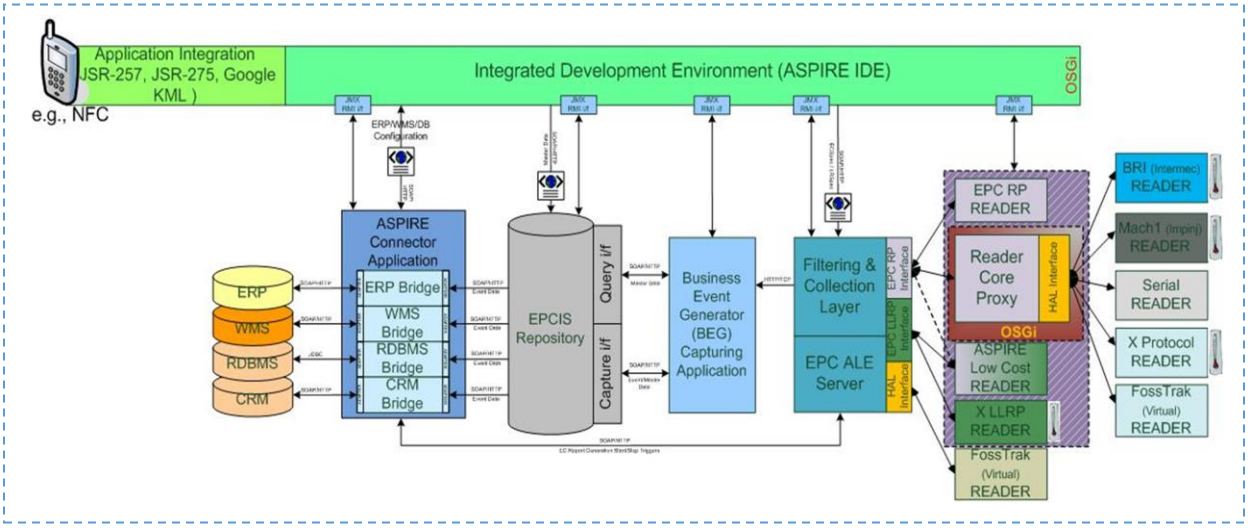


Figure 33. Architecture de la plateforme AspireRFID [51]

Son architecture implémente plusieurs spécifications comme celles de EPCglobal *Inc.*, NFC Forum, JCP et OSGi Alliance. Cette architecture est donnée par la figure 33 sur laquelle, on peut distinguer les composants suivants [51] :

3.1. Hardware Abstraction Layer (HAL)

Ce composant représente la couche de plus bas niveau dans la plateforme AspireRFID. Il fournit une abstraction du matériel afin d'unifier la manière avec laquelle le middleware interagit avec les lecteurs des différents constructeurs et qui utilisent donc des protocoles de communication différents. Les spécifications qui satisfont le besoin d'une norme à ce niveau sont le protocole RP (EPCglobal Reader Protocol) et LLRP[22] (voir Chapitre 1). Les méthodes de la couche HAL et le matériel varient suivant le constructeur (nécessité d'une connexion série ou Ethernet, *etc.*). Le protocole de communication peut lui aussi varier entre TCP (Transfer Control Protocol), SSL (Secure Sockets Layer) ou HTTP (HyperText Transfer Protocol).

3.2. Reader Core Proxy (RCP)

Ce composant sert à rendre tous les lecteurs compatibles avec les lecteurs EPC. Il est utilisé comme un médiateur entre les lecteurs supportant un protocole donné et l'interface RP du composant de filtrage.

[21] OW2 est une communauté open-source crée en janvier 2007, et dédiée au développement des technologies middlewares libres et fiables [http://www.ow2.org].
[22] Lower Level Reader Protocol

3.3. Filtering and Collecting (ALE)

Cette couche fournit une interface flexible pour ALE[23] (Application Level Events) pour le filtrage et le comptage des opérations qui produisent des rapports en réponse aux requêtes des clients. Ces rapports sont produits par la collecte et la fusion des données arrivant des différentes sources, puis distribués aux applications intéressées (à la manière de Fosstrak qu'on a vu précédemment).

3.4. Business Event Generator (BEG)

Ce composant se situe entre la couche de filtrage et le module de partage d'informations (EPCIS). Son rôle est d'automatiser le mappage entre les rapports du composant de filtrage et les événements du système. En d'autres termes, le BEG peut être vu comme une couche qui analyse les rapports EPC-ALE, les fusionne avec des données « sous un contexte d'entreprise » (à l'aide de l'IDE AspireRFID qui décrit les processus commerciaux d'une entreprise) pour créer des « business events » qu'il stocke dans l'EPCIS.

3.5. Integrated Development Environment (AspireRFID IDE)

Il rend possible la gestion visuelle de tous les fichiers de configuration et des métadonnées qui sont requis pour le fonctionnement du système. Il inclut :

- Un éditeur pour la configuration des lecteurs physiques et leurs paramètres.
- Un éditeur pour la configuration et la définition des lecteurs logiques avec les protocoles RP ou LLRP, ainsi que la simulation des lecteurs physiques.
- La gestion des spécifications de lecture et de filtrage.
- L'exécution des commandes ALE du composant de filtrage sur les lecteurs ou tout composant implémentant les spécifications ALE, *etc.*

4. Middleware Sun RFID

SUN Java System RFID est une plateforme RFID développée par Sun Microsystems et qui prend en charge dans sa conception les grands standards acceptés par les industriels (comme ceux développés par EPCglobal Inc.) [52]. Il est conçu de façon à fournir un grand niveau de fiabilité et d'évolutivité du réseau EPC en simplifiant l'intégration de celui-ci dans les systèmes existants des entreprises.
Sun RFID est responsable de la communication et du traitement des données EPC et des événements entre tags, lecteurs et applications des chaînes logistiques comme les **ERP** (Enterprise Ressource Planning). Il est basé sur deux composants principaux : « *Event Manager* » et « *Information Server* » comme illustré dans la figure 34.

[23] ALE est une spécification logicielle des fonctions d'un système RFID (à savoir l'activité de lecture et celle d'écriture) et toutes les fonctionnalités qu'elles impliquent (comme le traitement des données).

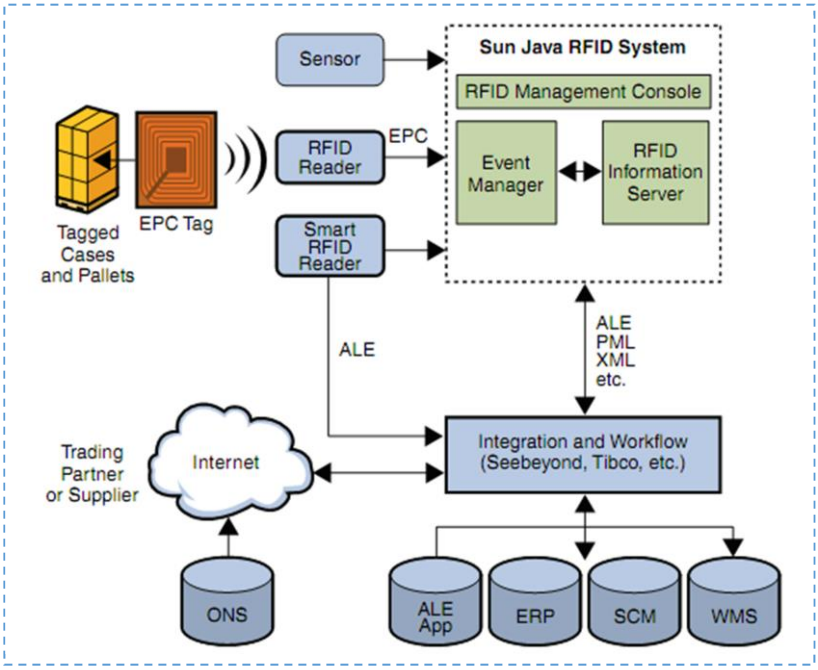


Figure 34. Plateforme Sun Java System RFID [53]

4.1. RFID Event Manager (gestionnaire des événements)

Il communique avec les lecteurs EPC pour le traitement des données EPC qui arrivent vers le système. Il communique aussi avec les applications utilisateurs, ainsi qu'avec le composant RFID Information Server pour le log (l'enregistrement) des événements et des données EPC. Ce composant est basé sur la technologie Jini[24] (dont le but est de simplifier l'interfaçage des composants informatiques hétérogènes) afin de faciliter la capture, le filtrage et le stockage des événements EPC générés par les lecteurs connectés au réseau. Son but principal est de s'interfacer avec les lecteurs, recueillir les événements EPC, filtrer les données redondantes et alimenter le serveur d'information RFID ou toute autre application ERP par les événements EPC importants pour davantage de traitements.

Event Manager consiste en l'exécution d'un ou plusieurs agents. Un agent est constitué de trois types de composants comme le montre la figure 35.

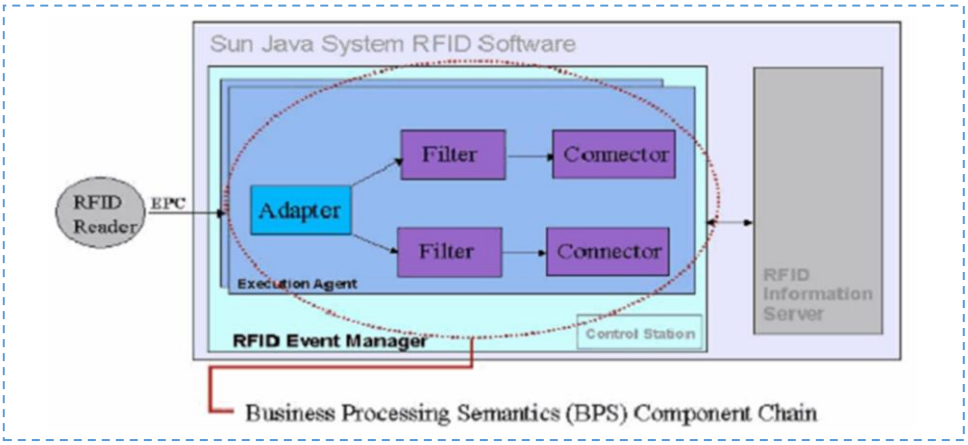


Figure 35. Structure du RFID Event Manager [54]

[24] Jini est une architecture réseau pour la construction de systèmes distribués sous forme de services coopérants modulaires. Elle simplifie l'interfaçage et la connexion entre les systèmes d'information : PC, PDA, Téléphone mobile, Caméra, Fax, GPS, Alarme, *etc.* [java.net/projects/jini]

4.1.1. Adapter (Adaptateur)

Il s'occupe de recevoir l'EPC du tag, et génère un événement daté contenant la source de ce dernier (lecteur et/ou antenne qui a détecté le tag). Puis, il l'envoie au composant de filtrage.

4.1.2. Filters (Filtres)

Il s'occupe du filtrage des données suivant les règles définies (filtrage à base de l'ID du lecteur ou du tag…) et transmet les données résultantes.

4.1.3. Loggers or Connectors (Connecteurs)

Il sert comme un connecteur aux autres applications. Il leur fournit les données dans un format adapté notamment pour le serveur d'information pour le stockage.

4.2. RFID Information Server

RFID IS est une application J2EE qui sert comme interface pour la capture et la recherche des données. Ce serveur est responsable du stockage et de l'agrégation des données qui concernent un événement EPC donné. Il est typiquement utilisé pour la transformation des données récoltées (qui sont dans un « bas-niveau d'observation ») vers un plus haut niveau de représentation (qui est plus adapté aux applications d'entreprise). Ce composant fonctionne sur le serveur Sun Java System Application Server. Il s'interface aussi avec d'autres systèmes d'information à travers l'échange de messages en XML. Cet échange est supporté par le protocole HTTP ou bien JMS (Java Message Service).

5. Middleware IBM WebSphere RFID

La société IBM a introduit, elle aussi, son outil middleware appelée *WebSphere RFID* pour les systèmes RFID, pour la collecte et l'analyse des données de différentes sources, ainsi que leur diffusion aux applications clientes. IBM a commencé par diviser le système RFID et le représenter sous forme de plusieurs domaines [55], comme le montre la figure 36 :

- **Tagged Object Domain** (le domaine des objets étiquetés).
- **Antenna & Reader Domain** (le domaine des lecteurs),
- **Edge Domain** (pour les fonctionnalités de filtrage et d'agrégation des données).
- **Premises Domain** (pour le filtrage et l'agrégation de plus haut niveau. Ce filtrage ne concerne que les événements dont la valeur est importante pour les entreprises).
- **Business Process Integration Domain** (pour la connexion des infrastructures RFID avec les applications des entreprises).
- **Enterprise and Business Application Domain** (représente les différentes applications des entreprises exploitant les données qui proviennent des systèmes RFID).
- **Object Directory Domain** (responsable de fournir des informations sur les objets étiquetés en utilisant leur ID comme clé de recherche).
- **System Management Domain** (permet la gestion des systèmes RFID à distance dans un environnement distribué comme la possibilité de configuration et de mise-à-jour des équipements).

IBM WebSphere RFID est une solution middleware qui s'étale sur les trois domaines *Edge*, *Premises* et *Business Process Integration Domain*. Elle permet l'interconnexion des équipements RFID avec les systèmes d'information des entreprises. Le middleware *WebSphere RFID* est constitué de trois composants :

5.1. Premises Server (PS)

PS est la pièce centrale de la solution d'IBM. C'est une application J2EE, considérée comme un intermédiaire entre le monde physique (équipement RFID utilisant le « *Edge Domain* ») et le monde des technologies de l'information qui utilise le « *Business Integration Domain* ». Il permet une bonne

intégration en offrant des possibilités de personnaliser la solution et de l'optimiser suivant les spécifications et les exigences de l'entreprise.

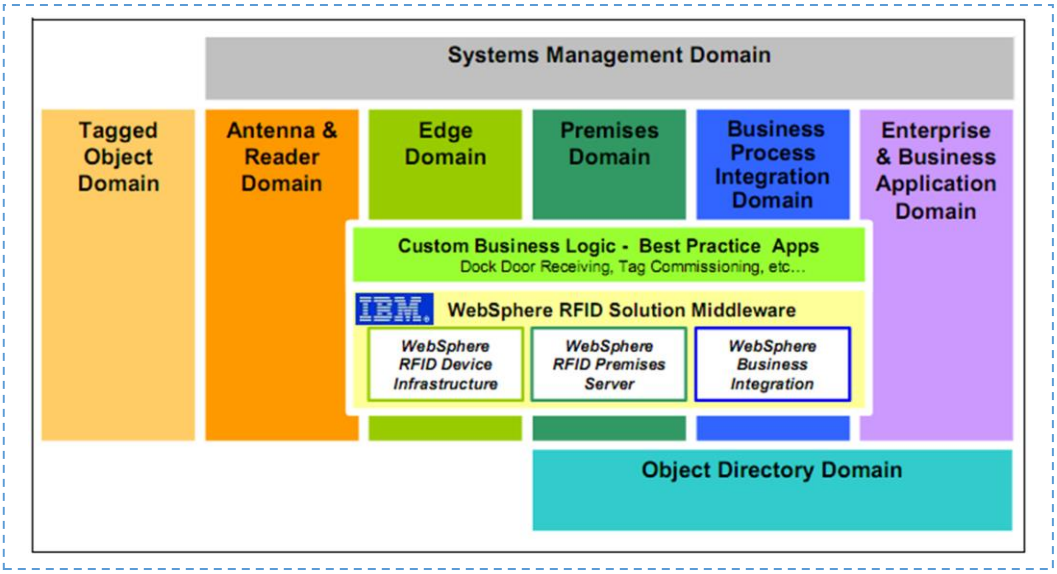


Figure 36. Solution IBM RFID [55]

5.2. Device Infrastructure (DI)

DI est un ensemble de technologies sous licence fourni aux fabricants des équipements RFID programmables comme les lecteurs intelligents. C'est une plateforme OSGi[25] (Open Services Gateway initiative) qui permet elle aussi la personnalisation de la solution RFID pour des besoins particuliers.

5.3. Business Integrated Server (BIS)

BIS spécifie comment la solution RFID d'IBM permet de connecter la plateforme RFID à savoir « *Premises Server* » et les « *edge controllers* » au système d'information de l'entreprise. Il propose un ensemble de services qui permettent à l'entreprise d'intégrer des applications Web avec les applications existantes.

6. Middleware FlexRFID

FlexRFID est une solution middleware conçue pour la gestion et le contrôle des lecteurs et d'autres équipements de capture et de détection (de température, de mouvement, d'accélération, *etc.*), ainsi que le traitement de leurs données [20] [56]. C'est un outil facile à mettre en place même dans un environnement hétérogène où plusieurs standards et matériels (de différents constructeurs) coexistent. FlexRFID est organisé en quatre couches qui interagissent entre elles, comme le montre la figure 37.

6.1. Device Abstraction Layer (DAL)

Cette couche est responsable de l'interaction avec le matériel de capture de données indépendamment de ses caractéristiques. Elle se décompose en trois modules :

- **Data Source Absraction Module (DSAM)** : fournit une vue standard des données quel que soit le protocole de communication (EPC Gen2, ISO 15693, ISO 14443A), l'interface air (UHF, HF) ou bien le type de mémoire du tag.

[25] Une plateforme de services fondée sur Java et qui peut être gérée de manière distante.

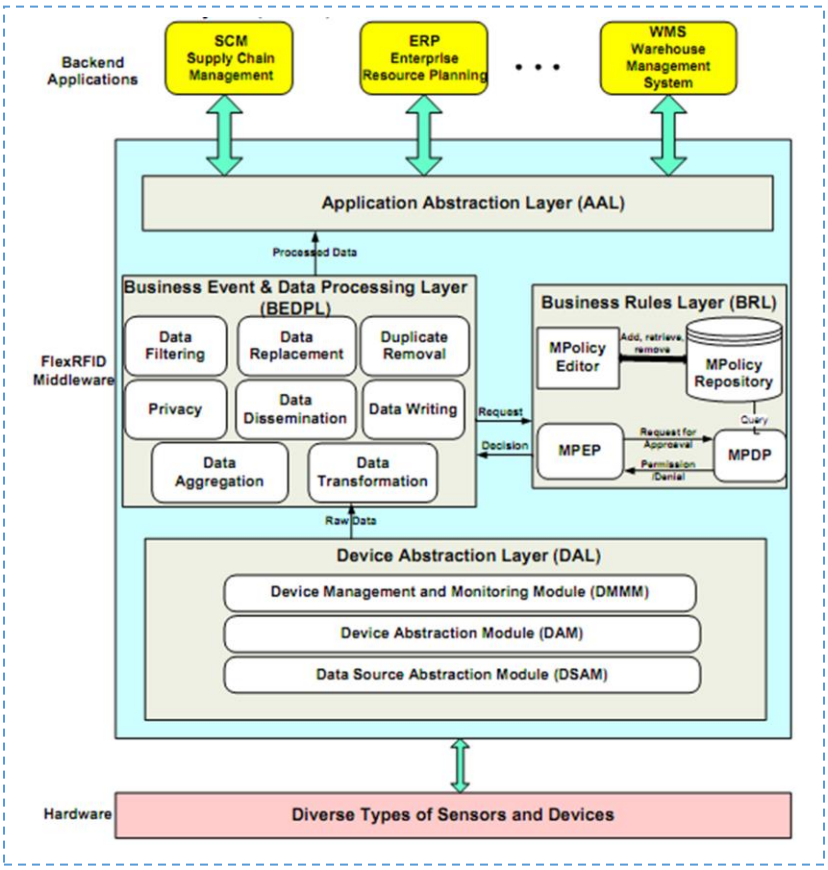


Figure 37. Architecture du middleware FlexRFID [20]

- **Device Abstraction Module (DAM)** : fournit une interface commune pour l'accès aux matériels des différents constructeurs.
- **Device Management and Monitoring Modules (DMMM)** : permet la configuration des lecteurs suivant les spécifications des couches supérieures. Ce module est responsable du chargement/déchargement (de façon dynamique) des librairies des drivers et des adaptateurs du matériel. Ceci laisse le middleware léger, puisqu'un pilote n'est chargé que si nécessaire.

6.2. Business Event and Data Processing Layer (BEDPL)

Cette couche joue le rôle d'un médiateur entre la couche DAL et la couche AAL (*Application Abstraction Layer*) ; *i.e.*, elle reçoit les données brutes de la couche inférieure, les transforme puis les transmet à la couche AAL. Elle est aussi responsable du traitement des requêtes du client suivant les règles d'accès. Le traitement d'une requête par la BEDPL ne se fait que si le service suscité par la requête est autorisé par la couche BRL (*Business Rules Layer*). Les services fournis par cette couche sont : la diffusion, l'agrégation, la transformation, le filtrage, la suppression, le remplacement[26] et la confidentialité des données.

6.3. Business Rules Layer (BRL)

Cette couche a pour rôle la gestion des stratégies et des politiques qui définissent les règles qui accordent ou refusent l'accès aux ressources et aux services du middleware. Cela est fait en déterminant

[26] Il arrive que le tampon où sont stockées les données en attente de traitement par l'application soit plein, ce qui génère des problèmes de Buffer Overflow. Le service de remplacement des données permet à l'application de spécifier les actions à exécuter (la politique de remplacement des données) lorsque de tels problèmes surviennent.

quelle règle appliquer lorsqu'une application demande un service de la BEDPL. Le composant MPE (*Middleware Policy Editor*) permet la récupération, la création et la suppression des règles de la base de données du MPR (*Middleware Policy Repository*).
NB : la base de données contient toutes les règles de gestion d'accès aux différents services du middleware.

6.4. Application Abstraction Layer (AAL)

Cette couche fournit une interface pour les différentes applications des entreprises à travers laquelle ces dernières accèdent aux services offerts par le middleware FlexRFID.

7. Middleware REFiLL (Rfid Event FiLtering tooLchain)

REFiLL est une plateforme middleware légère, simple, mais puissante pour la capture et le filtrage des données qui offre une certaine programmabilité à l'aide des fichiers XML [57]. Son architecture est donnée par la figure 38. Cette plateforme permet de faciliter sensiblement le développement des systèmes RFID surtout pour les PME (Petites et Moyennes Entreprises) qui ne peuvent pas investir dans les applications industrielles de la RFID. Cette plateforme s'insère dans l'architecture proposée par EPCglobal Inc., à savoir EPC Network, comme une solution de filtrage souple qui se situera entre la couche d'abstraction[27] des lecteurs (les adaptateurs) et le système d'information (voir la figure 39). La plateforme REFiLL permet de traiter les données et les événements en temps réel par des composants de filtrage programmables (des graphes de filtrage).

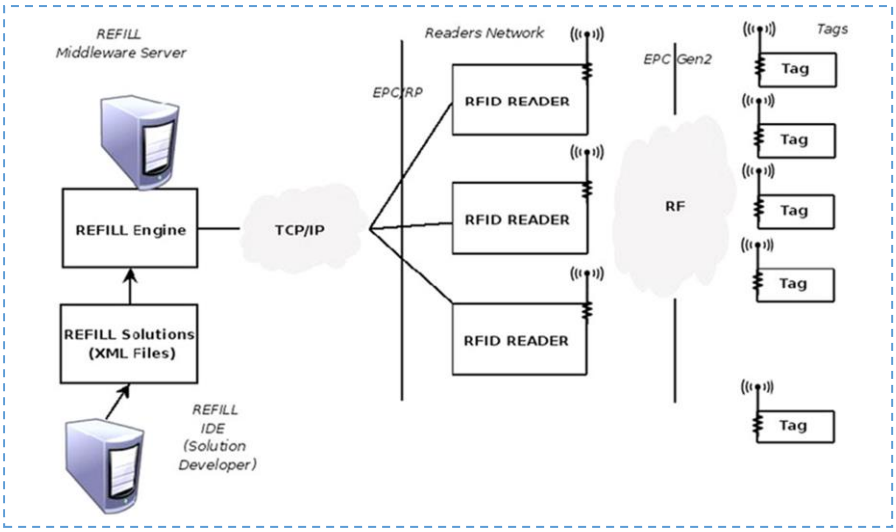


Figure 38. Architecture globale de la Plateforme REFiLL [57]

REFiLL adopte le concept de filtre programmable, qui a ses origines dans les commandes d'Unix (comme "grep" ou "awk"). Un filtre reçoit les données de la part des lecteurs et des autres filtres également et applique les règles de filtrage pour produire des sorties suivant sa configuration (les règles de filtrage sont définies dans des fichiers XML qui sont appelés des « REFiLL Solutions »). REFiLL accepte les « REFiLL solutions » constituées de deux types de fichiers codés en XML ; à savoir des fichiers **FML** (Filter Markup Language) pour la définition des filtres et des fichiers **FGML** (Filter Graph Markup Language) pour la définition des liaisons entre ces filtres. Ces fichiers XML seront traduits à l'aide du compilateur FMLCC en code Java puis en bytecode, qui sera exécuté par le middleware.

[27] REFiLL permet l'abstraction du matériel en proposant une API commune basée sur le protocole EPC-RP pour l'accès à ce matériel (essentiellement des lecteurs RFID).

Le langage FML fournit un moyen de spécifier les métadonnées du filtre comme son nom, sa version, ainsi que le nombre des ports d'entrée et de sortie. Il permet de faire des traitements évolués comme :

- L'exécution d'une séquence d'actions à l'aide du block < ForEveryTag >.
- Le traitement conditionnel à l'aide du block < If >.
- La définition de variables à l'aide du block < Var > pour le stockage temporaire lors du traitement (les types supportés sont Integer, Float, Double, Date, String,…).

La figure 39 montre la compatibilité de l'architecture de la plateforme REFiLL avec la norme EPC. Cette figure met l'accent sur les types de tâches requises dans un système RFID afin de répondre aux besoins des applications (comme la gestion des chaînes logistiques) :

- Interfaçage avec les lecteurs RFID.
- Collecte, filtrage et diffusion d'une multitude de données.
- Fourniture d'un accès transparent aux données pour les applications à travers une interface compatible EPC-ALE.

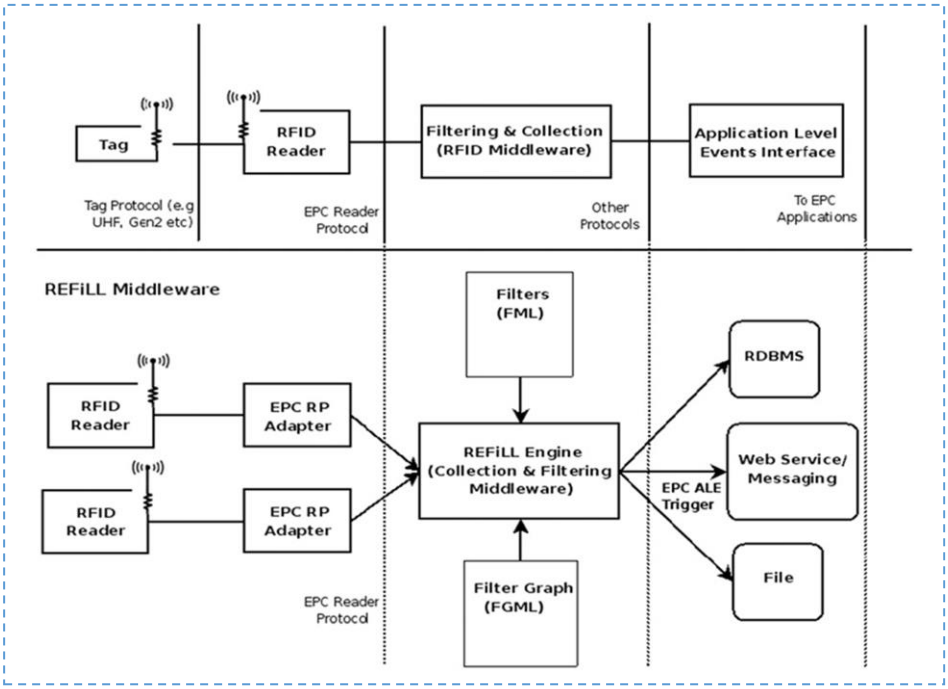


Figure 39. Comparaison entre l'architecture REFiLL et celle d'EPGglobal [57]

La plateforme REFiLL est composée de :

7.1. EPC RP Adapters

L'exécution des filtres est déclenchée par les événements générés par les lecteurs. REFiLL communique avec les lecteurs à travers l'EPC Reader Protocol (EPC-RP). Le middleware spécifie une API comme un sous ensemble de EPC-PR qui décrit les fonctionnalités du système telles que la connexion, la gestion et la souscription aux lecteurs. Ainsi, cette API minimise les efforts nécessaires pour la prise en charges des différents lecteurs par REFiLL (*i.e.*, tout lecteur supportant ladite API peut communiquer avec REFiLL).

7.2. Output Managers

Après le traitement des données par les filtres, REFiLL peut soit enregistrer le résultat ou bien exécuter un ensemble d'actions à la suite du traitement. *Output Managers* fournit une abstraction du système de l'entreprise et des sources de données comme le système de gestion de base de données (Data Base Management System ou DBMS), des fichiers, et aussi un accès aux informations des partenaires à travers un ensemble de services Web. *Output Manager* dispose aussi d'une API qui permet aux développeurs d'applications de personnaliser leurs gestionnaires de sortie (*Output Managers*) pour se conformer à leurs besoins.

7.3. EPC ALE and REFiLL ALE

Il existe plusieurs concepts communs dans EPC ALE et REFiLL mais REFiLL n'implémente pas toutes les spécifications de EPC ALE, comme les groupes d'exclusion par exemple. Ces groupes permettent aux développeurs d'exécuter des processus de filtrage à travers différents cycles de lecture. Cela est utile pour enregistrer des événements qui ont subi des changements de contexte. EPC ALE et REFiLL ALE prennent en charge les noms logiques et les alias des lecteurs RFID ; ils supportent les lecteurs virtuels avec de multiples configurations (comme un lecteur virtuel supporte les événements qui arrivent de plusieurs lecteurs physiques). En effet, REFiLL supporte ces configurations grâce l'utilisation des adaptateurs et des graphes de filtrage. Il peut créer un lecteur virtuel à partir de plusieurs sources de données en se basant sur le concept de graphes REFiLL qui peuvent accepter plusieurs sources de données. La figure 40 montre justement un graphe de filtrage constitué de trois filtres, chacun d'eux disposant de plusieurs entrées (pour la réception des données de plusieurs sources) et de plusieurs sorties pour la diffusion des données (soit vers un autre filtre, soit vers les gestionnaires de sorties).

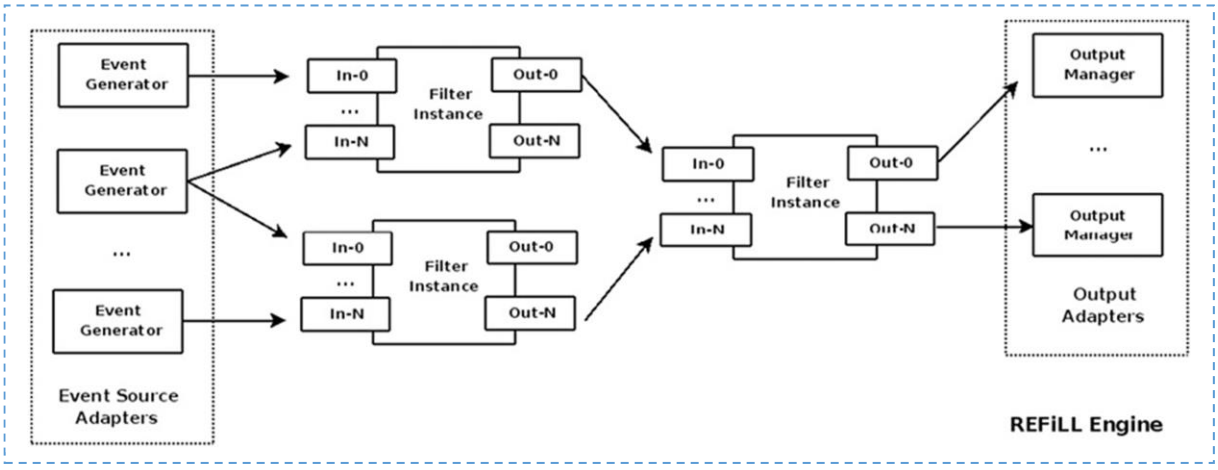


Figure 40. Mécanisme de filtrage du middleware REFiLL [57]

Un graphe de filtrage est un ensemble de composants de filtrage appelés aussi filtres, liés entre eux pour former une structure plus complexe à l'aide du langage FGML (Filter Graph Markup Language). Cette structure est appelée graphe de filtrage (Filter Graph). Un filtre est constitué d'un ou plusieurs processeurs d'événements, chacun d'eux reçoit ses données à partir des ports d'entrée[28].

Les filtres de REFiLL ne sont pas limités au traitement des événements, mais ils peuvent accomplir d'autres actions comme les requêtes de bases de données et les web services. De telles actions permettent de faciliter les échanges avec les applications des entreprises. Les filtres RERiLL sont

[28] L'implémentation de REFiLL interdit que plusieurs processeurs d'événements du même filtre reçoivent leurs données du même port d'entrée ce qui peut conduire à des problèmes de synchronisation.

modélisés de façon à offrir à l'utilisateur la possibilité d'inscrire des instructions (des règles de filtrage) dans des fichiers de définition de filtres (filter definition files) en utilisant le langage FML. Les filtres peuvent être combinés entre eux pour former des composants plus complexes (les graphes de filtrage évoqués précédemment). Cette composition est décrite avec le langage de spécification FGML qui permet de spécifier le nombre de composants (lecteurs, filtres et adaptateurs) et leurs interconnexions. Ceci réduit la programmation (la configuration) des filtres REFiLL essentiellement à la création de fichiers XML.

IV. Techniques de tolérance aux fautes existantes

La RFID est de plus en plus utilisée dans les systèmes critiques notamment dans le domaine médical ou les transports. A titre d'exemple, soit un système de manutention de bagages dans un aéroport. Ce système montre l'importance du bon fonctionnement du lecteur qui communique avec le moteur d'aiguillage (bagage aiguillé vers une destination autre que la sienne) ou bien des autres lecteurs qui servent à localiser les bagages (bagage tombé du tapis et impossibilité de le localiser).

Plusieurs techniques sont apparues pour répondre à ces nouveaux besoins de sûreté de fonctionnement. Dans ce qui suit, nous présentons la sûreté de fonctionnement et ses différentes techniques ainsi que le vocabulaire associé.

1. Terminologie de la sûreté de fonctionnement

La sûreté de fonctionnement est définie comme étant la propriété d'un système informatique qui permet à ses utilisateurs de placer une confiance justifiée dans le service que délivre le système. En d'autres termes, un système sûr est un système qui peut continuer à fonctionner et à remplir ses tâches même en présence de défaillances logicielles ou matérielles.

Dans la terminologie de la sûreté de fonctionnement, les termes « faute », « erreur » et « défaillance » sont utilisés [58]. Ces termes ont une relation de cause à effet comme le montre la figure 41. Une faute est toute cause, événement ou action comme un défaut physique qui peut causer des erreurs qui peuvent affecter le comportement du système. *Ainsi*, une erreur est un état interne du système causé par une faute et qui conduit éventuellement à une défaillance (déviation du comportement normal du système).

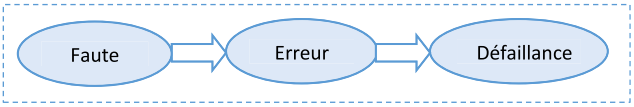


Figure 41. Relation cause à effet

1.1. Objectifs de la sûreté de fonctionnement [59] [60]

Les objectifs de la sûreté de fonctionnement sont classés en six catégories :

- **Fiabilité (Reliability) :** Aptitude d'une entité à accomplir une fonction requise dans des conditions données, durant un intervalle de temps donné.
- **Sécurité (Safety) :** Aptitude d'une entité à ne pas conduire à des accidents inacceptables (ne pas dépasser le niveau "acceptable" de risque).
- **Disponibilité (Availability) :** Aptitude d'une entité ou d'un composant à être en état de marche à un instant donné.
- **Maintenabilité (Maintainability) :** Aptitude d'un composant à être maintenu ou à être remis en état de fonctionnement.
- **Intégrité (Integrity) :** Aptitude d'une entité à n'autoriser la modification de son contenu qu'aux ayants droit et qu'aux parties pour lesquelles ils ont le privilège d'écriture ; *i.e.*, les modifications qui ne sont pas censées arriver ne doivent jamais l'être.

- **Confidentialité (confidentiality, privacy) :** Elle consiste à s'assurer que seules les personnes autorisées à manipuler des ressources données aient accès à ces dernières, donc rendre ces ressources inintelligibles et inaccessibles par d'autres personnes qui ne possèdent pas ce privilège.

1.2. Moyens de la sûreté de fonctionnement

Les moyens de la sûreté de fonctionnement sont classés en trois catégories :

- **Tolérance aux fautes :** Techniques qui tentent de fournir au système la possibilité d'accomplir ses tâches même en présence de fautes[29] (techniques de redondance de composants matériels, logiciels, de données,…).
- **Elimination des fautes :** Techniques dont le but est de réduire le nombre et la gravité des fautes (revue de code, tests unitaires,…)
- **Prévision des fautes :** Prévention contre l'introduction des erreurs (Techniques d'injection des fautes, Analyse des modes de défaillance, leurs effets et leurs criticités AMDEC[30],…).
    - AMDE (Analyse des Modes de Défaillance et de leurs Effets) : L'AMDE, ou FMEA (Failure Modes and Effects Analysis) est une méthode d'analyse de système afin de dégager les causes et les effets des défaillances qui peuvent affecter les composants du système. Nous pouvons aussi retrouver l'AMDEC qui représente une AMDE avec une étude de la criticité ou du niveau de gravité de la défaillance sur le système [61]. Cette technique peut être utilisée : (a) durant le développement du produit, pour prendre en compte la tolérance aux fautes du produit dès les premières étapes de son développement [62]. (b) durant le fonctionnement du produit, pour faciliter sa maintenance et augmenter la disponibilité des services fournis par ce dernier.

  L'AMDE sera appliquée sur les systèmes RFID dans le but d'identifier les défaillances à prendre en compte dans nos mécanismes de tolérance aux fautes proposés. Le résultat de l'AMDE des systèmes RFID est présenté en annexe C.

1.3. Attributs d'une faute

Les attributs utilisés pour la description d'une faute et sa caractérisation sont divers. Les principaux d'entre eux sont [58]:

- Causes de la faute et ses conséquences sur le système.
- Nature (type de de la faute : logicielle, matérielle).
- Durée (durée de la présence de la défaillance).
- Ampleur (localisation et effets sur le reste du système).
- Taux d'apparition (fréquence d'apparition de la faute).

Nous pouvons aussi citer dans ce cadre l'attribut « gravité des conséquences de la faute », appelé aussi « gravité de la défaillance » qui représente l'importance des effets de la défaillance sur le système et son environnement.

1.4. Etape de la tolérance aux fautes

La tolérance aux fautes peut être accomplie en plusieurs étapes [58] :

- **Détection** : découverte de l'erreur. Elle se fait en ligne ou hors ligne et peut se faire de plusieurs façon : utilisation de la redondance dans les données comme les codes détecteurs d'erreurs

[29] Une faute (permanente, transitoire ou intermittente) peut résulter d'une mauvaise conception, d'une mauvaise implémentation, de composants défectueux, de perturbations externes, *etc.*

[30] L'AMDEC peut être utilisée lors de la phase de conception comme aide aux testeurs, ou bien après la mise en marche du système comme aide à la maintenance.

(Cyclic Redundancy Check, Pairing Code …) ou de la redondance des composants du système ; utilisation de la machine à états finis du système pour surveiller et détecter tout écart de comportement normal du système.

- **Masquage** : masquage de l'erreur dès son apparition dans le système avec le mécanisme de redondance. Le niveau de redondance utilisé ici est plus important qu'à l'étape de détection.
- **Localisation** ou **Diagnostic** : détermination du composant défectueux et éventuellement les causes de la défaillance. Cette étape est nécessaire pour permettre l'isolation et la réparation de l'élément défectueux.
- **Isolation** : mise à l'écart du composant ou de la faute afin d'éviter sa propagation. Cette étape utilise les mécanismes de détection et de localisation des composants défaillants.
- **Réparation** : recouvrement et restauration du système à un état opérationnel. Par exemple : dans les systèmes multiprocesseurs, lors de la défaillance d'un processeur, le rétablissement du système se fait par la désactivation définitive ou temporaire de ce dernier. La désactivation temporaire est suffisante dans certains cas où la défaillance est causée par exemple, par la surchauffe du processeur.

1.5. Champs de la sûreté de fonctionnement

Les champs d'application de la sûreté de fonctionnement sont divers. Parmi eux, nous pouvons énumérer les systèmes suivants :

- Systèmes destinés à une longue durée de vie : domaine de l'aéronautique, du spatial…
- Systèmes critiques : domaine médical, ferroviaire, *etc*.
- Systèmes à haute disponibilité : applications bancaires, *etc.*
- Systèmes sans maintenance ou à maintenance difficile : station de traitement à distance, applications embarquées sur les satellites, les voitures, *etc.*

2. Tolérance aux fautes dans les systèmes RFID

Les solutions middlewares RFID comme celles vues dans ce chapitre se concentrent sur le traitement des données RFID brutes et la gestion de leurs sources. Souvent, elles ne considèrent pas la sûreté de fonctionnement de ces systèmes. Parmi les solutions middlewares que nous avons vues, nous pouvons citer le middleware RF²ID [39] qui se base sur des listes de lectures des lecteurs RFID précédents pour déduire s'il y a eu une perte de données ou pas. Il est clair que ce type de monitoring est bien adapté pour la surveillance des mouvements des tags, à savoir l'apparition et la disparition des tags dans le champ du lecteur. En contrepartie, cette technique est moins adaptée pour la détection des erreurs de lecture ou la baisse des performances du lecteur ; *e.g.*, précision de lecture faible, fréquence d'erreurs plus grande que la normale, *etc*. Cela s'explique par le fonctionnement des lecteurs RFID. Un lecteur RFID, lors d'un inventaire des tags, lance plusieurs tentatives de lecture pour identifier un tag. Même si la plupart des tentatives ont échoué (à cause par exemple d'un environnement perturbé par d'autres ondes électromagnétiques), il suffit qu'une seule d'entre elles réussisse à identifier le tag pour que ce dernier fasse partie de la liste des tags identifiés dans l'inventaire. Nous pouvons déduire par-là que RF²ID ne pourra jamais détecter cette baisse de performance du lecteur, ni ses causes.

De manière générale, nous pouvons classer les mécanismes de monitoring des systèmes RFID en deux catégories [63] ; *Monitoring de l'état du lecteur* et *Monitoring des performances du lecteur*.

2.1. Monitoring de l'état du lecteur :

Cette approche se classe dans la catégorie du monitoring intrusif car elle injecte dans le système surveillé des données supplémentaires pour vérifier si le lecteur est allumé, connecté au réseau ou si ses antennes sont opérationnelles. Par exemple nous pouvons vérifier si un lecteur RFID est connecté au réseau par une requête SNMP (Simple Network Management Protocol) :

```
ping "Adresse IP du lecteur"
```

L'inconvénient du monitoring de l'état du lecteur est qu'il ne permet de détecter que des défaillances simples qui concernent l'état du lecteur comme « lecteur non allumé ou non connecté au réseau ». Mais, il ne permet pas de détecter les défaillances relatives aux tags ou bien à l'environnement d'exécution comme l'apparition ou la disparition des tags ou bien une baisse des performances du lecteur (*e.g.*, beaucoup d'erreurs de lecture).

Une deuxième catégorie de monitoring, appelée « monitoring des performances du lecteur » a fait son apparition pour prendre en compte des défaillances plus complexes qui touchent au comportement du lecteur.

2.2. Monitoring des performances du lecteur

Cette approche se classe dans la catégorie du monitoring non-intrusif, car elle n'exploite que des informations déjà présentes dans le système lors de son fonctionnement pour mesurer et surveiller les changements des performances des lecteurs ; *i.e.*, cette technique se base sur les variations des performances d'un lecteur pour vérifier s'il est défaillant ou pas. Dans la littérature, nous pouvons retrouver les paramètres de performances suivant :

- *Read Error Rate* (RER) [64] représente le nombre de lectures erronées (échouées) qui concernent un tag durant un inventaire[31].
- *Read Error to Total Reads* (RETR) [63] représente le nombre de tentatives de lectures échouées sur l'ensemble des tentatives du lecteur pour identifier tous les tags présents dans son champ de lecture (ce paramètre peut représenter plusieurs inventaires de tags).
- *Reading Accuracy* ou *Read Rate* représente le cas inverse du RER. Il désigne le nombre de tentatives de lectures réussies durant un inventaire de tags ou bien sur l'ensemble des tentatives de lectures (qui peut concerner plusieurs inventaires).
- *Average Tag Traffic Volume* (ATTV) [63] représente le nombre moyen de tags qui sont identifiés par un lecteur par unité de temps. Par exemple, si un lecteur identifie une moyenne de 100 tags par heure et qu'un jour, ce taux descend à 20 tags/h, alors cela peut être une indication sur une éventuelle défaillance du lecteur.
- *Read Rate Profile Monitoring* [65] se base sur l'approche « Read Rate » vue précédemment pour établir un profil de lecture entre les tags lus et le lecteur. Le profil est l'ensemble ordonné décroissant des taux de lectures de l'ensemble des tags. Cet ensemble de valeurs constitue une courbe qui représente le profil de lecture associé au couple {*lecteur*, *groupe de tags*}. La figure 42 montre un exemple de profil de lecture associé à un couple {*lecteur*, *groupe de tags*} sous forme d'une courbe (ligne discontinue de couleur verte sur la figure) dont la partie supérieure est formée par les taux de lecture des tags les plus forts du groupe (les tags les plus visibles) et la partie inférieure est formée par les tags les plus faibles (les tags les moins performants). Cette approche nécessite une phase d'apprentissage dans laquelle un profil limite est établi pour le groupe de tags (ligne continue bleue). Un exemple de profil défaillant est donné dans la même figure et représenté par la courbe rouge. Ce profil est considéré défaillant et par conséquence le coupe {lecteur et groupe de tags} qu'il représente l'est aussi, car il ne respecte[32] pas la limite établie à certains endroits (aux alentours de l'index 24 sur la figure 42).

[31] Un inventaire de tags est l'opération durant laquelle, un lecteur RFID essaie d'identifier tous les tags qui se trouvent dans le champ de ses antennes.

[32] Tout profil dont les valeurs à certains index sont inférieures aux valeurs du profil limite est considéré comme étant un profil défaillant.

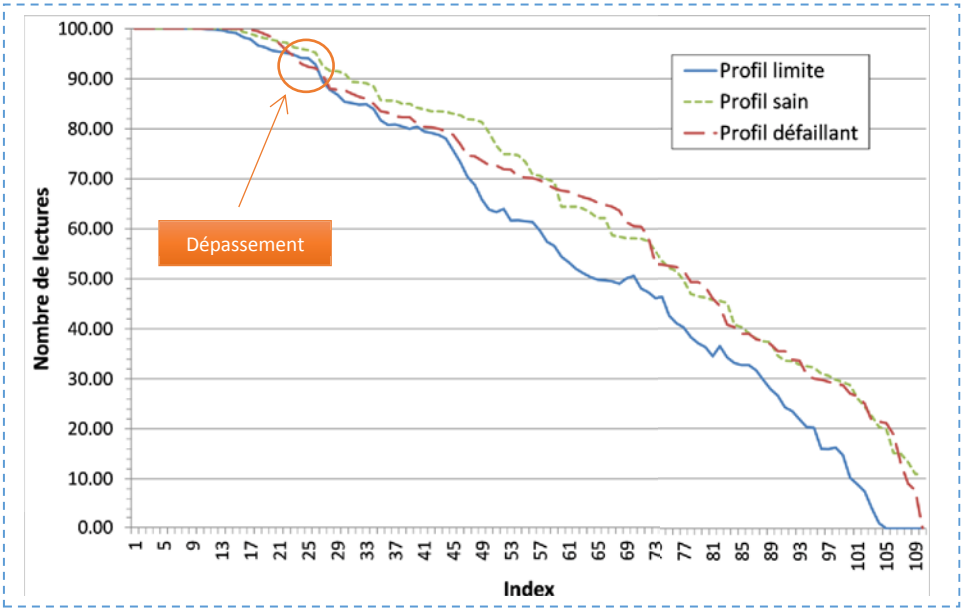


Figure 42. Exemple de profils sain et défaillant d'un groupe de tags

Le monitoring des performances des lecteurs présente plusieurs inconvénients notamment dans un environnement rude où les défaillances peuvent venir de plusieurs sources comme l'usure d'un composant, les perturbations électromagnétiques du médium de communication, *etc*. En effet, cette technique, en plus de présenter l'inconvénient de nécessiter une période d'apprentissage afin de fixer la limite du paramètre de performance que le lecteur ne doit pas dépasser, présente l'inconvénient de ne pas disposer d'une vue globale du système, vu qu'elle surveille chaque lecteur séparément. Ainsi, nous ne pouvons pas savoir avec cette technique si la défaillance vient du lecteur qui n'arrive pas à lire correctement les tags, des tags eux-mêmes ou de l'environnement d'exécution.

V. Comparatif des middlewares précédents

	Normes	Approches de Tolérance aux fautes / fiabilité	Performances	Technologies
WinRFID	- Support de différents standards : EPC, ISO, ICODE… - non compatible avec EPC Network.	- Gestion des données sûre et intelligente - Gestion et surveillance des lecteurs. - Authentification des lecteurs.	- Support d'une variété de lecteurs et tags de différentes fréquences (LF, HF et UHF) - Meilleure interopérabilité grâce à XML. - Adapté pour le traitement de grandes quantités de données.	".NET" et XML pour la représentation des données.
RF²ID	- Support de EPC C1 Gen2. - non compatible avec EPC Network.	- Plus de fiabilité et de disponibilité grâce aux chemins et lecteurs virtuels. - Equilibrage de charge (load balancing). - Comparaison des listes de lecture pour détecter des défaillances.	- Alien ALR - 9800 - Mode "simulation". - Filtrage puissant grâce aux chemins virtuels. - Lenteur dans les traitements (beaucoup de redondance).	Java. Utilisation des sockets pour l'implémentation des VR.
Fosstrak	- Compatible EPC Network	- Fiabilité (il suit les normes EPCglobal Inc. et supporté par Auto-ID Labs). - Surveillance des lecteurs. - Accès distant à la configuration des lecteurs pour maintenance …	- Large gamme de lecteurs. - Mode "simulation".	Java et XML
Aspire RFID	Spécifications EPCglobal, NFCForum, JCP, OSGi Alliance.	- Pas de mécanisme particulier.	- Large gamme de lecteurs.	Java, XML, HTML.
Sun RFID	Prend en charge les grands standards comme ceux d'EPCglobal Inc.	- Pas de mécanisme particulier.	- Large gamme de lecteurs. - Utilisation d'agents de filtrage	J2EE, XML
WebSphère	Implémente une partie des spécifications EPC, mais pas EPC Network.	- Gestion et surveillance des différents composants de la solution.	- Plusieurs types de lecteurs. - Solution destinée pour l'industrie.	J2EE, J2ME, XML, *etc.*
FlexRFID	Plusieurs standards (EPC C1G2, ISO 15693…) Mais pas EPC Network	- Gestion et surveillance des lecteurs.	- Plusieurs types de lecteurs.	XML et autres.
REFiLL	- Compatible EPC Network.	- Composants programmable.	- Filtrage à base de composants programmables. - Large gamme de lecteurs compatible EPC.	Java et XML

VI. Conclusion

Nous avons vu dans ce chapitre une étude des middlewares RFID existants et nous avons effectué une comparaison de leurs caractéristiques. Au cours de cette étude, nous avons introduit la terminologie de la sûreté de fonctionnement et les techniques de tolérance aux fautes dans les systèmes RFID. Cette étude a été nécessaire dans la mesure où nous proposons de concevoir un nouveau middleware RFID qui accueillera les solutions de tolérance aux fautes que nous proposons. Cela nous permet d'identifier les fonctionnalités principales à implémenter et la façon dont notre middleware doit fonctionner.

Dans le prochain chapitre, nous présenterons la première proposition de notre contribution. Cette dernière consiste en une spécification d'une solution middleware compatible avec le protocole LLRP et d'un algorithme de diagnostic probabiliste pour l'analyse des données RFID.

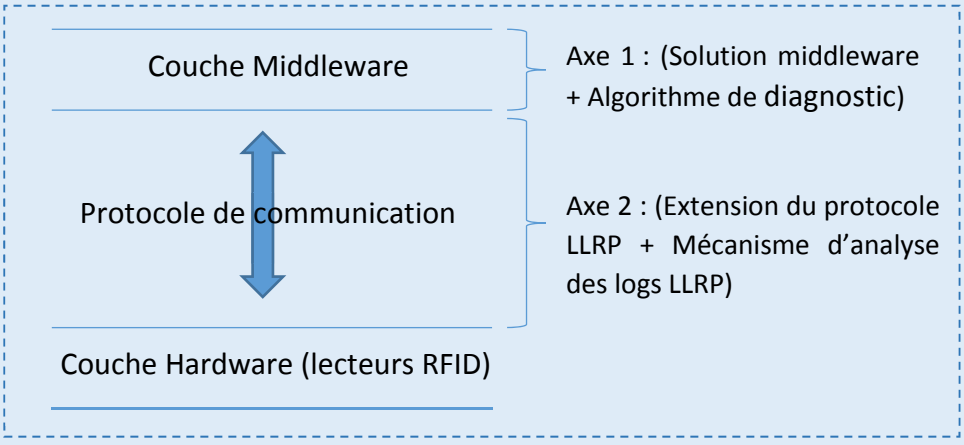
Contribution

Chapitre 3 : Contribution au niveau « Couche Middleware »

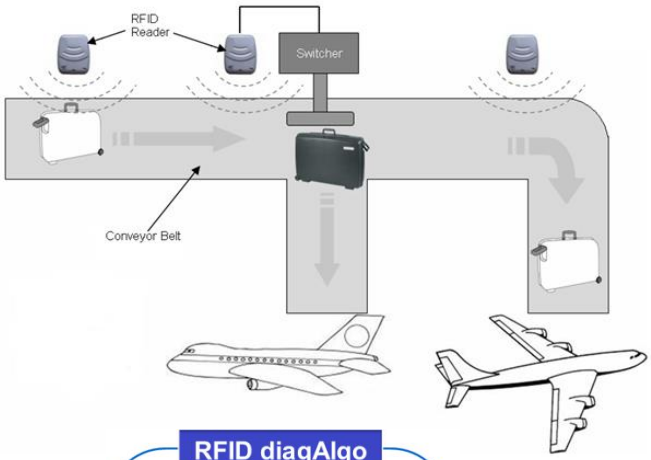
- Middleware RFID « SafeRFID-MW »
- Algorithme de diagnostic probabiliste « RFID diagAlgo »

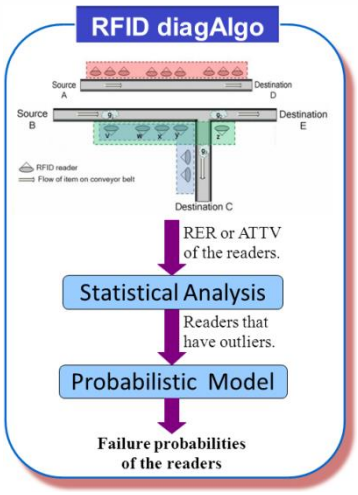
Chapitre 4 : Contribution au niveau « Interface de Communication »

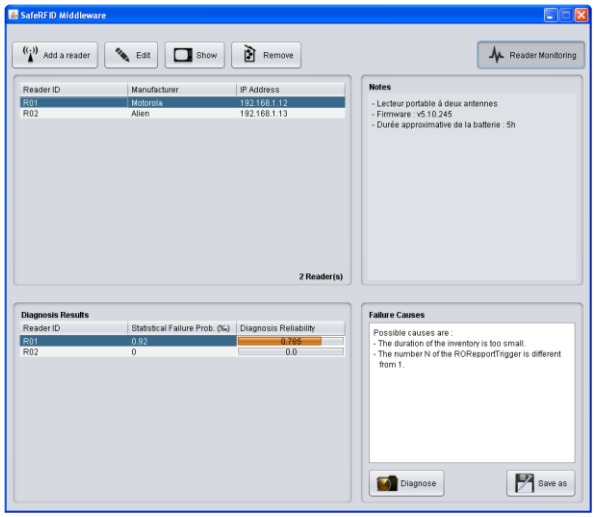
- Extension du protocole LLRP
- Mécanisme d'analyse de logs LLRP



Chapitre 3
Contribution au niveau « Couche Middleware »







Sommaire

Mots Clés

Middleware RFID
SafeRFID-MW
RFID diagAlgo
Sûreté de fonctionnement
Test en ligne
Cas d'utilisation
Algorithme probabiliste
Analyse statistique
Probabilité de défaillance
Identifiabilité
Précision du diagnostic
Correct Négatif
Faux Négatif

CHAPITRE 3 CONTRIBUTION AU NIVEAU « COUCHE MIDDLEWARE »

I. Introduction

La technologie RFID voit son flux de données augmenter de plus en plus et son champ d'application s'étendre vers les domaines critiques tels que le domaine médical, le ferroviaire ou l'aérien où la notion du temps réel est vitale, alors qu'auparavant, cette technologie était essentiellement utilisée pour la gestion des chaînes logistiques. Ainsi, le besoin en sûreté de fonctionnement se fait sentir de plus en plus.

La figure 43 montre un exemple d'utilisation de la technologie RFID pour la gestion et l'acheminement des bagages vers différentes destinations dans un aéroport. Les lecteurs RFID jouent un double rôle dans cet exemple. Ils permettent d'une part, le bon acheminement des bagages, et d'autre part, la localisation de ces derniers. Il est clair que si le système RFID mis en place est défaillant (*i.e.*, lecteur en panne, erreurs de lecture, *etc.*), le système de manutention des bagages ne pourra pas remplir ses fonctions. Ainsi, il serait impossible de localiser un bagage qui serait tombé du tapis roulant ou bien d'acheminer un bagage vers la bonne destination. Ceci introduit la nécessité d'améliorer la fiabilité des systèmes RFID.

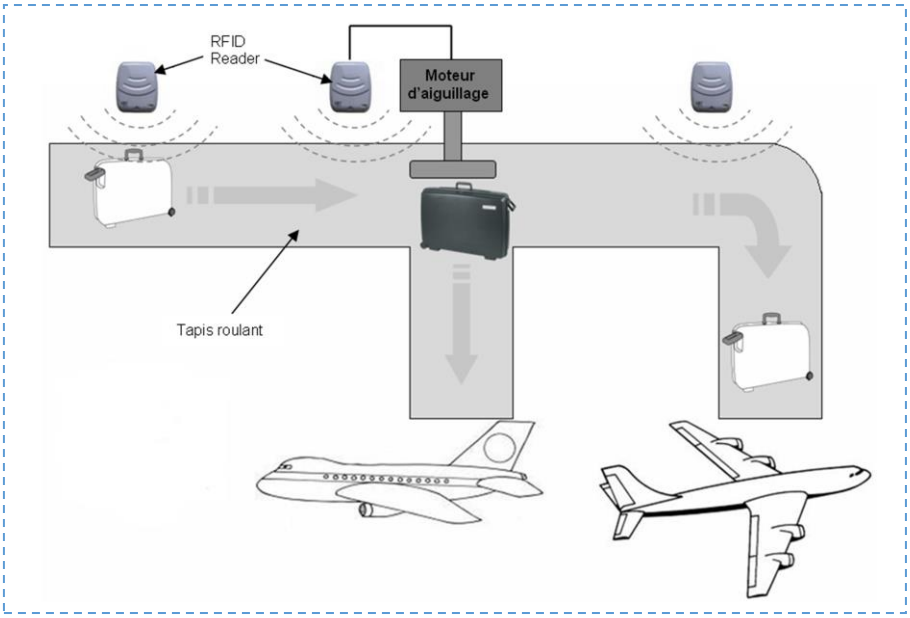


Figure 43. Système de manutention de bagages

Pour faire face à ce nouveau besoin en sûreté de fonctionnement, bien que plusieurs projets aient été déjà menés, plus de travaux de recherche sont nécessaires pour répondre à la demande croissante en tolérance aux fautes de cette technologie. En effet, les techniques présentées dans le chapitre 2 ne sont pas efficaces dans certains cas, spécialement dans des environnements hostiles (usure des composants RFID) ou des environnements distribués où plusieurs sources d'ondes radios

s'entrecroisent ; *i.e.*, les techniques existantes n'ont qu'une vision limitée et non globale du système en surveillant chaque lecteur séparément et indépendamment des autres. Ainsi, lorsqu'un lecteur échoue à lire certains tags, lesdites techniques ne permettent pas d'identifier ou de localiser les composants défaillants, à savoir si la défaillance vient du lecteur[33], des tags ou de l'interface air (présence de perturbations électromagnétiques lors de la communication).

Dans ce chapitre, nous établissons les spécifications d'une solution middleware appelée « SafeRFID-MW » pour la gestion des flux de données RFID et de leurs sources. Nous proposons aussi une solution de tolérances aux fautes intégrée au middleware SafeRFID-MW et qui consiste en un algorithme de diagnostic probabiliste des lecteurs et des tags RFID.

II. Middleware SafeRFID-MW

Le middleware « SafeRFID-MW »[34] [66] se base sur le standard de communication LLRP[35] (Low Level Reader Protocol) pour gérer les lecteurs et récupérer les données RFID. Ce middleware est programmé en langage Java et il est destiné à accueillir et à tester les solutions de tolérances aux fautes que nous proposons.

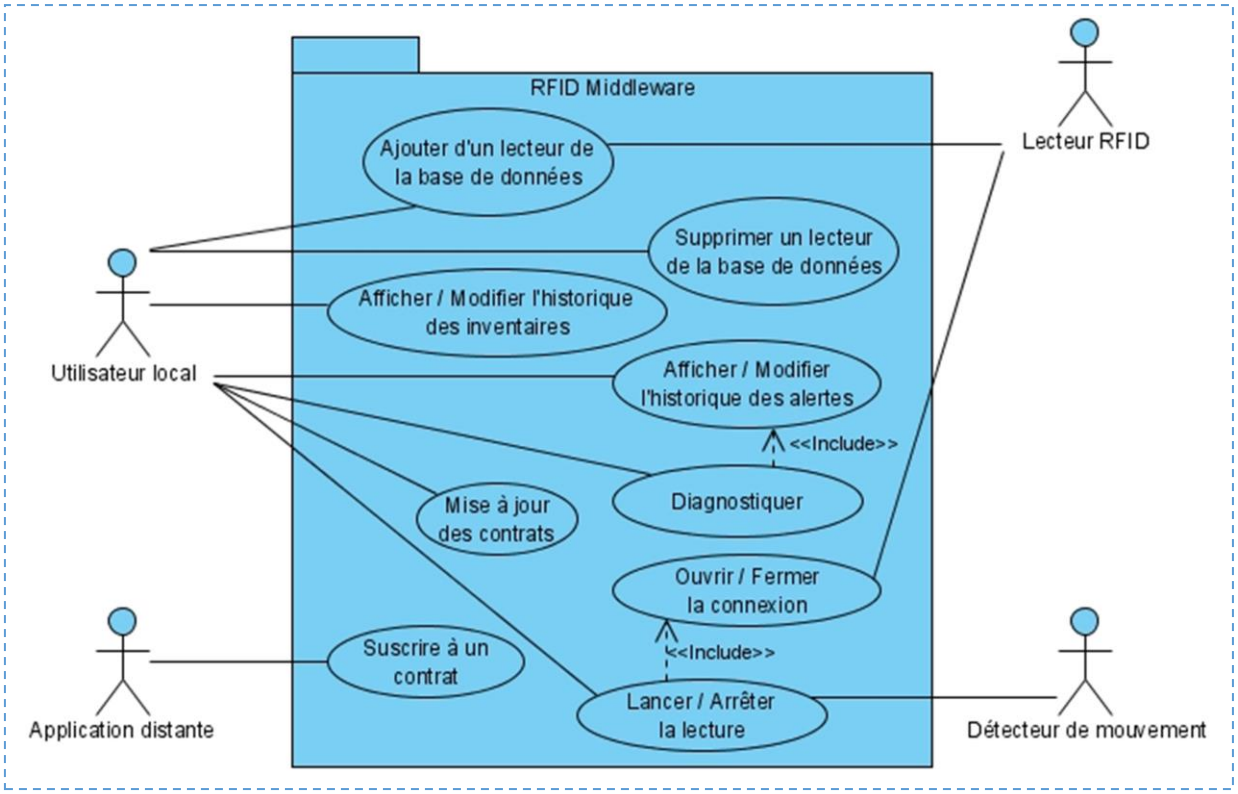


Figure 44. Cas d'utilisation du middleware SafeRFID-MW

[33] Un lecteur défaillant est soit un lecteur hors service soit un lecteur avec de faibles performances.

[34] Ces travaux font partie du projet SAFERFID supporté par l'Agence Nationale de la Recherche ANR (www.agence-nationale-recherche.fr) et dont le but est la proposition d'approches de tolérance aux fautes pour les systèmes RFID.

[35] LLRP est le protocole de communication entre les lecteurs RFID et les applications clientes. Il repose sur l'échange de messages au format XML pour effectuer l'inventaire et l'accès aux mémoires des tags RFID. La compatibilité du middleware SafeRFID-MW avec le protocole LLRP a été testée avec un simulateur RFID « RIFIDI simulator » et un lecteur RFID réel « Motorola MC3190-Z ».

1. Spécification de SafeRFID-MW

1.1. Cas d'utilisation

Les cas d'utilisation présentés dans la figure 44 regroupent toutes les fonctions essentielles d'un middleware RFID. Ils serviront comme fil conducteur à l'élaboration de de notre middleware SafeRFID-MW.

Comme le montre la figure 44, le middleware est en interaction avec quatre acteurs extérieurs.

- **Avec l'utilisateur local :** cet utilisateur est le principal acteur du système, il entre en jeu dans tous les cas d'utilisation du système. Ainsi, il a la possibilité de mettre à jour les données des lecteurs et des inventaires, de définir ou de modifier les contrats[36] entre lui et le middleware ou bien entre une application et le middleware, de lancer des opérations de diagnostic, *etc.*
- **Avec les lecteurs RFID :** les lecteurs RFID auront la possibilité d'ouvrir et de fermer la connexion avec le middleware notamment s'ils implémentent le protocole LLRP. Si un lecteur se connecte pour la première fois, le middleware lui associera une fiche d'information dans la base de données.
- **Avec les applications distantes :** les applications distances auront la possibilité de souscrire à des contrats avec le middleware dans lesquels seront définis tous les besoins de ces applications (règles d'échange, type de lecteur, nombre de lecteurs, type et format de données, *etc.*).
- **Avec les détecteurs de mouvement :** le middleware aura aussi à s'interfacer avec d'autres appareils notamment les détecteurs de mouvement. Cela permet surtout un gain d'énergie, vu que le middleware ne déclenche la lecture qu'à la réception du signal du détecteur.

1.2. Diagramme de classes du middleware RFID

La figure 45 représente l'architecture globale de notre middleware RFID sous forme de quatre composants principaux.

1.2.1. Composant d'interaction avec les données (DataInteraction)

Le composant d'interaction avec les données est en charge du traitement des données brutes, de leur protection et de leur sauvegarde. Il contient 6 classes principales :

- *MiddlewareSafeRFID* représente la classe principale. Elle gère tout le système et s'occupe de lancer les différentes tâches.
- *DataTransformation* et *ThreadHandleTag* pour l'extraction et la transformation des données.
- *DataFiltering* pour le filtrage des données.
- *DataBaseManagement* pour la gestion et le stockage des données.
- *DataProtection* pour la protection des données et la gestion des droits d'accès.

1.2.2. Composant d'interaction avec les lecteurs RFID (ReaderInteraction)

Le composant d'interaction avec les lecteurs est responsable des échanges de données avec ces derniers. Il est composé de 5 classes :

- *DataReader* pour la l'inventaire et l'accès en lecture aux mémoires des tags.
- *DataWriter* pour l'accès en écriture aux mémoires des tags.
- *HAL* (Hardware Abstraction Layer) pour l'abstraction du matériel RFID.
- *BufferR et BufferW* pour le stockage temporaire des données lues ou à écrire.

[36] Un contrat regroupe toutes les informations dont l'entité qui l'a défini a besoin. A titre d'exemple, une application peut définir un contrat avec le middleware dans lequel elle spécifie le nombre, le type ou l'emplacement des lecteurs RFID qui vont lui être associés, ou tout simplement définir le type des données ou des produits qui l'intéressent, *etc*.

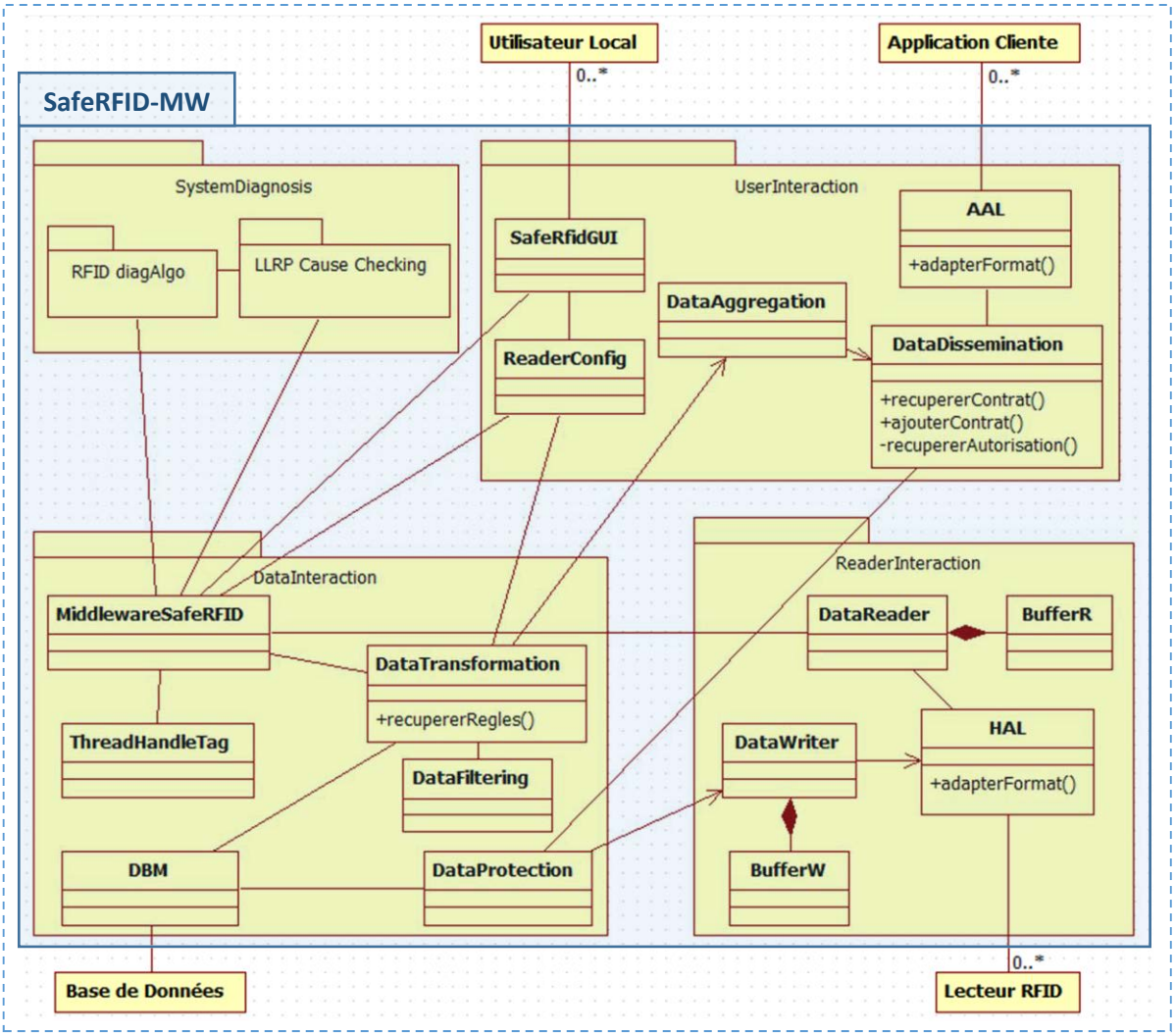


Figure 45. Diagramme de classes du middleware SafeRFID-MW

1.2.3. Composant d'interaction avec les utilisateurs (UserInteraction)

Le composant d'interaction avec les utilisateurs est en charge des échanges de données avec les clients (personnes ou applications), notamment en définissant des contrats de données avec eux. Il est composé de 5 classes principales :

- *DataDissemination* pour la diffusion des données suivant les contrats établis et les règles de sécurité (*i.e.*, règles de confidentialité des données gérées par la classe *DataProtection*).
- *DataAggregation* pour l'agrégation et la classification des données.
- *AAL* (Application Abstraction Layer) pour l'interprétation et l'adaptation des données des utilisateurs finaux.
- *SafeRfidGUI* représente l'interface principale de communication avec les utilisateurs externes. Elle permet l'ajout et la suppression des lecteurs, le lancement du processus de diagnostic, *etc*.
- *ReaderConfig* est l'interface de gestion et de communication avec un lecteur RFID. Elle a comme tâche de lancer les opérations de lecture ou d'écriture sur les tags et d'afficher les résultats de ces opérations, ainsi que les performances du lecteur.

Exemple : une application qui veut souscrire ou établir un nouveau contrat, envoie une requête à la classe *AAL* qui peut éventuellement adapter son format et la transmettre vers la classe *DataDissemination* qui vérifie les droits d'accès auprès de la classe *DataProtection*. Cette dernière récupère les règles de sécurité de la base de données à travers la classe *DBM*. Une fois l'autorisationdonnée, la classe *DataDissemination* crée le contrat et le stocke dans la base de données.

1.2.4. Composant de surveillance du système (SystemDiagnosis)

Le composant de surveillance est en charge d'assurer la sûreté de fonctionnement du système. Il intègre deux sous-composants

- RFID diagAlgo : il assure la détection des défaillances par une comparaison statistique des données des lecteurs. Il sera traité en détails au chapitre 3.
- LLRP cause checking : il assure l'identification des causes de la défaillance à travers la vérification des valeurs des paramètres du protocole LLRP. Ce composant sera traité au chapitre 4.

1.3. Diagramme de séquences de SafeRFID-MW

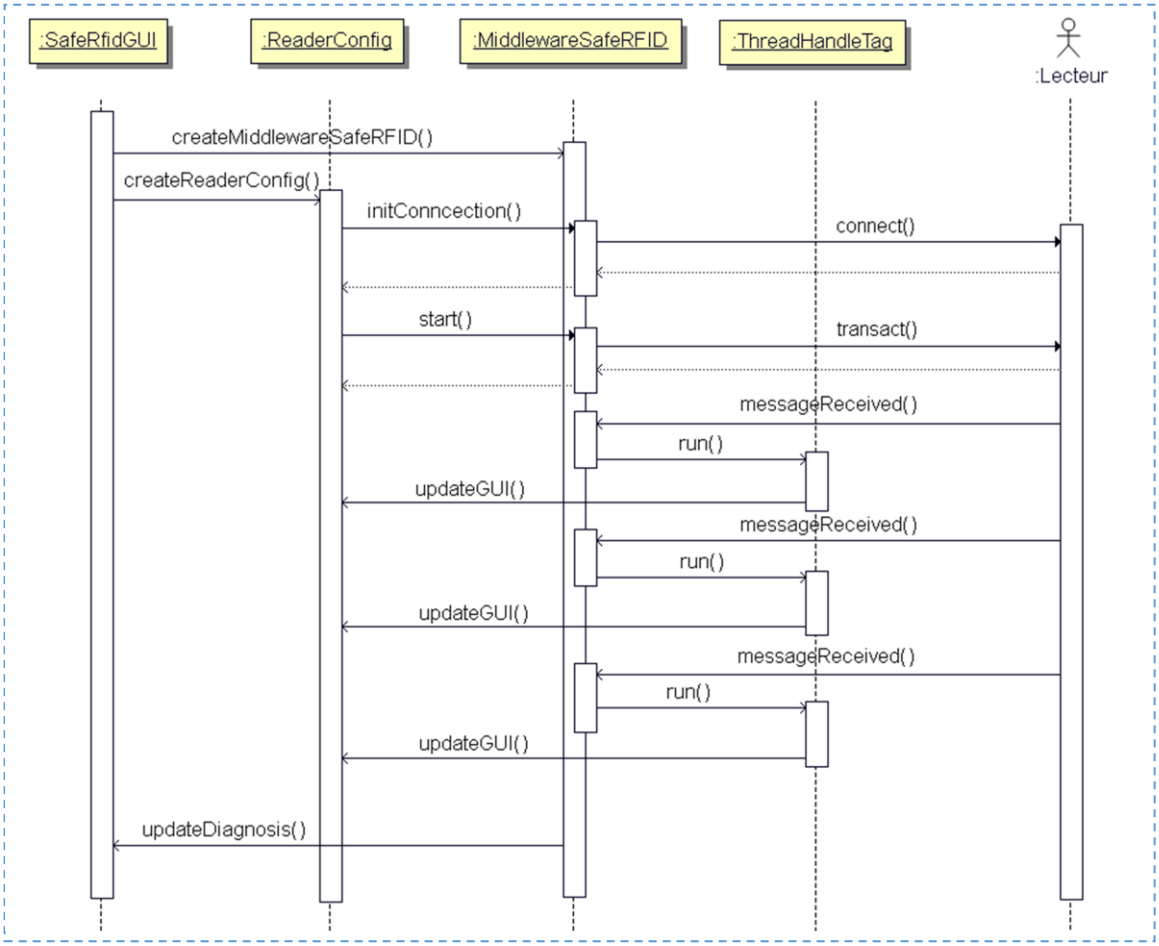


Figure 46. Diagramme de séquences de SafeRFID-MW

La figure 46 montre un exemple de comportement du middleware SafeRFID-MW. L'exécution de SafeRFID-MW commence par le lancement de son interface graphique SafeRfidGUI qui permet de créer des instances de ReaderConfig pour la gestion de chaque lecteur RFID. ReaderConfig permet d'établir une connexion LLRP avec un lecteur à travers la méthode *initConnection()* et de lancer les opérations de lecture/écriture sur les tags. A chaque fois que le lecteur veut transmettre un message au middleware, il appelle de façon asynchrone la méthode *messageReceived()* de la classe MiddlewareSafeRFID. Cette dernière instancie un thread pour le traitement du message reçu et la mise à jour des informations des tags et du lecteur sur l'interface graphique ReaderConfig. La classe MiddlewareSafeRFID permet aussi de mettre à jour les informations de diagnostic du lecteur à travers la méthode *updateDiagnosis()* en faisant appel à d'autres classes de diagnostic qui ne sont pas présentées dans le diagramme de la figure 46 par souci de clarté.

2. Prise en main de SafeRFID-MW

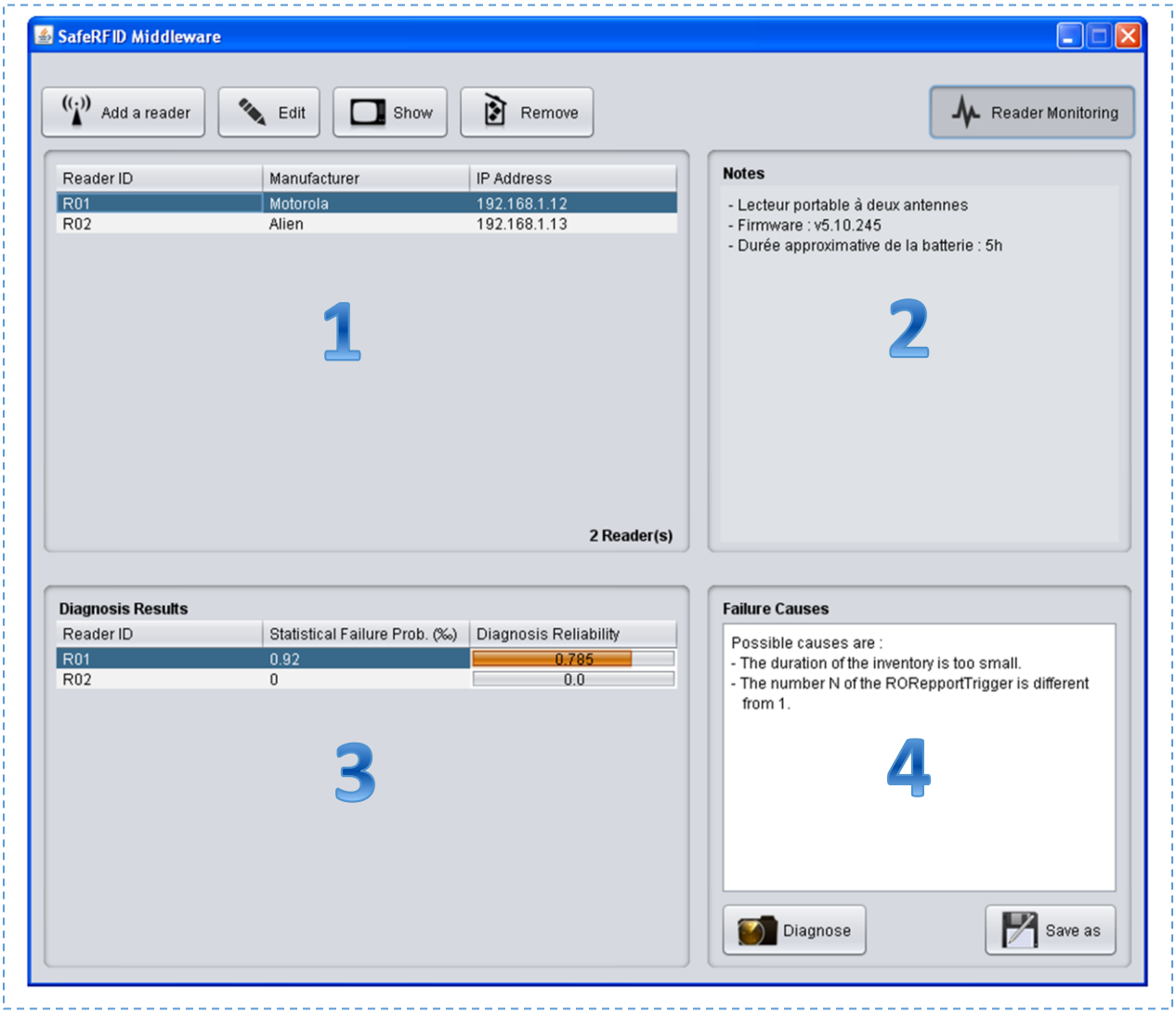


Figure 47. Interface principale de SafeRFID-MW

La figure 47 montre l'interface principale de SafeRFID-MW qui est composée de quatre parties. La première partie (cadre 1 de la figure) liste tous les lecteurs RFID utilisés. La deuxième (cadre 2) donne les informations et les notes saisies sur chaque lecteur. La troisième partie (cadre 3) présente les différents

résultats de diagnostic des lecteurs utilisés, *i.e.*, quels sont les lecteurs défaillants et avec quelles probabilités. Enfin, la dernière partie (cadre 4) présente les causes possibles de la défaillance. Les parties 3 et 4 sont reliées respectivement à un algorithme de diagnostic probabiliste et un analyseur de fichiers log qui seront présentés par la suite.

La figure 48 montre l'interface de gestion de chaque lecteur. Cette interface permet à l'utilisateur de charger différents messages LLRP (fichiers XML) pour effectuer l'inventaire des tags et l'accès aux mémoires des tags. Cette interface donne à l'utilisateur les résultats de lecture des tags ainsi que d'autres informations comme l'heure à laquelle chaque tag a été vu pour la première et dernière fois, le nombre de fois où chaque tag a été vu et les performances de lecture de chaque tag. Cette fenêtre présente aussi les performances de lecture du lecteur RFID utilisé.

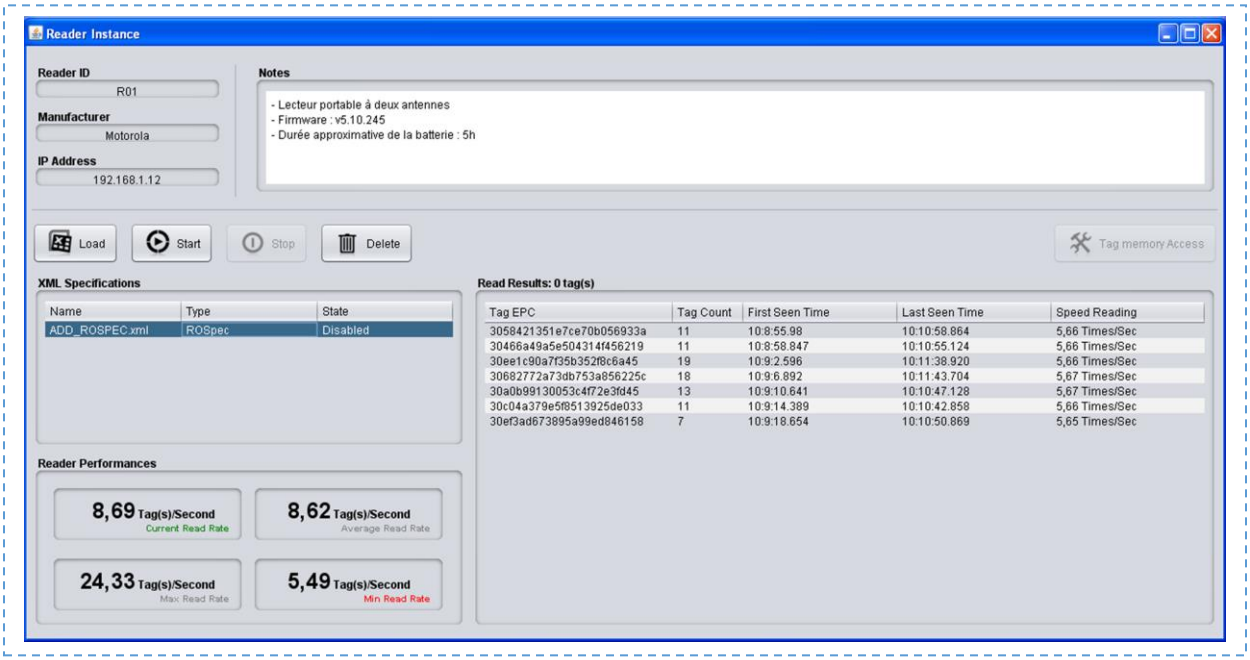


Figure 48. Interface de gestion d'un lecteur RFID

SafeRFID-MW propose aux utilisateurs avancés ou aux administrateurs du système RFID une autre interface alternative. Cette interface présentée en figure 49 permet à l'utilisateur de travailler directement sur les messages XML (messages LLRP) qui sont échangés entre le middleware et le lecteur. Cette approche permet à l'utilisateur d'exploiter toutes les possibilités offertes par le protocole LLRP comme récupérer les capacités d'un lecteur, modifier sa configuration, activer ou désactiver différents types de messages d'erreurs, *etc*.

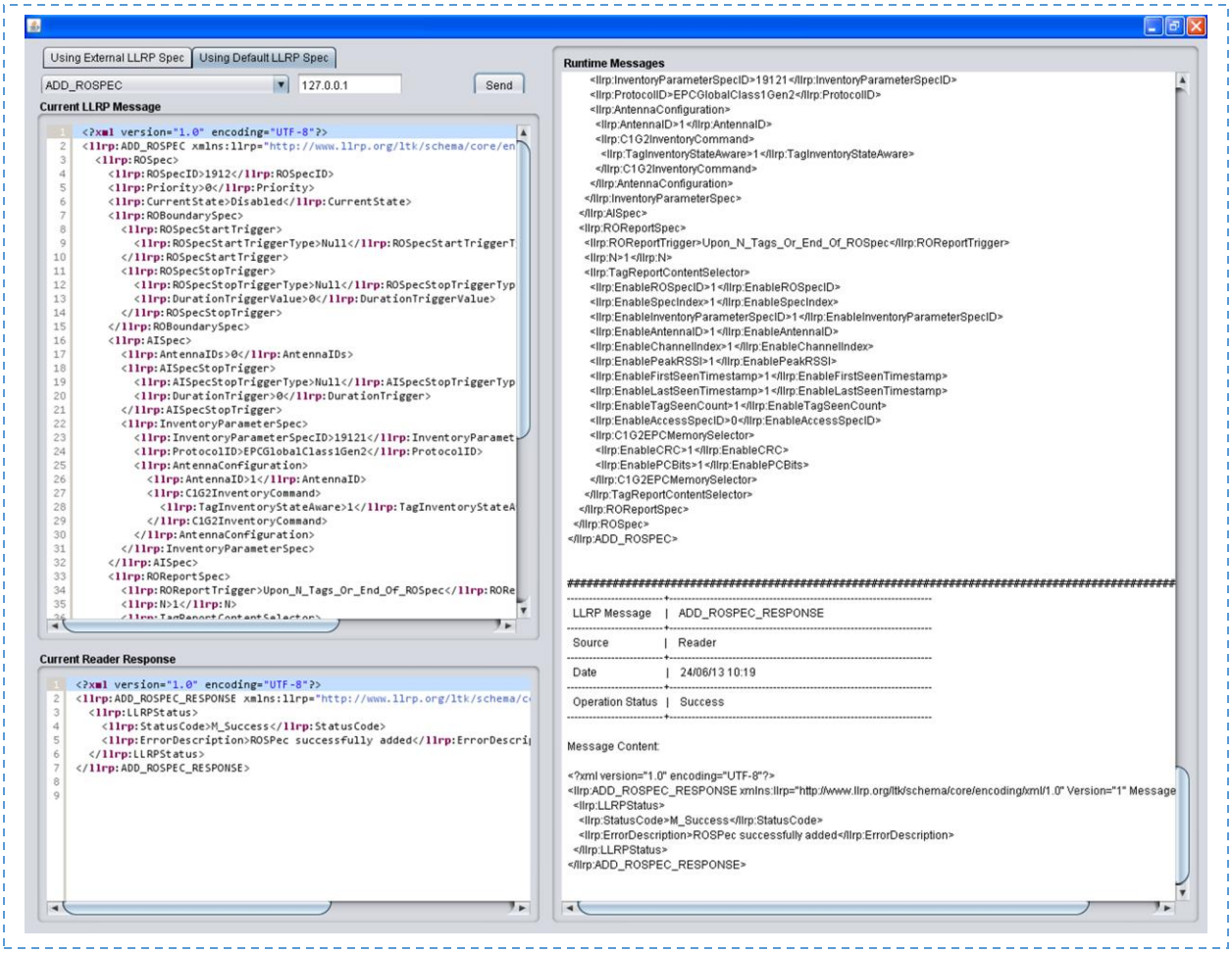


Figure 49. Interface de contrôle avancé de SafeRFID-MW

III. Algorithme de diagnostic probabiliste « RFID diagAlgo »

Les systèmes RFID sont très similaires aux systèmes multiprocesseurs distribués dans leur structure. Les lecteurs sont connectés dans un chemin logique grâce au flux de données RFID, tout comme les processeurs sont connectés physiquement dans le système pour traiter les mêmes données. Dans les systèmes distribués, nous utilisons la redondance physique pour surveiller le système comme l'ont décrit D. Fussell et S. Rangarajan dans leurs travaux sur le diagnostic des systèmes multi-processeurs [67] [68]. Ils ont utilisé un modèle de comparaison pour calculer la propabilité de défaillance de chaque processeur. Sur la base de ces similarités, nous proposons une approche semblable qui utilise la variation des performances des lecteurs pour surveiller les systèmes RFID. Cette approche est bien adaptée pour les déploiements à large échelle de la RFID comme les systèmes de manutention de bagages dans les aéroports où chaque bagage (tag RFID) dispose d'un chemin bien défini et donc d'un ensemble de lecteurs qui doivent l'identifier avant qu'il atteigne sa destination.

L'approche proposée consiste en un algorithme de diagnostic probabiliste appelé *RFID diagAlgo*. Cet algorithme se déroule en trois étapes comme le montre la figure 50. La première étape vise à regrouper les lecteurs suivant les groupes de tags qu'ils sont amenés à lire. Le fait d'organiser les lecteurs en groupes facilite dans la deuxième étape, la comparaison statistique des différents résultats de lecture. Cette comparaison permet d'identifier les tags ou les lecteurs qui sont « potentiellement » défaillants en

se basant sur la variation des performances de lecture des lecteurs RFID. Dans la troisième étape, l'algorithme *RFID diagAlgo* estimera la précision du diagnostic (fiabilité du diagnostic) à chaque détection d'une défaillance.

Figure 50. Etapes de l'algorithme RFID diagAlgo

1. Modèle de faute

1.1. Défaillances RFID

Les défaillances auxquelles nous nous intéressons peuvent être permanentes ou temporaires et varient d'un composant hors service (tag ou lecteur cassé) à un composant endommagé qui ne bloque pas complètement le fonctionnement du système mais réduit ses performances (*e.g.*, fréquence d'erreurs de lecture plus élevée, vitesse de lecture plus faible, *etc.*). Ces défaillances sont classées en trois catégories :

- L'apparition ou la disparition de tags (*e.g.*, un tag qui est lu par un lecteur et non par un autre).
- La non-réponse d'un lecteur (*e.g.*, le lecteur est hors service).
- La baisse de performance du lecteur (*e.g.*, une des antennes du lecteur est abîmée, perturbations externes).

1.2. Application de l'approche proposée

RFID diaAlgo peut être utilisé pour analyser les tags séparément ou en groupe. Dans ce qui suit, nous présentons et discutons les deux approches.

1.2.1. Traitement individuel des tags

Dans ce cas, nous nous intéressons à l'identifiant du tag que chaque lecteur a lu et nous considérons les situations suivantes :

- La majorité des lecteurs identifie correctement le tag : le tag est considéré valide et tous les lecteurs qui ne l'ont pas identifié sont considérés défaillants.
- La majorité des lecteurs n'a pas identifié correctement le tag : ces lecteurs sont déclarés
  i. défaillants (même s'ils représentent la majorité) : si le nombre des lecteurs restants (la minorité) est supérieur à deux et que ces lecteurs ont identifié le même tag. En effet, la probabilité p d'avoir N lecteurs défaillants ($N \geq 2$) qui identifient le même tag au même moment, sachant que l'identifiant EPC du tag est sur 96 bits est quasiment nulle ($p = \frac{1}{2^{96} \times N}$).
  ii. valides : dans ce cas, si la minorité est réduite à un lecteur (un seul lecteur qui retourne un tag ID[37]), le tag et ledit lecteur sont déclarés défaillants et les autres lecteurs (la majorité) sont considérés sains.

Le principal défaut de cette approche est que l'analyse des tags séparément ne permet pas la détection des défaillances dont la conséquence est une réduction des performances du lecteur ; *e.g.*, un lecteur dont l'antenne est endommagée peut continuer à identifier des tags mais avec des taux de lecture faibles à cause de la réduction de la puissance du signal radio émis et reçu par son antenne.

[37] L'identifiant retourné peut ne pas s'agir du vrai identifiant du tag, mais un identifiant resté dans la mémoire du lecteur.

1.2.2. Traitement des tags par groupe

Ce cas peut être vu sous deux angles :

- Nous pouvons considérer l'analyse de la liste des tags observés par chaque lecteur. Nous retrouvons cette situation dans le middleware RF²ID [39] vu au chapitre 2. Mais cette approche présente les mêmes défauts que le traitement individuel des tags ; elle ne permet pas de détecter les défaillances qui ne bloquent pas complètement les opérations de lecture/écriture mais qui affectent les performances du système.
- Nous pouvons considérer la variation d'un paramètre de performance donné suivant chaque groupe de tags. Cette approche nécessite une période d'apprentissage pour déterminer les valeurs références du paramètre de performance choisi. Des exemples de paramètres de performance sont donnés au chapitre 2.
    i. Si la majorité des lecteurs respectent le paramètre de performance défini, alors le reste des lecteurs (la minorité) sont considérés défaillants ; *i.e.*, la minorité qui montre des performances faibles par rapport à la normale est considérée défaillante.
    ii. Si la majorité des lecteurs ne respectent pas le paramètre de performance défini sur un groupe de tags, alors ce groupe, ainsi que les autres lecteurs (la minorité) sont considérés défaillants (*i.e.*, principe du vote de la majorité).

Le traitement des tags en groupe permet de surveiller les performances de chaque lecteur et ainsi de détecter toute baisse de performance. En contrepartie, cette approche ne permet pas de détecter la disparition ou l'apparition de certains tags si leur nombre n'est pas assez significatif pour affecter le paramètre de performance surveillé. Elle ne permet pas non plus de localiser les tags défaillants au sein du groupe.

Dans le cas idéal, il est préférable de surveiller les lecteurs en tenant compte de leurs listes de tags lus et de leurs performances de lecture. Mais dans ce qui suit et pour simplifier la démarche proposée, nous ne parlerons que de paramètres de performances pour la surveillance des systèmes RFID.

Les deux approches de traitement des tags (à savoir « traitement individuel ou en groupe des tags ») que nous venons de voir, gardent toujours le défaut de ne pas pouvoir localiser la source de la défaillance (lecteur, tag ou interface air). Ainsi, nous proposons de surveiller les lecteurs eux aussi en groupe de manière à pouvoir comparer leurs résultats de lecture.

2. Description de l'algorithme « RFID diagAlgo »

Le fonctionnement de *RFID diagAlgo* est résumé en figure 51. (1) *RFID diagAlgo* regroupe les lecteurs suivant les groupes de tags que ces derniers sont amenés à lire. Ensuite, (2) *RFID diagAlgo* effectue une comparaison des différents résultats de lecture afin d'identifier les tags ou les lecteurs qui sont « potentiellement » défaillants. Enfin, (3) *RFID diagAlgo* associe une probabilité de défaillance appelée aussi « précision du diagnostic » aux éléments aberrants suivant un modèle probabiliste adapté des travaux de D. Fussell et S. Rangarajan sur le diagnostic des systèmes multiprocesseurs [67] [68]. *RFID diagAlgo* a une complexité de $O(t \times 2n)$ [38] dans la version présentée dans l'annexe D. Nous avons délibérément choisi cette représentation (sans optimisation) pour plus de clarté. A titre indicatif, le temps d'exécution de *RFID diagAlgo* en faisant abstraction du délai[39] nécessaire pour l'obtention de tous les résultats des lecteurs ne dépasse pas 100 *millisecondes* (néanmoins, cela varie selon la configuration matérielle du système hôte).

[38] *n* et *t* sont respectivement le nombre total des lecteurs en cours d'analyse et le nombre total des groupes de tags.
[39] Ce délai dépend de la distance qui sépare chaque lecteur des autres et de la vitesse de déplacement des tags.

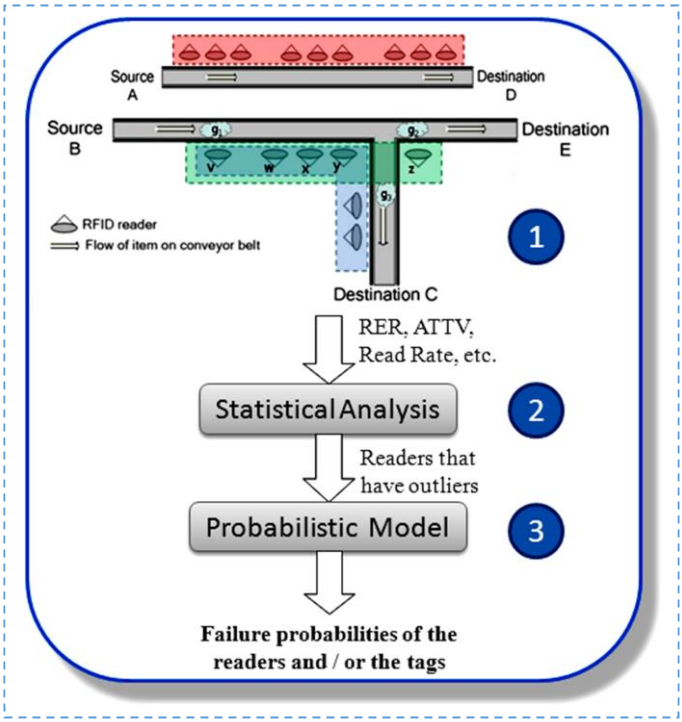


Figure 51. Algorithme de Diagnostic probabiliste

2.1. Partitionnement des lecteurs en groupes

Si nous reprenons l'exemple du système de manutention de bagages dans un aéroport, nous pouvons partitionner les lecteurs qui se trouvent le long du tapis roulant suivant le chemin que les différents bagages prennent pour atteindre leur destination. La figure 52 montre un partitionnement possible des lecteurs. Ainsi, tous les lecteurs qui sont situés sur le chemin {*A, D*} seront regroupés ensemble (*zone rouge sur la figure*). Ce partitionnement est naturel, d'autant plus que chaque bagage dispose d'une destination et donc d'un chemin bien spécifique à suivre. De ce fait, *RFID diagAlgo* connaît tous les lecteurs qui doivent identifier ce bagage et ainsi, sont évités les cas où l'algorithme peut prendre le silence d'un lecteur à propos d'un ou plusieurs bagages pour une défaillance. Ce partitionnement sera aussi utilisé pour comparer les résultats des lecteurs à l'intérieur d'un même ensemble. Mais lors du processus de comparaison des résultats, différents ensembles de lecteurs peuvent y participer (*i.e.*, un lecteur peut appartenir à des groupes différents suivant les bagages (les tags) qu'il analyse).

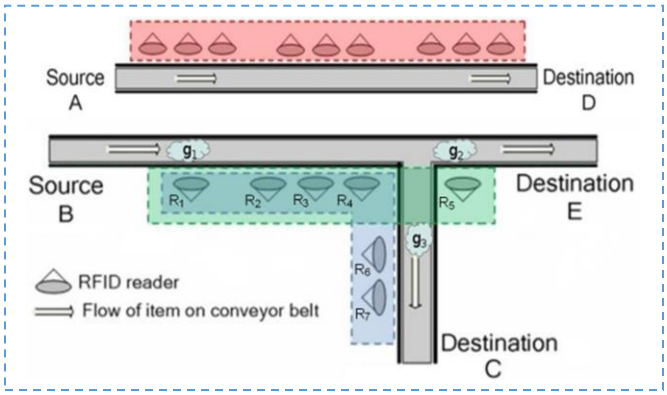


Figure 52. Partitionnement des lecteurs en groupes

2.2. Comparaison des résultats des lecteurs

Dans cette étape, la comparaison ne se fait pas directement sur les résultats de lecture de chaque lecteur mais plutôt sur un des paramètres de performance des lecteurs. Par abus de langage, nous utiliserons l'expression « résultats de lecture » pour indiquer si les tags ou les groupes de tags analysés respectent bien le paramètre de performance choisi. Ce paramètre peut être :

- *Read Error to Total Read Rate* [69] : représente le ratio entre le nombre d'erreurs de lecture et le nombre total de tentatives de lecture de tous les tags.
- *Read Error Rate*[40] [70] : représente le ratio entre le nombre d'erreurs de lecture et le nombre total de tentatives de lecture de chaque tag.
- *Approche par Profil* [65] : paramètre proposé dans le cadre du projet SAFERFID. Ce paramètre concerne l'analyse d'un ensemble de tags en même temps (*e.g.*, une palette de produits). Chaque ensemble ou groupe de tags se voit assimiler un profil de lecture pour chaque lecteur. Ce profil est représenté par une courbe qui correspond à l'ensemble des taux de lecture de chaque tag de ce groupe. Dans la suite de cet article, nous utiliserons cette approche pour la surveillance des lecteurs et des groupes de tags.

Si la valeur de ce paramètre dépasse une limite prédéterminée ou bien représente une valeur aberrante par rapport à celle des autres lecteurs, le lecteur est déclaré *potentiellement défaillant*.

Table 5. Comparaison des résultats de lecture

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

Exemple : reprenons l'exemple de la figure 52, où nous avons 7 lecteurs formant l'ensemble $R = \{R_1, R_2, R_3, R_4, R_5, R_6, R_7\}$ et *3* groupes de tags $G = \{g_1, g_2, g_3\}$. La table 5 montre le résultat de la comparaison des profils de lecture de tous les groupes de tags.

- Nous appellerons *M* la matrice des résultats de lecture, représentée par la table 5 sans la dernière ligne ni la dernière colonne.
- $M(R_n, g_t) = 1$ (*resp.*, $M(R_n, g_t) = 0$) indique que le groupe de tags g_t respecte (*resp.*, ne respecte pas) son profil lorsqu'il a été analysé par le lecteur R_n.
- g_1 et g_2 ne sont pas analysés par R_6 et R_7 car ils empruntent un chemin autre que celui où ces lecteurs se trouvent (idem pour g_3 par rapport à R_5). Cela montre l'utilité de la première étape de notre algorithme qui évite de considérer le silence de certains lecteurs sur certains groupes de tags comme une défaillance.
- $S(g_t)$ est une fonction qui répertorie les lecteurs et groupes de tags dont les résultats sont aberrants (par rapport à celui de la majorité) lors de l'analyse du groupe g_t. ; *e.g.*, la majorité des lecteurs disent que g_1 respecte bien son profil d'origine sauf le lecteur R_2 ; donc R_2 est considéré

[40] Ce paramètre est utile pour la surveillance individuelle des tags RFID.

défaillant. $S(g_t)$ identifie les lecteurs défaillants à chaque analyse d'un groupe de tags. Cette défaillance peut être temporaire ou intermittente ; *e.g.*, R_2 est déclaré défaillant sur 2 groupes de tags et R_7 sur un seul groupe. $S(g_t)$ permet aussi d'indiquer si le groupe de tags analysé est défaillant ou pas.

- *SF* permet de répertorier tous les lecteurs qui sont considérés comme défaillants après l'analyse de tous les groupes de tags. Sur notre exemple, un lecteur est déclaré défaillant s'il est déclaré défaillant au moins sur un groupe de tags. Nous pouvons varier le nombre de fois où un lecteur doit être déclaré défaillant dans $S(g_t)$ pour qu'il soit réellement considéré défaillant. Cela permet notamment de réduire les faux positifs.

2.3. Calcul de la précision du diagnostic

Cette approche a pour rôle de valider les décisions prises dans l'étape précédente sur l'état des lecteurs et des tags en estimant la fiabilité du diagnostic à chaque détection d'une défaillance. Cette précision du diagnostic qui dépend de la configuration courante du système peut aussi être considérée comme la probabilité de défaillance du composant RFID. La précision du diagnostic lors de la phase de comparaison des résultats revient à respecter les deux cas suivants.

- Un lecteur valide doit être identifié comme valide. Nous appellerons ce cas "*Correct Négatif* (CN)".
- Un lecteur défaillant doit être identifié comme défaillant. Pour des raisons de simplicité, nous nous intéressons au cas inverse; *i.e.*, le cas où un lecteur défaillant est considéré comme valide. Ce cas sera désigné par "*False Négatif* (FN)".

Soit l'**Identifiabilité**[41] la probabilité d'identifier correctement l'état des lecteurs diagnostiqués. Cette probabilité représente la précision du diagnostic et elle est donnée par la formule probabiliste suivante :

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

Où :

- $CN(t, n, p, r)$ et $FN(t, n, p, r)$ sont respectivement la probabilité d'avoir un correct négatif et la probabilité d'avoir un faux négatif. Elles seront traitées aux sections III.2.3.1 et III.2.3.2.
- n est le nombre total des lecteurs.
- t est le nombre de groupes de tags.
- p est la probabilité initiale de défaillance d'un lecteur[42]. Dans l'algorithme présenté en annexe D cette probabilité est propre à chaque lecteur et est égale au nombre de fois où le lecteur est déclaré défaillant dans le processus de comparaison des résultats sur le nombre total des groupes de tags analysés.
- r (*Rational Behavior)* est la probabilité que la défaillance d'un lecteur se manifeste. En informatique, un programme contenant des erreurs ne devient défaillant que lorsque le chemin d'exécution qui contient l'erreur est emprunté. r représente donc, l'estimation de ce phénomène. Une grande valeur de r indique que la plupart, voire toutes les erreurs du programme ont des effets visibles sur le système. Nous reviendrons sur ce paramètre lors de l'implémentation de *RFID diagAlgo* au chapitre 5.

2.3.1. Correct Négatif (CN)

Les corrects négatifs sont représentés par la probabilité $CN(n, t, p, r)$. Cette dernière désigne le cas

[41] L'*Identifiabilité* est la capacité de notre l'algorithme *RFID diagAlgo* à « bien distinguer » les lecteurs défaillants des lecteurs non défaillants.

[42] Pour ne pas alourdir les différentes formules probabilistes, nous considérons que les lecteurs ont la même probabilité de défaillance.

où un lecteur valide est considéré comme tel dans la phase de comparaison des résultats. Pour cela, nous distinguons les deux cas suivants :

- La majorité des lecteurs (autres que celui en cours d'analyse) sont valides (*i.e.*, leur nombre est entre $n/2$ et $n-1$ lecteurs) et donc, forcément leurs résultats concordent avec ceux du lecteur en cours d'analyse. Ce cas est représenté par

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

C(x, y) désigne le nombre de combinaisons de x éléments parmi y éléments.

- La majorité des lecteurs sont défaillants, mais sur chaque groupe de tags, il y a un certain nombre d'entre eux qui ont des résultats corrects (*i.e.*, ils ont un comportement non défaillant) de façon à ce que le nombre de lecteurs (défaillants ou pas) en accord avec le lecteur en cours d'analyse représente la majorité. Ce cas est représenté par

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

Ainsi, la probabilité d'avoir un correct négatif est

$$\boldsymbol{CN(t, n, p, r) = (1) + (2)}$$

2.3.2. Faux Négatifs (FN)

Les faux négatifs sont représentés par la probabilité *FN(n, t, p, r)*. Cette dernière désigne le cas où un lecteur défaillant est considéré comme sain. Cela se traduit par les deux cas suivants :

- Le lecteur défaillant traite correctement tous les groupes de tags (*i.e.*, la défaillance ne se manifeste pas). Ce cas est représenté par

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

- Le lecteur défaillant traite i groupes de tags ($1 \leq i \leq t$) incorrectement, mais pour chaque groupe de tags traité, il y a au moins la moitié des lecteurs qui le traite incorrectement (et qui sont donc défaillants) de façon à représenter la majorité à chaque fois. Ce cas est représenté par

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

Ainsi, la probabilité d'avoir un faux négatif est

$$\boldsymbol{FN(t, n, p, r) = (3) + (4)}$$

Remarque : le détail de l'obtention de ces formules est donné dans l'annexe E.

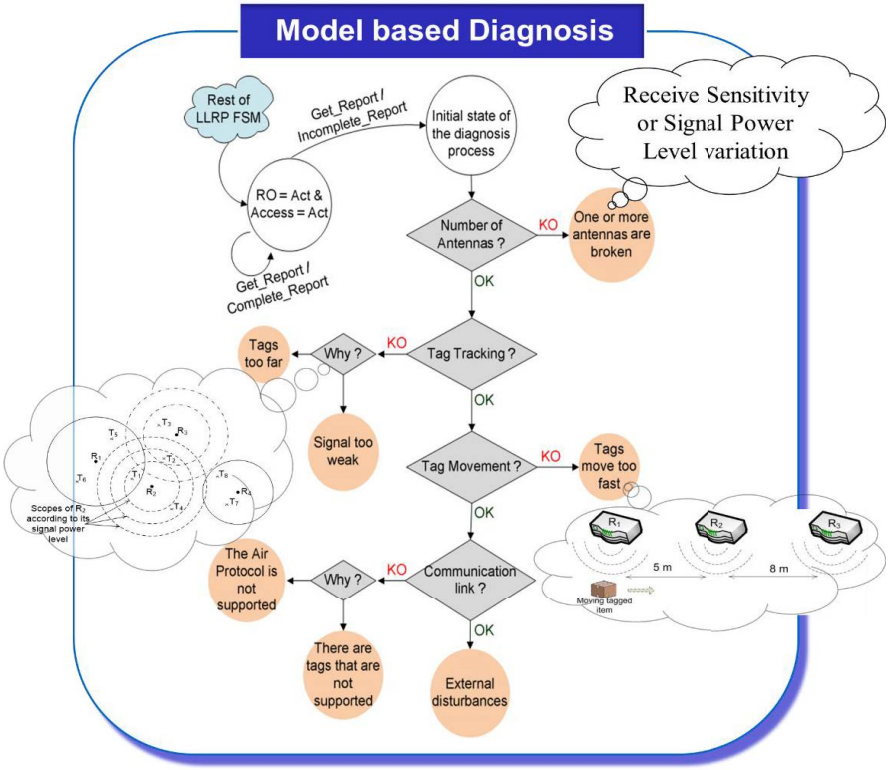
IV. Conclusion

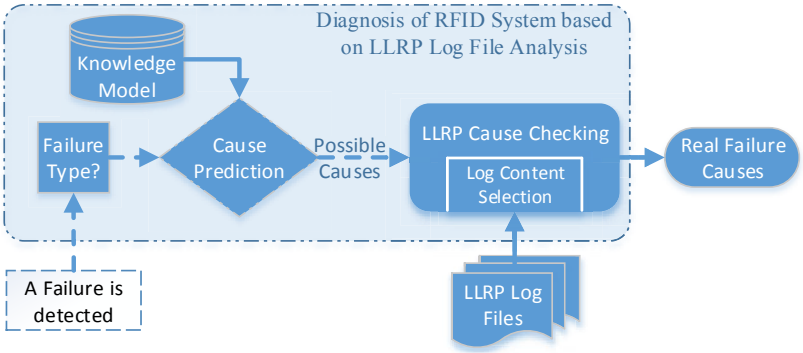
Nous avons vu dans ce chapitre l'architecture du middleware SafeRFID-MW avec quelques diagrammes UML illustrant son fonctionnement et ses cas d'utilisation. Puis nous avons introduit une solution de sûreté de fonctionnement appelée « Algorithme *RFID diagAlgo* », développée dans nos travaux et intégrée au middleware SafeRFID-MW. Cet algorithme consiste en trois étapes. La première est le groupement des lecteurs se situant sur le même flux de données RFID. La deuxième est la comparaison des résultats des lecteurs du même groupe dans le but d'identifier les lecteurs probablement défaillants. La troisième et dernière étape vise à valider la détection de la défaillance en associant à cette détection une probabilité appelée *Identifiabilité* ou fiabilité du diagnostic.

Dans le prochain chapitre, nous présenterons la deuxième partie de notre contribution qui consiste en une extension du protocole LLRP qui donne à ce protocole des capacités de tolérance aux fautes et un mécanisme d'analyse des logs LLRP dans le but de déduire les causes de la défaillance.

Chapitre 4
Contribution au niveau « Interface de Communication »

Sommaire





Mots Clés

Low Level Reader Protocol
LLRP
Distinguishing sequences
Successor tree
Automate à états finis
Arbre de diagnostic
Défaillance LLRP
ROSpec
AccessSpec
Log4j
Log LLRP
Modèle de connaissances
Vérification des causes

CHAPITRE 4
CONTRIBUTION AU NIVEAU « INTERFACE DE COMMUNICATION »

I. Introduction

Le standard de communication entre les lecteurs RFID et le middleware, à savoir le protocole LLRP se base sur des fichiers de configuration XML afin que le middleware ou l'application cliente puisse communiquer et configurer les lecteurs RFID suivant les besoins de l'utilisateur. Ce protocole complexe dispose de certains mécanismes de contrôle d'erreurs comme le mécanisme de contrôle de redondance cyclique CRC[43] (Cyclic Redundancy Check), les messages de notification d'erreurs dans les paramètres d'un message LLRP et les accusés de réception des messages (ACK) [36]. Mais ce protocole n'est pas conçu pour détecter les erreurs qui ne sont pas liées à l'état des données transmises ou pour diagnostiquer les causes des erreurs, alors que cela est vital pour remettre le système RFID en service. Ce protocole est donc inadapté pour une utilisation dans un environnement rude où la sûreté de fonctionnement est requise.

Malgré les avancées technologiques, la RFID, notamment la RFID passive, demeure non fiable ; les lecteurs RFID disposent d'une précision de lecture ne dépassant pas souvent 70% [7] [8] [9], et l'interface lecteur/tag est très sensible aux perturbations externes comme la présence d'ondes radios ou d'obstacles (liquides, objets métalliques, *etc.*). Cela conduit souvent à faire dévier le système de son comportement normal. Dans ce chapitre, nous présentons deux solutions de sûreté de fonctionnement au niveau de l'interface de communication entre le middleware et le lecteur RFID. La première solution consiste en une extension du protocole LLRP pour prendre en charge les défaillances RFID. La deuxième solution consiste en un mécanisme d'analyse des fichiers log LLRP dans le but de déduire les causes de la défaillance.

II. Test et diagnostic des systèmes RFID par extension du modèle LLRP

Dans cette approche, nous étudions comment exploiter les techniques existantes pour faire face à certaines défaillances (voir Section II.1). Pour cela, nous proposons d'exploiter une technique existante sur le modèle LLRP sans l'étendre. Cette technique s'appelle les *Distinguishing Sequences*. Elle permet d'identifier l'état courant du système, et surtout de déduire l'état initial dans lequel se trouvait le système au moment de la défaillance. Ensuite, nous présentons une extension du protocole LLRP [12] afin de prendre en charge les défaillances qui ne sont pas prises en compte par les techniques existantes. Cette extension consiste en un ensemble d'états et transitions que le système emprunte dès qu'une défaillance survient afin de localiser et d'identifier la cause de la défaillance.

1. Défaillances LLRP

L'ensemble des défaillances auxquelles nous nous intéressons sont identifiées grâce à l'application d'une AMDE (Analyse des Modes de Défaillance et de leurs Effets) sur les composants d'un middleware RFID [12]. L'AMDE appliquée aux lecteurs RFID est présentée dans les travaux de G. Fritz [64]. Le résultat complet de cette analyse est donné en annexe C. Les défaillances RFID principales peuvent être classées suivant leurs causes en deux catégories :

[43] Le CRC ou Contrôle de redondance Cyclique est un mécanisme de contrôle d'erreurs dans les données.

1.1. Défaillances dues à une mauvaise conception

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

1.2. Défaillances dues aux conditions d'exécution

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

2. Distinguishing Sequences

La première catégorie de défaillances peut être gérée par notre middleware sans le besoin d'étendre le modèle LLRP existant. Cela est possible grâce notamment aux « Distinguishing Sequences » qui permettent de vérifier l'état interne du lecteur RFID ou du middleware. Une « Distinguishing Sequence » est un ensemble de transitions qui permet l'identification de l'état courant et de l'état initial du middleware ou du lecteur [71] [72].

Cette approche consiste à prouver que le lecteur ou le middleware se trouvait dans un état autre que celui supposé lors de la défaillance ; *e.g.*, il se trouvait dans un état où il ne s'attendait pas à recevoir de données comme le résultat d'un inventaire ou toute autre commande, ce qui l'a amené à les ignorer.

Pour illustrer ce principe, supposons que le comportement d'un lecteur RFID est réduit à l'automate de la figure 53. Cette figure représente un automate à trois états avec deux symboles en entrée {A, B} et deux autres en sortie {0, 1}.

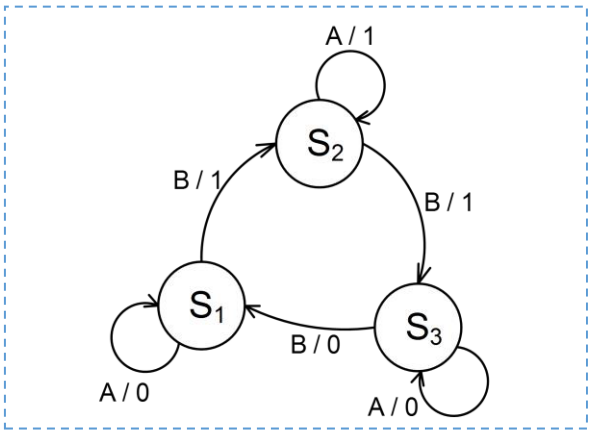


Figure 53. Automate arbitraire à trois états

Lorsqu'une défaillance apparaît dans le système, nous ne savons pas dans quel état se trouve celui-ci. Nous procédons alors, par la construction de l'arbre successeur[44] qui servira comme arbre de décision pour identifier les « distinguishing sequences ».

[44] L'arbre des successeurs ou « Successor tree » montre le comportement de l'automate en partant de tous les états possibles avec toutes les séquences d'entrées possibles [71].

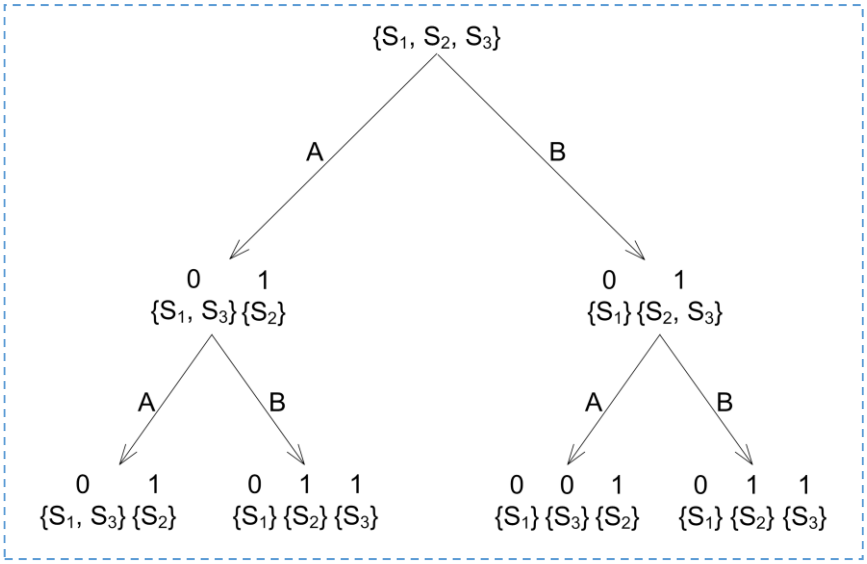


Figure 54. Arbre successeur de l'automate précédent

La figure 54 montre l'arbre successeur de l'automate précédent. Nous pouvons voir que l'entrée A sépare l'état S_2 de S_1 et S_3. Puis l'entrée B sépare S_1 de S_3. Comme exemple, la séquence {A/1} est une séquence distinctive pour S_2 (*i.e.*, après l'application de l'entrée A et l'observation de la sortie 1, nous sommes sûrs que l'état courant ainsi que l'initial du système sont S_2). La séquence {A/0, B/0} est une séquence distinctive de S_3 ; *i.e.*, dans l'automate de la figure 53, lorsque nous appliquons la séquence inverse de la séquence distinctive de S_3 (*i.e.*, {B/0, A/0}), le système sera dans l'état S_3 qui est l'état d'origine.

Une séquence distinctive mène toujours vers un état final unique. Par exemple, la séquence {A/0, A/0} n'est pas une séquence distinctive car elle ne distingue pas S1 de S3, et nous ne pouvons pas remonter pour déduire l'état initial du système. Aussi, l'inverse d'une séquence distinctive doit aussi mener vers un état unique pour que la séquence distinctive d'un état soit valide. L'automate de la figure 55 montre les difficultés rencontrées lors de la construction des séquences distinctives. Toutes les séquences qui commencent par la transition A/0 ne peuvent jamais être des séquences distinctives, vu que leurs séquences inverses sont non-déterministes dans l'arbre de décision ; (*i.e.*, nous ne pouvons jamais déduire si l'automate était dans S_2 ou S_3).

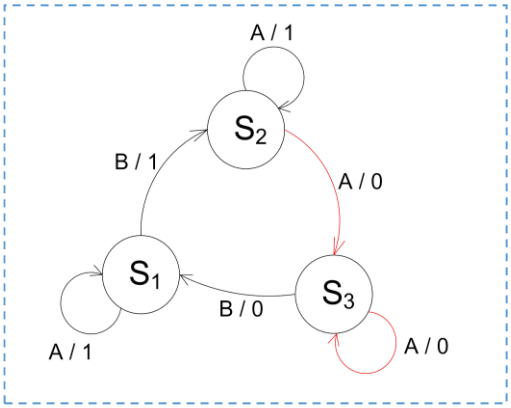


Figure 55. Automate illustrant la difficulté de construire les séquences distinctives

Pour diagnostiquer les défaillances de la première catégorie (cf. Section II.1), nous utilisons comme mentionné précédemment le principe des séquences distinctives présentées ci-dessus. Par exemple, pour détecter la cause de la défaillance « Masking of useful data by the reader » (*i.e.*, le middleware ne reçoit pas toutes les informations demandées sur les tags), nous exécutons des commandes LLRP suivant la machine à états finis de ce protocole, et avec la technique précédente (*i.e.*, « Distinguishing Sequences »), nous montrons que le lecteur RFID se trouvait dans un état erroné. Une cause possible pourrait être par exemple la non-exécution de la AccessSpec pour accéder aux mémoires des tags. A la fin du diagnostic, nous pouvons démontrer que le lecteur était dans un de ces états :

- Etat 1 : la ROSpec est active mais non la AccessSpec.
- Etat 2 : la ROSpec et la AccessSpec sont actives, mais la AccessSpec ne figure pas dans la liste des AccessSpec éligibles pour exécution par la ROSpec.
- Etat 3 : la ROSpec et la AccessSpec sont actives, mais les tags identifiés ne remplissent pas les conditions de déclenchement de la AccessSpec.

3. Extension du protocole LLRP

La seconde catégorie de défaillances nécessite quant à elle d'étendre le modèle existant vu que ces défaillances ne dépendent pas de l'état interne du lecteur ou du middleware mais généralement de causes externes au système. Le principe de l'extension du protocole LLRP est illustré dans la figure 56. Quand la communication entre le middleware et le lecteur se passe sans erreur, le système (lecteur et middleware) transite dans les états prévus par le modèle LLRP. Mais une fois qu'une erreur est détectée (transmission de données erronées, valeur d'un paramètre de performance[45] anormale, *etc.*) le système migre vers un autre état de l'extension que nous proposons afin de pouvoir diagnostiquer les raisons de cette défaillance.

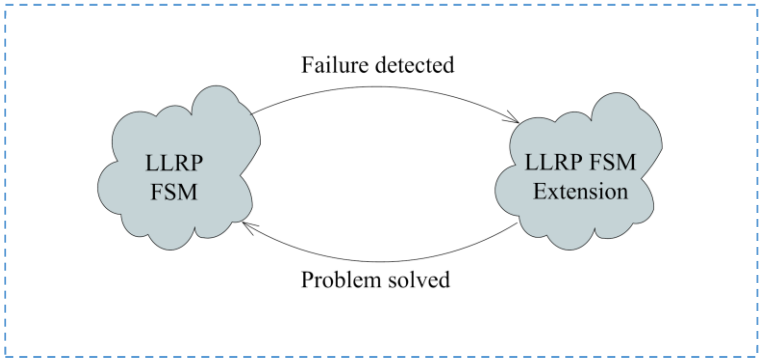


Figure 56. Modèle LLRP étendu

Pour des raisons de simplicité, le modèle LLRP étendu n'est pas directement présenté ici, mais nous présentons plutôt comment l'exploiter pour le diagnostic et le test. Dans ce qui suit, nous donnons des exemples pour montrer comment construire des séquences de test pour identifier les causes des défaillances précédentes (défaillances dues aux conditions d'exécution).

1) La défaillance **« Slow execution, which can lead to Component overload »** peut être due à la quantité de données à traiter (*e.g.*, beaucoup de lecteurs RFID) qui mène à une surcharge du middleware qui les gère. Il faut savoir que le middleware associe à chaque lecteur qui se connecte une instance de l'automate LLRP afin de maintenir avec lui une communication cohérente. En conséquence, pour établir que le problème vient de la quantité de données

[45] Voir Section IV.2.2 du chapitre 2.

qui sont remontées par les lecteurs vers le middleware, ce dernier n'a qu'à compter le nombre de lecteurs qui sont dans un état de communication (grâce aux instances de l'automate LLRP associées à chaque lecteur) et vérifier que ce nombre ne dépasse pas la limite du middleware. Si tel est le cas, le middleware pourra demander à certains lecteurs disposant de mémoires internes de mettre en pause la communication et de stocker temporairement leurs données.

2) Les défaillances **« No data capture »** et **« Erroneous received data »** sont les plus susceptibles d'apparaître. Ces défaillances peuvent être causées par les mêmes causes que celles de la première catégorie, mais aussi par les causes suivantes :

- L'antenne du lecteur est cassée ou endommagée.
- La présence de perturbations externes (ondes radio d'autres appareils, présence d'obstacles comme l'eau et le métal, etc.).
- Les tags sont hors de la portée du lecteur.
- Les tags bougent trop vite.
- Le type de tags est non supporté par le lecteur, ou c'est le protocole air (air protocole comme UHF C1G2) qui n'est pas supporté par les tags.

L'idée est d'inclure ces causes de défaillances dans le processus de diagnostic sous forme d'un modèle LLRP étendu. La figure 57 montre cette extension sous forme d'un organigramme, où les losanges représentent les entrées du système et les feuilles (les cercles) représentent les causes possibles de la défaillance. Comme nous pouvons le voir sur la figure, quand le lecteur exécute une ROSpec avec une AcessSpec (*i.e.*, la ROspec et la AccessSpec sont actives), le middleware peut demander au lecteur le résultat de ses spécifications. Si le rapport est incomplet (*i.e.*, il y a des tags qui ne sont pas correctement inventoriés), le middleware lance le processus de diagnostic.

Pour identifier la cause d'une défaillance, le middleware procède comme suit :

i. Le middleware peut vérifier si une antenne d'un lecteur est endommagée ou cassée en demandant au lecteur de lui fournir le nombre d'antennes opérationnelles, ou bien en définissant une RFSurvey (si elle est supportée par le lecteur) sur toutes les antennes. Une RFSurvey est utile pour deux raisons. La première est qu'elle permet au middleware de connaître le nombre d'antennes opérationnelles, et de le comparer avec le nombre d'origine d'antennes. La deuxième est que le middleware peut connaître les niveaux de puissance du signal supportés par chaque antenne. Cela est utile pour vérifier si le lecteur n'a pas utilisé un signal dont la puissance dépasse la capacité de l'antenne. Si le lecteur n'est pas capable d'effectuer les RFSurveys et sachant que la plupart des lecteurs disposent généralement d'au moins deux antennes (*e.g.*, une pour la transmission et une autre pour la réception), le lecteur peut transmettre un signal par une antenne et le recevoir par l'autre. Si cette opération est un succès, alors le middleware suppose que les antennes ne sont pas cassées. Maintenant, pour vérifier si une des antennes est endommagée, le middleware peut faire varier la puissance du signal (signal power level) ou la

sensibilité des antennes (receive sensitivity[46]) pour calculer la portée de l'antenne et déduire si celle-ci est endommagée ou pas (une antenne endommagée couvre une zone moins importante qu'une antenne en bon état). S'il existe plus de deux antennes pour un lecteur, le middleware les testera deux à deux.

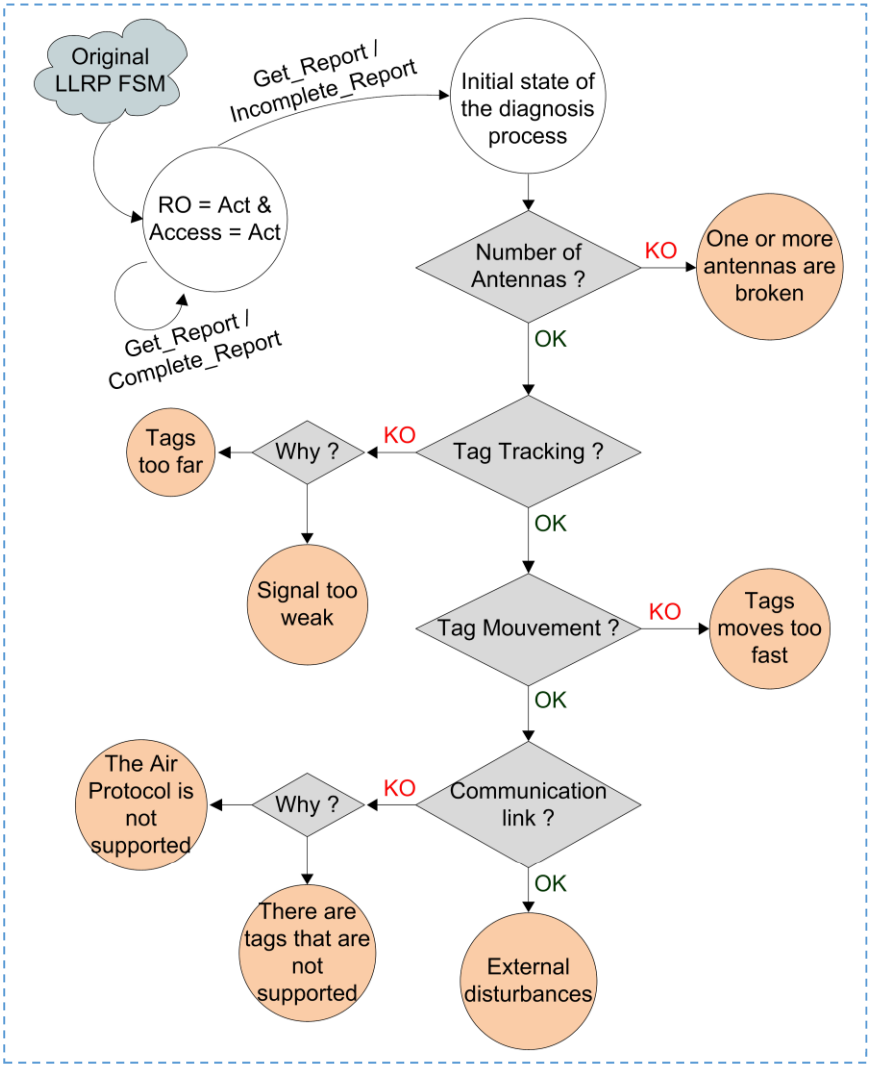


Figure 57. Arbre de diagnostic LLRP

ii. Pour identifier la seconde cause *i.e.*, « **the tags are out of the scope of the antenna** », le middleware localise les tags en variant la puissance du signal ou la sensibilité des antennes comme expliqué précédemment. La figure 58 montre une zone peuplée par 4 lecteurs (R_1, R_2, R_3, R_4) et 8 tags (T_1 à T_8). Cette figure met l'accent sur la localisation des tags par la variation de la portée des antennes des lecteurs.

[46] Receive Sensitivity indique la puissance minimale d'un signal radiofréquence qui garantit la réception avec succès.

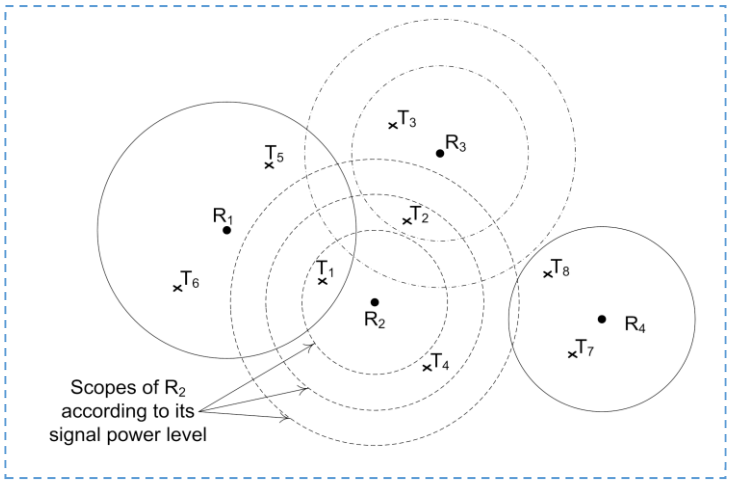


Figure 58. Technique de localisation des tags

Par exemple, pour vérifier que le tag T_2 est trop loin du lecteur R_1, le middleware va utiliser les voisins de R_1 pour localiser le tag T_2 en réduisant la portée de chaque lecteur suffisamment pour localiser précisément le tag.

iii. Pour déduire que la défaillance vient du mouvement trop rapide des tags, le middleware procède comme suit :

- Si les tags sont transportés sur un tapis roulant comme illustré dans la figure 59, le middleware n'a qu'à connaître la vitesse du tapis roulant et voir si cette vitesse ne dépasse pas celle supportée par le lecteur.

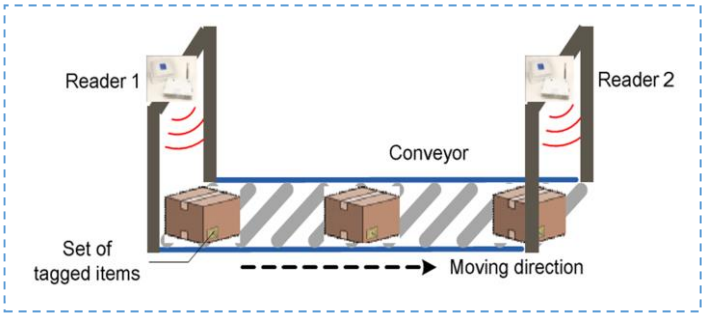


Figure 59. Transport de produits étiquetés sur un tapis roulant

- La deuxième possibilité est de calculer la vitesse de déplacement des tags (ou de son accélération si besoin) en utilisant les lecteurs qui les ont déjà identifiés. Par exemple, si le lecteur R_1 de la figure 60 identifie un tag à 11:14:25, et le lecteur R_2 l'identifie à $11{:}14{:}35$, le middleware peut déduire sa vitesse de déplacement (qui est de $0{,}5\ \mathrm{m/s}$). Ainsi, il peut même prévoir quand le tag sera visible par le lecteur R_3 et avec quelle vitesse (il peut aussi calculer l'accélération ou la vitesse moyenne de déplacement du tag au cas où la vitesse de ce tag n'est pas constante).

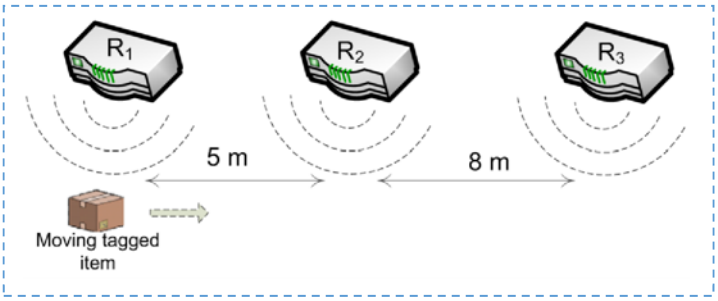


Figure 60. Calcul de la vitesse de déplacement d'un tag

iv. Pour déclarer que la défaillance est due au lien de communication, le middleware procède comme suit :

- Si des données erronées sont reçues par le middleware, alors nous pouvons être sûrs que le problème vient du lien de communication entre le lecteur et le middleware, car sinon les données erronées seront filtrées par le lecteur (*e.g.*, cela peut conduire à un inventaire incomplet).
- Pour vérifier que le problème vient du lien de communication entre le lecteur et les tags, le middleware peut demander aux voisins du lecteur en question, quel protocole ils ont utilisé pour identifier correctement ces tags et ainsi, déduire si le lecteur en question a utilisé un protocole non supporté par ces tags. Autrement dit, le middleware vérifie si le lecteur supporte ou non ce type de tags (*e.g.*, un lecteur UHF qui essaie de lire des tags HF).
- Si aucune de ces causes n'est vérifiée, le middleware supposera que la défaillance est causée par des perturbations externes (comme la présence d'objets métalliques, d'ondes radio parasites, de liquide, *etc*.) entre le lecteur et les tags.

Notons aussi que les défaillances telles que « inventaire ou accès aux mémoires des tags impossible » peuvent venir du fait que le nombre maximal des ROSpec ou des AccessSpec a été atteint. Dans ce cas, le middleware n'a qu'à supprimer les anciennes spécifications pour ajouter de nouvelles. Nous avons intentionnellement omis ce genre de défaillances car nous supposons que le middleware vérifie le nombre de spécifications qu'un lecteur peut gérer. Cette vérification peut être faite par la commande LLRP « GET_READER_CAPABILITIES ».

III. Analyse des logs LLRP

Le protocole LLRP se base sur des spécifications XML pour configurer et fixer un nombre important de paramètres d'un lecteur RFID et assurer un inventaire et un accès correct aux mémoires des tags [36]. Cette configuration est faite manuellement par l'utilisateur et sans contrôle ou validation par le protocole, ce qui peut être source d'erreurs. Comme ce protocole est relativement nouveau, il n'y a pas de travaux significatifs qui abordent les défaillances que peut engendrer une mauvaise utilisation de ce protocole.

L'approche que nous proposons se base sur l'analyse des événements LLRP stockés dans les fichiers log pour le monitoring des systèmes RFID [13]. Cette approche peut être utilisée en complément avec l'algorithme de diagnostic probabiliste RFID diagAlgo présenté au chapitre 3 ou avec toute autre technique de détection de défaillances. Cette approche permet, une fois que la défaillance a été détectée (notamment grâce à l'algorithme RFID diagAlgo) de trouver le ou les paramètres LLRP qui sont mal définis par l'utilisateur.

L'implémentation actuelle du protocole LLRP [73] utilise le système de génération des logs Log4j [74]. Ce système permet à l'utilisateur d'initialiser et de fixer le format et le contenu des fichiers log de manière simple sans toucher à l'implémentation du protocole LLRP. Cette approche repose sur l'analyse des logs hors ligne[47] (*off-line diagnosis*) pour permette de construire la suite d'événements qui ont conduit à la défaillance du système RFID. Cette approche est similaire au mécanisme des "*Distinguishing Sequences*" qui vise à identifier l'état du système au moment de la défaillance grâce à l'exécution d'une suite « bien spécifique » de commandes sur le système fautif. Cette suite de commandes correspond à la suite de transitions dans la machine à états finis du système RFID (automate LLRP).

Les *distinguishing sequences*

- présentent l'inconvénient d'être un mécanisme intrusif vu qu'elles nécessitent d'injecter des données supplémentaires dans le système.
- n'existent pas toujours (par définition) et encore moins pour tous les états du système ; ce qui implique une couverture du nombre de défaillances réduite.
- les appliquer sur un système défaillant
    - n'est pas toujours possible (exemple d'un système bloqué ou hors service).
    - n'est pas souhaité car il mène le système vers des états indésirables qui n'ont rien à voir avec le fonctionnement normal du système ; ce qui nécessite souvent l'arrêt du service fourni par le système RFID pour effectuer correctement le diagnostic.

Toutes ces raisons motivent notre approche d'analyse des fichiers log LLRP. Ce mécanisme d'analyse non intrusif permettra d'avoir un maximum d'informations sur les circonstances de la défaillance et mieux aider le processus de diagnostic.

Le mécanisme d'analyse de logs [13] a été intégré dans le middleware SafeRFID-MW et il est déclenché à chaque fois qu'une défaillance est détectée. La figure 61 résume le fonctionnement du processus de vérification des causes. Une fois la défaillance reconnue, un processus de prédiction des causes est lancé afin de sélectionner un ensemble de causes probables de cette défaillance qu'il faut vérifier. Une fois cette étape passée, un autre processus est déclenché pour interpréter les fichiers logs à la recherche de tout comportement suspect (trace d'exécution) suivant la liste des causes probables déjà établies. Ceci permet de remonter et de déduire la ou les causes de la défaillance.

Figure 61. Processus de vérification des causes des défaillances

1. Modèle de faute

LLRP est un protocole complexe avec un grand nombre de configurations possibles. Cette caractéristique est certainement très utile car elle rend le protocole extrêmement adaptable aux besoins de l'utilisateur. Cependant, cette même caractéristique représente le principal défaut du protocole. En effet, l'utilisateur peut facilement se tromper dans les valeurs données à certains paramètres, et ainsi

[47] L'analyse des événements LLRP peut aussi être appliquée en ligne (*online diagnosis*). Il suffit pour cela qu'au lieu d'utiliser le log vers des fichiers, le système transmet directement les informations à notre solution d'analyse qui construira la suite d'événements au fur et à mesure du fonctionnement du système. Ainsi, dès qu'une défaillance est détectée, le mécanisme d'analyse se met à analyser les séquences d'événements à la recherche de toute déviation du comportement du système sous surveillance.

avoir un comportement indésirable des lecteurs ; *e.g.*, avoir un inventaire des tags incomplet, des écritures impossibles sur des tags, *etc*.

Nous proposons dans le cadre de cette approche d'étudier les défaillances issues des erreurs de configuration et de manipulation du protocole LLRP. Nous classons ces défaillances en trois catégories :

- **Pas de tags identifiés** : le lecteur n'arrive à identifier aucun tag dans son champ de lecture.
- **Inventaire incomplet (tags manquants)** : la liste des tags identifiés par le lecteur est incomplète.
- **Problème d'accès aux mémoires des tags** : le lecteur n'a pas pu effectuer les opérations demandées sur les mémoires des tags (lecture, écriture, verrouillage, *etc.*)

2. Diagnostic des systèmes RFID à base d'analyse des fichiers log

Le processus d'analyse des fichiers log du protocole LLRP peut être résumé en 4 étapes comme le montre la figure 62 : (i) Quand une défaillance est détectée par un mécanisme de monitoring donné, comme celui que nous avons proposé au chapitre 3 (Algorithme RFID diagAlgo), celle-ci est classée suivant le modèle de fautes. (ii) Un processus de prédiction de causes est lancé pour prédire l'ensemble des causes de la défaillance et cela en se basant sur le type de la défaillance détectée et un ensemble de règles qu'il récupère d'un modèle de connaissances que nous verrons par la suite. (iii) Un sous-ensemble d'informations basé sur les causes à vérifier est extrait des fichiers log. Cette étape est très importante pour l'efficacité de ce mécanisme de diagnostic car elle permet de se concentrer uniquement sur l'analyse des données qui concerne la défaillance. Cela est d'autant plus important que le volume journalier de données stockées dans les fichiers log est de l'ordre du gigaoctet par lecteur. (iv) Un processus de vérification des causes de défaillance utilisant les informations extraites dans l'étape précédente vérifie chaque cause issue de l'étape 2 et ainsi identifie les causes réelles de la défaillance.

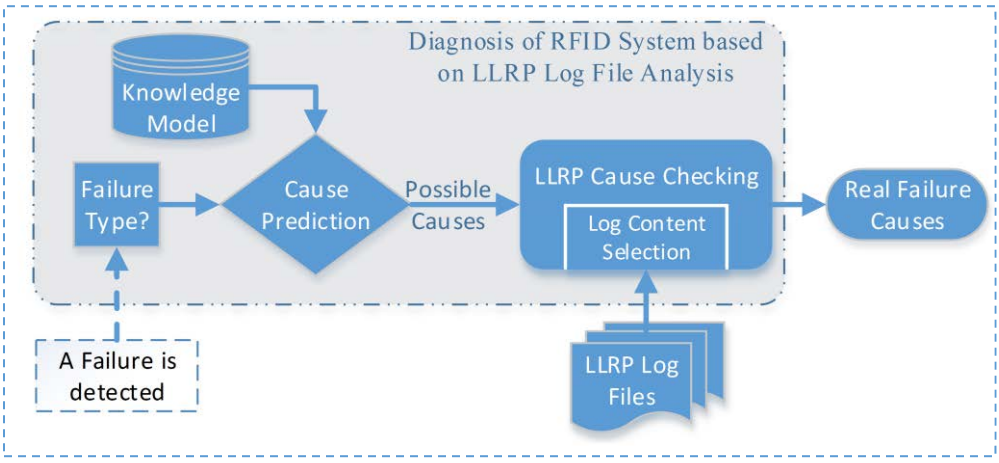


Figure 62. Diagnostic des systèmes RFID à base d'analyse de logs LLRP

2.1. Modèle de connaissances

Le modèle de connaissances est un ensemble de règles de diagnostic qui consiste à associer pour chaque défaillance, un ensemble de causes possibles. Ces règles sont définies à l'aide d'une AMDE (Analyse des Modes de Défaillances et de leurs Effets) appliquée sur un système RFID compatible LLRP. Le modèle de connaissances dispose aussi suivant chaque cause possible d'un pattern ou d'une signature qui sera utilisée lors de l'analyse du fichier log. Ces règles de diagnostic sont résumées dans la table 6.
L'implémentation actuelle du protocole LLRP [73] utilise le système de log de la fondation Apache « Log4j » [74]. Log4j dispose de plusieurs formats de log. Il permet à l'utilisateur de sélectionner les différentes informations qu'il veut stocker. La figure 63 donne un exemple de fragment d'un fichier log LLRP par défaut qui montre un envoi réussi d'un message START_ROSPEC.

```
5936 [AnonymousIoService-6] INFO org.llrp.ltk.net.LLRPIoHandlerAdapterImpl  - Message START_ROSPEC successfully transmitted
5940 [AnonymousIoService-6] DEBUG org.llrp.ltk.net.LLRPIoHandlerAdapterImpl  - <?xml version="1.0" encoding="UTF-8"?>
<llrp:START_ROSPEC xmlns:llrp="http://www.llrp.org/ltk/schema/core/encoding/xml/1.0" Version="1" MessageID="0">
        <llrp:ROSpecID>1</llrp:ROSpecID>
</llrp:START_ROSPEC>
```

Figure 63. Format du log par défaut de LLRP

Dans le cadre de notre approche, nous proposons de reconfigurer le système de log de LLRP afin que les données en sortie soient au format XML en vue de faciliter leur traitement. La figure 64 montre un bout de code en java qui permet d'avoir un fichier log XML dont le nom est « LLRP_event_log.xml ».

```
try {
        FileAppender fa1 = new FileAppender(
            new XMLLayout(),"LLRP_event_log.xml");
        fa1.setName("LLRP_event_log");
        BasicConfigurator.configure(fa1);
}catch (IOException ex){
        ex.getMessage();
}
```

Figure 64. Bout de code Java pour la génération des logs XML

Les fichiers log XML sont bien adaptés pour les traitements automatiques. Par contre, ils le sont moins pour les traitements manuels par un administrateur par exemple. Pour répondre à cette limitation, un nouveau fichier log au format plus lisible sera créé en parallèle du fichier log XML et sera archivé toute les 24 heures sous le nom « nom initial du fichier + date d'archivage ». La figure 65 montre le code java qui permet de mettre en forme et de sélectionner le contenu du fichier log. La figure 66 donne un exemple de fichier log résultant du code de la figure 65 et qui se résume à la date de l'événement, la source de l'événement et la description de l'événement ou éventuellement le contenu du message.

```
try {
        DailyRollingFileAppender  fa2 =
            new DailyRollingFileAppender (new PatternLayout(
            "Date: %d{dd/MM/yy HH:mm:ss.SSS}%n"
            +"Source: %t: %c%n"
            +"Msg [%p]: %m %n%n"), "LLRP_events.log","'.'yyyy-MM-dd");
        BasicConfigurator.configure(fa2);
} catch (IOException ex) {
        ex.getMessage();
}
```

Figure 65. Génération de fichiers log avec sélection du contenu et mise en forme

```
Date: 13/04/14 15:13:25.561
Source: AnonymousIoService-8: org.llrp.ltk.net.LLRPIoHandlerAdapterImpl
Msg [INFO]: Message START_ROSPEC successfully transmitted

Date: 13/04/14 15:13:25.561
Source: AnonymousIoService-8: org.llrp.ltk.net.LLRPIoHandlerAdapterImpl
Msg [DEBUG]:
<?xml version="1.0" encoding="UTF-8"?>
<llrp:START_ROSPEC xmlns:llrp="http://www.llrp.org/.../1.0" Version="1" MessageID="0">
        <llrp:ROSpecID>1</llrp:ROSpecID>
</llrp:START_ROSPEC>
```

Figure 66. Format du fichier log final

2.1.1. Lien causes à effet

La table 6 montre la relation qui existe entre les défaillances et leurs causes possibles. Nous rappelons que les défaillances auxquelles nous nous intéressons sont :

- Pas de tags identifiés.
- Inventaire incomplet (tags manquants).
- Problème dans l'accès aux mémoires des tags.

Dans cette approche, nous ne considérons que les causes de défaillances qui sont dues à une mauvaise utilisation du protocole LLRP. Ainsi, dans le cas de la défaillance « Pas de tags identifiés », nous ne considérons pas les causes « Lecteur non allumé » ou « Connexion LLRP perdue », *etc.* qui peuvent être facilement vérifiées avec par exemple une commande ping.

Table 6. Défaillances RFID et leurs causes LLRP

Défaillance	Causes possibles	Eléments à vérifier
Aucun tag identifié	Ajout de la ROSpec impossible	L'état de la ROSpec est différent de « disabled »
		L'identifiant de la ROSpec existe déjà au niveau du lecteur.
		Le nombre maximal de ROSpecs ou de AISpecs a été atteint pour le lecteur RFID (ce nombre est accessible par la commande LLRP GET_READER_CAPABILITIES)
		La valeur de la priorité de la ROSpec n'est pas valide ; comparer la valeur spécifiée avec celles gérées par le lecteur (qui peuvent être obtenues avec GET_READER_CAPABILITIES)
		L'identifiant spécifié de l'antenne n'est pas valide
	Aucune ROSpec dans le lecteur	Aucun ajout de ROSpec par l'utilisateur ; vérifier s'il n'y a pas de message ADD_ROSPEC dans le fichier log LLRP
		Suppression de la ROSpec avant l'identification des tags ; vérifier l'existence du message DELETE_ROSPEC dans le fichier log LLRP
	ROspec non activée	Pas de message START de l'utilisateur
		Les conditions spécifiées dans « START CONDITIONS » ne sont pas satisfaites
		La valeur d'un des sous-paramètres de AISpecStopTrigger (DurationTrigger, NumberOfTags, T parameter) n'est pas adéquate
	Problème de Receive Sensitivity	Vérifier les valeurs spécifiées pour le paramètre « Receive Sensitivity » grâce à la commande LLRP GET_READER_CONFIG
	Inventaire non envoyé par le lecteur	Vérifier le paramètre de déclenchement de l'envoi du rapport RORepportTrigger
	Préemption de la ROSpec	Vérifier si la ROSpec a été interrompue par une autre plus prioritaire
Inventaire des tags incomplet	Résultats d'inventaire obtenus mais non envoyés au client	Vérifier la valeur du paramètre N de RORepportTrigger ; N est soit très grand ou très petit (mais différent de 1)
		Préemption : la ROSpec a peut-être été interrompue par une autre plus prioritaire avant la fin de son inventaire.

Défaillance	Causes possibles	Eléments à vérifier
	Résultats complet non encore obtenus	ROSpecStopTrigger est fixé à une valeur inadéquate. De ce fait, l'inventaire se termine avant l'identification de tous les tags
		Vérifier les valeurs spécifiées pour le paramètre « Receive Sensitivity » grâce à la commande LLRP GET_READER_CONFIG
		La valeur de TagPopulation est soit très inférieure ou très supérieure au nombre réel de tags ce qui réduit la vitesse de lecture des tags par le lecteur RFID
		La valeur d'un des sous-paramètres d'AISpecStopTrigger (DurationTrigger, NumberOfTags, T parameter) n'est pas adéquate
Problème d'accès aux mémoires des tags	Pas de AccessSpec	Vérifier dans le fichier log s'il n'y a pas eu de tentative d'ajout d'AccessSpec grâce au message ADD_ACCESSSPEC_RESPONSE
	AccessSpec Incohérante	Vérifier si l'AccessSpec est active
		Vérifier si l'AccessSpec est reliée à la ROSpec en cours d'exécution
		Vérifier si les tags identifiés par la ROSpec passent le filtre TagSpec de l'AccessSpec
		Vérifier si l'antenne ID spécifiée dans l'AccessSpec est la même que celle de ROSpec
		Vérifier si le contenu sélectionné dans l'AccessRepportTrigger correspond à ce qui est attendu
	Préemption	La ROspec responsable de déclencher l'AccessSpec a été interrompue (l'AccessSpec ne s'est pas exécutée)

2.1.2. Explication du modèle de connaissance

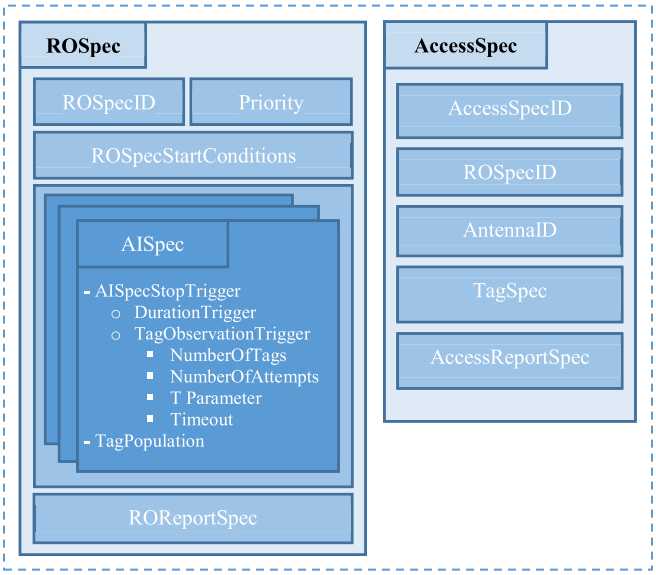


Figure 67. Contenu principal de ROSpec et AccessSpec

La figure 67 montre les principaux paramètres d'une ROSpec et d'une AccessSpec qui sont utilisés par le modèle de connaissances.

- `ROSpecStartCondition` peut être une requête START de l'utilisateur, une durée périodique (*e.g.*, exécuter la `ROSpec` chaque jour à 8:00 *am*) ou un événement externe (*e.g.*, un signal d'un détecteur de mouvement).
- `AISpec` représente le composant principal d'une ROSpec. Elle permet de définir tous les paramètres nécessaires pour effectuer un inventaire des tags par le lecteur.
- `DurationTrigger` spécifie la durée d'exécution d'une `AISpec`. Une fois que cette durée est écoulée, la `AISpec` en cours d'exécution sera arrêtée, et la `AISpec` suivante prend le relais jusqu'à ce qu'il n'y ait plus de `AISpec` à exécuter, ce qui marque la fin de l'inventaire des tags.
- `TagObservationTrigger` peut être utilisé au lieu de `DurationTrigger`. Il spécifie la condition d'arrêt de la `AISpec` suivant l'observation ou non des tags RFID. Ce paramètre peut être :
    - `NumberOfTags` : ce paramètre est utilisé pour spécifier au lecteur le nombre de tags à identifier. Une fois ce nombre atteint, la `AISpec` est arrêtée. Cela veut dire que le lecteur ne va identifier que le nombre spécifié de tags, même si d'autres tags non lus sont présents dans son champ de lecture.
    - `NumberOfAttempts` : ce paramètre représente le nombre maximal de tentatives de lecture que le lecteur peut utiliser pour l'identification des tags.
    - Paramètre `T` : ce paramètre, spécifié en millisecondes représente le temps maximal qu'il peut y avoir entre deux réponses de deux tags (temps d'inactivité). Cela signifie que si aucun tag ne répond au bout de `T` millisecondes, la `AISpec` sera arrêtée.
    - `Timeout` : ce paramètre est toujours spécifié si un des trois autres paramètres de `TagObservationTrigger` est utilisé. Il permet de terminer la `AISpec` (qui signifie généralement la fin de l'inventaire des tags) une fois que le temps spécifié est écoulé et cela indépendamment de l'utilisation du second paramètre de `TagObservation-Trigger`.
- `TagPopulation` donne au lecteur une estimation du nombre de tags présents dans son champ de lecture. Le lecteur utilise ce paramètre pour ajuster son algorithme de gestion des collisions (le paramètre Q du protocole EPC C1 Gen2). En règle générale, si la valeur de `TagPopulation` est trop petite par rapport à la réalité, il y aura beaucoup de collisions et par conséquence, l'inventaire de tags prendra beaucoup plus de temps. Si `TagPopulation` est trop grand par rapport à la réalité, il y aura trop peu de collision voire aucune, mais l'inventaire des tags prendra aussi beaucoup plus de temps. Cela s'explique par le fonctionnement de l'algorithme anticollision qui réserve un nombre de slots de temps suivant le nombre de tags attendus. Ces slots sont indexés et chaque tag choisira de manière aléatoire un slot pour communiquer son identifiant au lecteur. Les tags qui ont choisi un slot dont l'index est petit communiqueront avec le lecteur avant ceux qui ont choisi des slots de temps dont l'index est plus grand. Il est clair que si un des tags choisit un slot dont l'index est grand (alors qu'il y a des slots d'index plus petit), le lecteur prendra plus de temps pour l'identifier.
- `ROReportSpec` définit à travers `ROSpecReportTrigger` le moment où le lecteur doit envoyer le rapport (résultats de l'inventaire des tags) au client. L'envoi du rapport peut être déclenché grâce au nombre de tags identifiés ou à une durée de temps.
- `ROSpecID` dans une `AccessSpec` définit quelle `ROSpec` doit lancer cette `AccessSpec`.
- `AntennaID` dans une `ROSpec` ou dans une `AccessSpec` définit quelle antenne du lecteur utiliser lors de l'inventaire ou de l'accès aux mémoires des tags.

- `TagSpec` définit le ou les filtres que les tags doivent passer pour être élus ; (*e.g.*, ne sélectionner que les tags dont l'EPC commence par « aabc »). La `AccessSpec` qui a défini le paramètre `TagSpec` en question sera exécutée sur tous les tags sélectionnés.
- `AccessReportSpec` définit le format du rapport de la `AccessSpec` (résultats de l'accès aux mémoires des tags) et le moment où le lecteur doit l'envoyer au client.

2.2. Vérification des causes des défaillances LLRP

Le processus de vérification des causes se base sur un analyseur de logs LLRP. Il permet de charger et de n'extraire des fichiers log que les informations utiles au diagnostic de la défaillance identifiée suivant le modèle de défaillance. C'est une tâche très importante dès lors que les fichiers log peuvent atteindre une taille très importante (plusieurs giga-octets de données) et que leur traitement devient impossible ou nécessite un temps considérable. De ce fait, ne se concentrer que sur une partie du fichier log permet à notre mécanisme de vérification des causes d'être plus réactif et plus efficace.

Dans la suite de ce paragraphe nous donnons quelques exemples sur le traitement des fichiers log suivant la défaillance identifiée.

2.2.1. Pas de tags identifiés

Pour diagnostiquer ce genre de défaillances, l'analyseur de logs LLRP vérifie si la connexion LLRP est correctement établie entre le client et le lecteur RFID. La figure 68 montre un exemple de signature à chercher dans les fichiers log. Cette signature nous indique l'heure de la tentative de connexion et son état (succès ou échec).

```
<?xml version="1.0" encoding="UTF-8"?>
<llrp:READER_EVENT_NOTIFICATION
xmlns:llrp="http://www.llrp.org/ltk/schema/core/encoding/xml/1.0" Version="1"
MessageID="0">
  <llrp:ReaderEventNotificationData>
    <llrp:UTCTimestamp>
      <llrp:Microseconds>2013-05-18T12:46:53.659000+02:00</llrp:Microseconds>
    </llrp:UTCTimestamp>
    <llrp:ConnectionAttemptEvent>
      <llrp:Status>Success</llrp:Status>
    </llrp:ConnectionAttemptEvent>
  </llrp:ReaderEventNotificationData>
</llrp:READER_EVENT_NOTIFICATION>
```

Figure 68. Réponse d'un lecteur suite à une tentative de connexion du middleware

Après vérification de la connexion, l'analyseur de logs vérifie si la `ROSpec` destinée à lire les tags est correctement ajoutée et mise à l'état inactif avant que les conditions d'exécution ne soient vérifiées (*e.g*, une commande START de l'utilisateur, un signal d'un détecteur de mouvement, un déclencheur périodique) ou simplement si les conditions d'exécution sont correctement spécifiées pour déclencher l'exécution de la `ROSPec`. Si ces vérifications ne révèlent pas d'erreurs de configuration, alors le niveau du signal (Receive Sensitivity[48]) sera vérifié afin de voir s'il n'est pas fixé à une valeur trop faible entraînant une réduction de la portée des antennes du lecteur au moment du passage des tags.

[48] Receive Sensitivity ou la sensibilité de réception des antennes indique le niveau minimal pour qu'un signal radiofréquence soit reçu avec succès par un récepteur donné. Plus le niveau du signal que le récepteur peut interpréter est faible, plus la sensibilité de réception est grande [80].

2.2.2. Inventaire incomplet (tags manquants)

Cette défaillance est due à deux raisons différentes. Soit le lecteur a lu correctement les tags mais, il n'a pas envoyé le résultat au client, soit le lecteur n'a pas lu tous les tags. Dans les deux cas, l'analyseur de logs LLRP vérifie si les déclencheurs de l'envoi de l'inventaire sont correctement fixés. A titre d'exemple, le paramètre RORepportTrigger peut être fixé à un certain nombre N de tags à avoir dans le rapport. Si N est supérieur au nombre de tags dans le champ du lecteur, alors le lecteur n'enverra pas le rapport tant qu'il n'a pas identifié les N tags ou tant que la ROSpec n'est pas terminée. Si N est inférieur au nombre de tags dans le champ du lecteur, alors le lecteur enverra le rapport par partie de N tags et si la dernière partie n'atteint pas les N tags, nous retombons dans le premier cas. Il est clair que si la condition d'arrêt de la ROSpec ROSpecStopTrigger est fixée à « null » (exécution de la ROSpec indéfiniment), ces valeurs de N posent problème. Généralement, N est fixé à 1, pour éviter ce problème et avoir des lectures de tags en temps réel du côté du client.

```
<llrp:TagObservationTrigger>
    <llrp:TriggerType>Upon_Seeing_N_Tags_Or_Timeout</llrp:TriggerType>
    <llrp:NumberOfTags>5</llrp:NumberOfTags>
    <llrp:NumberOfAttempts>10</llrp:NumberOfAttempts>
    <llrp:T>1</llrp:T>
    <llrp:Timeout>10</llrp:Timeout>
</llrp:TagObservationTrigger>
```

Figure 69. Exemple de déclencheurs d'envoi du rapport de lecture

D'autres mauvaises configurations qui empêchent l'identification de tous les tags sont les valeurs fixées dans le paramètre TagObservationTrigger qui sont le nombre de tags attendus ou le nombre de tentatives nécessaire pour identifier tous les tags comme le montre le bout de code XML de la figure 69. Par exemple, si le nombre de tags attendus est inférieur au nombre réel de tags présents dans le champ du lecteur, alors ce dernier n'identifiera qu'une partie de ces tags.

2.2.3. Problème dans l'accès aux mémoires des tags

Dans ce genre de défaillances, l'analyseur de logs LLRP vérifie

- si la AccessSpec correspondante est active et qu'elle est reliée à la ROSpec en cours d'exécution.
- si les tags identifiés passent le filtre défini dans TagSpec de la AccessSpec.
- si l'antenne spécifiée dans la AccessSpec est bien la même que celle de la ROSpec.
- si le paramètre AccessReportSpec est correctement fixé pour déclencher l'envoi du rapport au bon moment.

Exemple : le lecteur surveillé dispose de deux antennes. La ROSpec en cours d'exécution est sur l'antenne #0 et la AccessSpec est prête à s'exécuter sur l'antenne #1. Alors les opérations définies dans la AccessSpec sont susceptibles d'échouer car les tags sont identifiés sur l'antenne #0 et que la portée de l'antenne #1 ne les atteint pas.

3. Implémentation du processus de vérification des causes

Le processus de vérification des causes peut fonctionner avec ou sans aide de l'utilisateur. Cette dernière n'est nécessaire que dans un but d'optimisation du processus de vérification. L'utilisateur peut indiquer au programme ce qu'il a constaté et à quelle date. Cela revient à sélectionner l'une des trois défaillances du modèle de faute précédent. Le processus de vérification des causes est organisé de manière hiérarchique. Il est logique de vérifier tout d'abord si le lecteur a identifié des tags avant de vérifier s'il a manqué d'identifier certains tags, puis, de vérifier si il y a eu un problème lors de l'accès aux mémoires des tags (*i.e.*, l'accès aux mémoires des tags ne peut se faire que si l'identification des tags par le lecteur a réussi). La date saisie par l'utilisateur est utile pour sélectionner le fichier log approprié sachant que le système de génération des logs crée un fichier log toutes les 24 heures ; ce fichier log

correspond aux événements LLRP du système RFID survenus entre 0h00 et 23h59. La figure 70 résume les grandes lignes de l'implémentation du processus de vérification de causes. La fonction checkFailureCauses("type de la défaillance", fichier_log_LLRP) permet de vérifier et d'identifier les causes de la défaillance. Le détail de cette vérification est donné en section III.2.2.

```
userObservation := getUserObservation();

userDateInput := getUserDateInput();

if(userDateInput != null){ //The user has entered a valid date
    logToAnalyse := "get the log file corresponding to the indicated date";
}
else{ //The user does not know when the failure occurred
    logToAnalyse := "get all log files"; //The analysis process will be done on each file
}

switch(userObservation){
    case 0://This case corresponds to "the user did not indicate any observation"
        //there is nothing to do (no "break" instruction)
    case 1://This case corresponds to "the reader does not identify any tag"
        checkFailureCauses("Pas de tags identifiés", logToAnalyse);
        if(the cause is found){
            break;
        }
    case 2://This case corresponds to "the reader misses to identify some tags"
        checkFailureCauses("Inventaire incomplet ", logToAnalyse);
        if(the cause is found){
            break;
        }
    case 3://This case corresponds to "there is a problem in accessing the tag's memories"
        checkFailureCauses("Problème d'accès aux mémoires des tags", logToAnalyse);
        if(the cause is not found){
            print("No cause is found. The verification process will exit...");
            exit();
        }
}
```

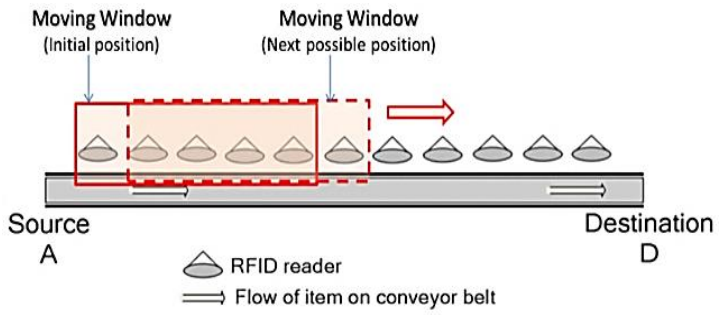
Figure 70. Vue globale de l'implémentation du processus de vérification des causes

IV. Conclusion

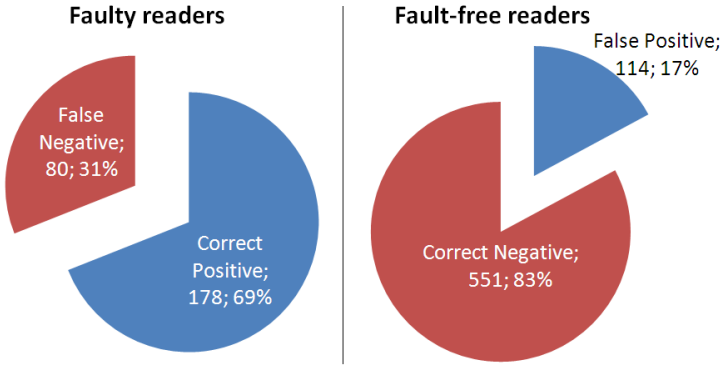
Nous avons vu dans ce chapitre comment faire face aux défaillances RFID en exploitant une technique existante, connue sous le nom de "Distinguishing Sequences". Cette technique permet d'identifier l'état interne du lecteur utilisant le protocole LLRP et ainsi, accélérer le processus de diagnostic. Ensuite, nous avons proposé une extension du modèle LLRP afin de prendre en charge les défaillances non traitées par le mécanisme des "Distinguishing Sequences". Enfin, nous avons présenté un mécanisme de diagnostic hors-ligne basé sur l'analyse des fichiers log LLRP. Il permet d'identifier la cause de la défaillance en comparant des signatures de messages XML avec les événements LLRP du système RFID. Ce même processus peut être adapté pour une utilisation en ligne. En effet, l'approche proposée peut analyser directement et en temps réel le flux de données échangées entre le middleware et le lecteur au lieu d'analyser les fichiers log.

Dans le chapitre suivant, nous présenterons trois implémentations de l'algorithme RFID diagAlgo présenté dans le chapitre 3 avec leur simulation sous différents scénarios afin de vérifier leur efficacité de détection des défaillances.

Chapitre 5
Implémentation et simulation de l'algorithme RFID diagAlgo




```
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.
----------------------------------------------------------------
```



Sommaire

Mots Clés

RFID diagAlgo
Probabilité de défaillance
Temps d'exécution
Métriques d'Halstead
Identifiabilité
Rational Behavior
Fenêtre d'analyse
Implémentation

CHAPITRE 5
IMPLEMENTATION ET SIMULATION DE L'ALGORITHME RFID DIAGALGO

I. Introduction

L'algorithme de diagnostic probabiliste RFID DiagAlgo est composé de 3 étapes comme nous l'avons vu dans le chapitre 3. La première étape est le regroupement des lecteurs suivant le flux de données. La deuxième étape est la comparaison statistique des résultats de lecture des différents lecteurs pour identifier toute déviation du comportement des lecteurs ou des tags. Nous rappelons que notre algorithme se base sur un paramètre de performance donné et que les résultats à comparer sont des matrices binaires qui se résument au respect ou non de la limite du paramètre de performance choisi (voir Chapitre 3 pour plus de détails). La troisième et dernière étape fait correspondre pour chaque défaillance détectée une probabilité que nous avons appelée Identifiabilité. Cette probabilité que nous qualifions de précision du diagnostic nous donne une estimation sur la fiabilité et la capacité de notre approche à bien identifier les composants RFID défaillants (lecteurs ou tags) et les composants valides, tout en tenant compte des conditions d'exécution telles que le nombre de lecteurs et de tags présents, la probabilité de défaillance d'un lecteur qui est calculée statistiquement à la 2ème étape de l'algorithme RFID DiagAlgo.

L'exécution de l'algorithme RFID DiagAlgo est très rapide, mais la phase de récupération des résultats des lecteurs prend un temps considérable dans le mode réel. De ce fait, il est nécessaire de proposer plusieurs implémentations en décomposant les groupes de lecteurs en sous-groupes, en définissant le nombre de groupes de tags à analyser à chaque étape et en adaptant la méthode de comparaison des résultats de façon à minimiser le temps d'attente des résultats, tout en gardant une précision de diagnostic acceptable. Il reste aussi à estimer le paramètre *Rational Behavior* en utilisant notamment les métriques de complexité d'Halstead[49] [75] qui permettent d'estimer le nombre d'erreurs dans un programme informatique et de déduire la probabilité qu'une erreur se manifeste. Enfin, nous simulerons cet algorithme sous différents scénarios de test afin de vérifier son efficacité.

II. Evaluation du modèle probabiliste

Pour étudier le comportement de l'Identifiabilité suivant les différents paramètres ; (*i.e.*, l'influence de chaque paramètre sur l'Identifiabilité), nous avons fait varier les paramètres r (*Rational Behavior*) et p (la probabilité de défaillance d'un lecteur) de 0 à 1, n (le nombre de lecteurs) de 2 à 20 et t (le nombre de tags) de 1 à 5.
Les figures 71, 72 et 73 montrent quelques résultats de cette simulation.

[49] Métriques introduites par Maurice Howard Halstead en 1977. Ce sont des métriques qui se basent sur le nombre d'opérateurs et d'opérandes dans un programme pour estimer le nombre d'erreurs qu'il est susceptible de contenir.

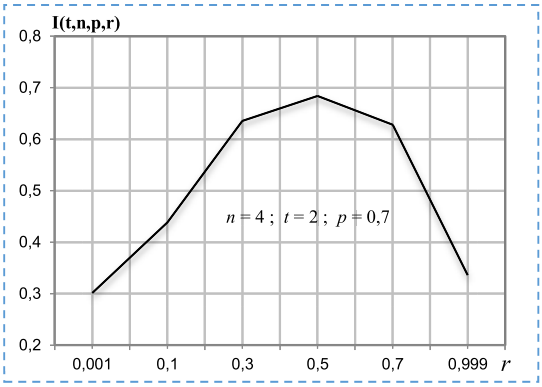


Figure 71. Variation de l'Identifabilité suivant le paramètre r

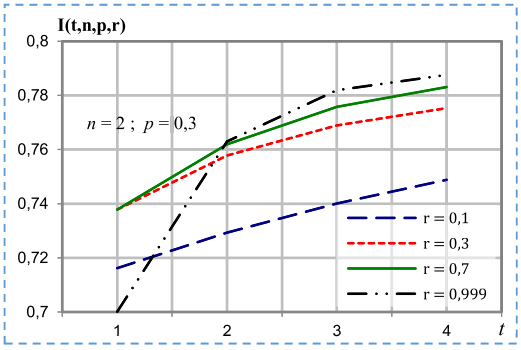


Figure 72. Variation de l'Identifabilité suivant le nombre de groupes de tags (t)

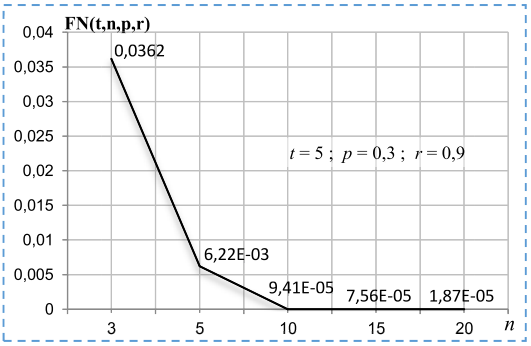


Figure 73. Effet du nombre de lecteurs sur FN

Explication

Avec $p = 0.7$ sur la figure 71, la plupart des lecteurs sont défaillants.

- Quand r est petit, la plupart des lecteurs défaillants ne manifestent pas leur défaillance, et donc ils ne sont pas détectés vu qu'ils ont les mêmes résultats que les lecteurs corrects (*i.e.*, la majorité des lecteurs sont défaillants et ne manifestent pas leurs défaillances).
- Avec l'augmentation de *r* jusqu'à atteindre environ 0.5 ; (*i.e.*, le nombre de lecteurs qui manifestent leurs défaillances augmente), l'Identifiabilité I augmente, vu que tous les lecteurs qui manifestent leurs défaillances représentent la minorité et sont donc détectés.
- Une fois que r a dépassé 0.5, son effet sur I s'inverse ; *i.e.*, la plupart des lecteurs défaillants manifestent leurs défaillances et donc, ils représentent la majorité et par conséquence ils sont considérés comme non défaillants (*i.e.,* l'Identifiabilité diminue).

La figure 72 montre l'effet du nombre de groupes de tags (*t*) sur I. L'augmentation de *t* implique l'augmentation de I quelle que soit la valeur de r. Aussi, I augmente avec r sans effet inverse lorsque $r > 0.5$ comme nous l'avons déjà vu dans la figure 71. Cela est dû à la probabilité de défaillance des lecteurs qui est faible $(p = 0.3)$; *i.e.*, les lecteurs défaillants n'ont aucune chance de représenter la majorité pour qu'ils soient déclarés non défaillants.

L'objectif de RFID diagAlgo est de réduire les faux négatifs (paramètre FN). La figure 73 met l'accent sur ce phénomène qui est rendu possible grâce notamment à la 2ème étape de notre algorithme (*i.e.*, un lecteur n'est considéré correct par rapport à un groupe de tags que s'il dispose d'un même résultat que

la *majorité*), mais aussi, grâce au nombre de lecteurs qui augmente au sein du même groupe. Nous voyons sur cette figure que lorsque le nombre de lecteurs dépasse 10, *FN* est quasiment nul. Cela montre une autre utilité qui est de pouvoir fixer le nombre nécessaire de lecteurs et/ou de groupes de tags pour avoir un meilleur diagnostic.

Nous pouvons aussi utiliser cette approche pour déterminer combien de groupes de tags, chaque lecteur du système doit analyser pour avoir une meilleure précision dans la détection des défaillances. La table 7 montre un exemple de situation où la probabilité de défaillance d'un lecteur est fixée à $p = 0.1$ et la probabilité *Rational Behavior* à $r = 0.5$. Cette table nous permet de fixer le nombre de lecteurs suivant le nombre de groupes de tags dont dispose le système ou vice-versa. Par exemple, si le système dispose de 5 lecteurs RFID, la précision de l'algorithme de diagnostic atteint sa valeur maximale avec 9 groupes de tags ; *i.e.*, les lecteurs doivent analyser 9 groupes de tags pour avoir une précision de diagnostic (Identifiabilité) à $I(t, n, p, r) = 0{,}997426$. Nous remarquons aussi que l'Identifiabilité est quasiment la même dès l'analyse de 4 groupes de tags. Nous pouvons alors déduire, toujours à l'aide de cette approche qu'il suffit d'exécuter l'algorithme de diagnostic à chaque lecture de 4 groupes de tags. Cela permet d'accélérer le processus de détection des défaillances et de ne pas perdre de temps à attendre la lecture des autres groupes de tags, tout en gardant la précision du diagnostic.

Table 7. Variation de l'Identifiabilité suivant le couple (t, n)

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

III. Estimation du paramètre Rational Behavior

Nous avons déjà vu que l'Identifiabilité dépend de 4 paramètres, et comme mentionné précédemment, la probabilité *Rational Behavior* (r) est spécifique pour chaque lecteur et concerne seulement ses défaillances ; *e.g.*, antenne endommagée, erreurs provenant du firmware du lecteur, *etc*. Une des possibilités pour estimer la probabilité r est de considérer les chemins d'exécution logiciels ou matériels. La figure 74 (b) montre un graphe de flot de contrôle correspondant au programme de la figure 74 (a). Ce graphe montre que le programme peut suivre 4 chemins différents durant son exécution.

Imaginons maintenant qu'au niveau de l'instruction $x := x + 1$ correspondant au nœud F du graphe, nous avons commis une erreur telle que $x := x - 1$. D'après le graphe, il existe deux chemins sur quatre qui passent par le nœud F et donc qui permettent à l'erreur de se manifester.

Maintenant, nous généralisons ce cas sur un programme à N chemins d'exécutions. En exploitant les mesures d'Halstead, nous pouvons estimer le nombre A d'erreurs dans le programme qui ont un effet visible sur le système [75]. Ainsi, la probabilité r ou la probabilité pour qu'au moins une des erreurs se manifestent en supposant une distribution homogène des erreurs sur les chemins d'exécution est $r = A / N$.

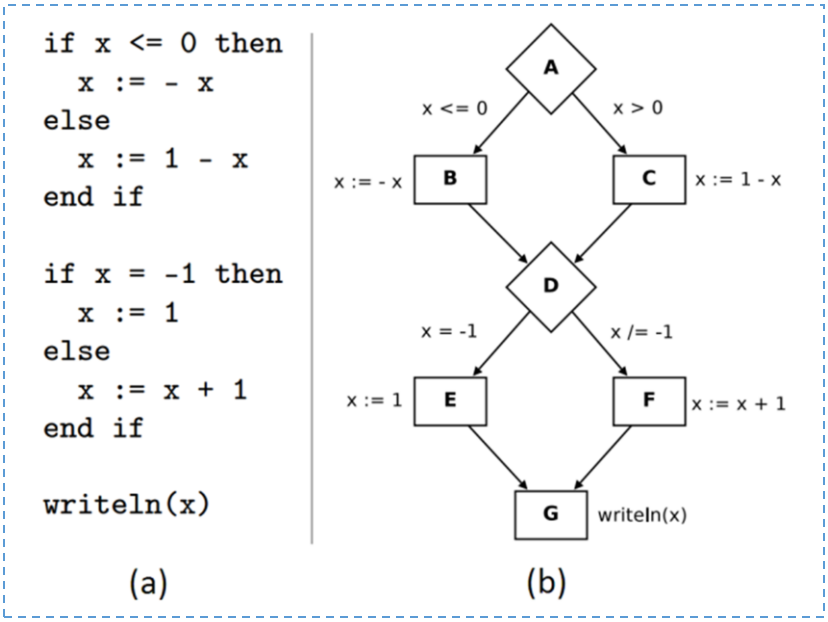


Figure 74. Graphe de flot de contrôle

Néanmoins, dans un déploiement RFID réel, les lecteurs RFID sont souvent des boîtes noires. Nous n'avons pas accès à l'architecture détaillée ou au code source du lecteur. Il devient alors impossible d'estimer le paramètre r. Dans ce cas, le paramètre r sera fixé à une valeur importante comme 0,999 ou même 1. Nous rappelons que r est la probabilité pour qu'un lecteur défaillant manifeste sa défaillance (*i.e.*, il traite incorrectement les tags). Ainsi, lorsque r est fixé à 1, nous nous intéressons qu'aux défaillances dont les effets sont visibles sur le système observé.

IV. Implémentation de RFID diagAlgo

RFID diagAlgo fonctionne comme suit. Après la phase de regroupement des lecteurs, l'algorithme procède aux comparaisons des résultats de lecture. A l'issue de cette étape, des lecteurs et/ou des groupes de tags sont marqués comme « probablement défaillants ». L'algorithme RFID diagAlgo représenté en annexe D a une complexité de $O(t \times 2n)$. Nous avons délibérément choisi cette représentation pour faciliter sa compréhension. Pour information, le temps d'exécution de l'algorithme une fois qu'il dispose de tous les résultats de lecture, ne dépasse pas 100 millisecondes[50]. Mais la démarche globale proposée pour le diagnostic des systèmes RFID prend beaucoup plus de temps. En effet, cette démarche repose sur la disponibilité de tous les résultats de tous les lecteurs. Ainsi, l'algorithme RFID diagAlgo ne sera lancé qu'une fois tous les résultats de lecture disponibles. Ce temps d'attente varie suivant le nombre de lecteurs, le nombre de groupes de tags à analyser, la disposition géographique des lecteurs et la vitesse de déplacement des tags d'un lecteur à un autre.

Ce temps influence négativement l'efficacité de l'approche proposée en retardant considérablement le temps de détection des défaillances. Pour répondre à cette problématique, nous avons proposé trois implémentations de l'algorithme RFID diagAlgo. La première, qui représente l'approche initiale, consiste à attendre tous les résultats de tous les lecteurs avant de commencer l'analyse. La deuxième implémentation consiste à analyser les lecteurs en groupes disjoints de 5. La troisième et dernière implémentation consiste à analyser les lecteurs au fur et à mesure que leurs résultats arrivent.

[50] RFID diagAlgo a été testé sur une machine à processeur Intel® Core™ i5 M560 à 2.67GHz.

1. Implémentation principale : "Calculation done at the end"

Dans cette implémentation, l'algorithme commence le processus du diagnostic une fois que toutes les données de lecture sont disponibles. Cette approche est intéressante du point de vue de la précision du diagnostic, vu que l'algorithme dispose d'une vue globale du système (de tous les résultats de lecture). A contrario, Cette approche prend beaucoup de temps pour les raisons évoquées précédemment. Le temps nécessaire pour la récupération des résultats des lecteurs varie de plusieurs secondes à quelques minutes alors que celui de l'analyse des résultats par l'algorithme ne dépasse pas 100 millisecondes. Ajouté à cela, ce temps d'attende augmente très rapidement avec l'augmentation du nombre de lecteurs ou de tags à analyser. Pour ces raisons, nous avons proposé deux autres approches alternatives dont le but est de réduire ce temps d'attente (*i.e.*, réduire le temps qui s'écoule entre le moment de la défaillance et le moment de sa détection) sans pour autant perdre en précision de diagnostic.

2. Implémentations alternatives

1.1. Calculation done five by five

Dans cette implémentation, le middleware lance l'analyse des lecteurs à chaque fois qu'il dispose des résultats de 5 lecteurs. Une fois que les 5 premiers lecteurs sont analysés, le middleware lance l'analyse des 5 lecteurs suivants et ainsi de suite. La taille des groupes de lecteur a été fixée à 5, car avec 5 lecteurs, le middleware n'a pas à attendre beaucoup de temps pour que les données à analyser soient disponibles et il dispose de suffisamment d'informations pour pouvoir effectuer un diagnostic correct.

1.2. Moving window implementation

Dans cette implémentation, le middleware analyse aussi les lecteurs en sous-groupes de 5, tout comme la précédente méthode. Par contre, cette méthode diffère de la précédente dans la procédure de sélection des 5 lecteurs suivants. En effet, à la première itération de l'analyse, le middleware doit attendre le résultat des 5 premiers lecteurs tout comme la méthode « Calculation done five by five ». Nous appellerons la procédure de sélection et d'analyse des lecteurs par « fenêtre d'analyse » ou « fenêtre glissante » à cause de son mouvement de déplacement tout au long des chemins des lecteurs à analyser. A la deuxième itération, une fois que le premier groupe de lecteurs est analysé, tous les lecteurs identifiés comme défaillants sont éjectés du groupe (de la fenêtre) et le même nombre de lecteurs non encore analysés sont sélectionnés pour faire partie du nouveau groupe à analyser. Si aucun des lecteurs n'est identifié comme défaillant dans la fenêtre courante, seul le premier lecteur du groupe est éjecté et la fenêtre avance d'un pas pour inclure le lecteur suivant dans la prochaine analyse comme le montre la figure 75.

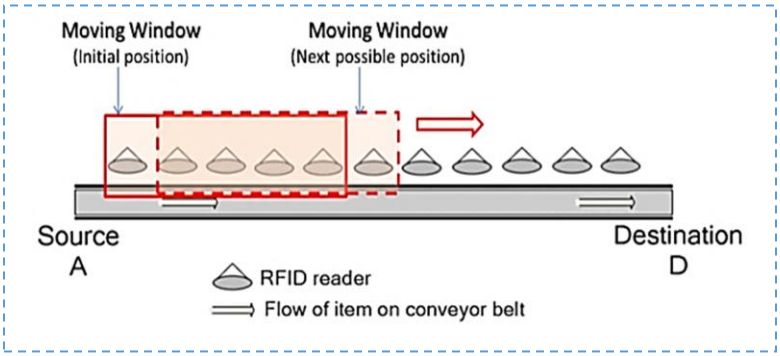


Figure 75. Moving window approach

V. Simulation et discussion des résultats

1. Description de la simulation

Les deux implémentations alternatives sont sans appel plus rapides que l'implémentation principale vue qu'elles n'attendent pas d'avoir tous les résultats de tous les lecteurs pour lancer l'algorithme de diagnostic. Etant donné que le temps d'analyse des données par l'algorithme est négligeable par rapport au temps nécessaire pour la disponibilité des données (quelques millisecondes contre plusieurs secondes voire plusieurs minutes dans le cas de la première approche), le défi principal est de vérifier si les deux approches alternatives restent précises même si leur procédure d'analyse ne se porte à la fois que sur une partie des données du système.

Pour évaluer ces deux implémentations alternatives, nous avons comparé leur taux de détection des défaillances par rapport à l'implémentation principale dans trois différents scénarios. Chaque scénario dispose de 44 cas de test. Chaque cas de test est représenté par une matrice binaire (lecteur, groupe de tags) qui représente les résultats de lecture de chaque lecteur sur chaque groupe de tags. Le nombre de lecteurs participant dans chaque cas de test varie de 3 à 60 lecteurs et le nombre de groupes de tags varie de 2 à 6. Sur l'ensemble des 44 cas de test de chaque scénario, le nombre total de lecteurs à diagnostiquer s'élève à 923. La table 8 donne un exemple d'un cas de test constitué de 10 lecteurs et de 4 groupes de tags.

Table 8. Exemple de cas de test avec 10 lecteurs et 4 groupes de tags

	g_0	g_1	g_2	g_3
R_0	1	1	0	1
R_1	1	1	0	0
R_2	1	1	0	0
R_3	1	0	1	0
R_4	1	0	1	0
R_5	1	0	1	0
R_6	1	0	1	0
R_7	1	0	1	0
R_8	1	0	1	0
R_9	1	1	1	0

Chaque ligne de la matrice de test représente les résultats de lecture ou de performance du lecteur. Nous rappelons qu'un paramètre de performance peut être le nombre d'erreurs commises par le lecteur à la lecture du groupe de tags, le nombre de tags identifiés par seconde, *etc*. (voir le chapitre 2 pour plus d'information). Dans notre simulation, nous ne nous intéressons pas au paramètre de performance choisi, mais plutôt au respect de sa limite par le couple (lecteur R_i, groupe de tags g_j). Quand $R_i(g_j)=1$ (*resp.*, $R_i(g_j)=0$), le couple (R_i , g_j) réalise des performances de lecture correctes (*resp.*, des performances de lecture médiocres) ; en d'autres termes, le lecteur R_i respecte la limite du paramètre de performance lors de la lecture du groupe de tags g_j.

Un lecteur est considéré défaillant s'il excède la limite de performance établie au moins sur un groupe de tags alors que la majorité des lecteurs a correctement lu le groupe de tags.

La table 9 montre le résultat du diagnostic des lecteurs effectué par les trois implémentations à partir du cas de test de Table 8. Cette table contient les probabilités de défaillance de chaque lecteur suivant chaque implémentation de l'algorithme de diagnostic. Cette table montre aussi les différents temps d'exécution de chaque implémentation (ce temps ne prend pas en charge les temps d'attente évoqués précédemment, qui sont nécessaires pour que toutes les données des lecteurs soient disponibles).

Dans les scénarios qui suivent, nous utilisons les appellations suivantes :

Table 9. Exemple de résultats de diagnostic

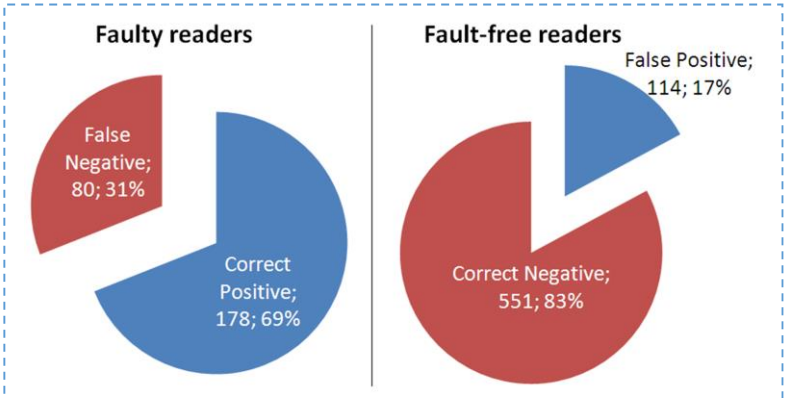
Moving window:	Calc. at the end:	Calc. 5 by 5:
R0: 0,9993592773	R0: 0,9846765334	R0: 0,9993592773
R1: 1,0000000000	R1: 0,9846765334	R1: OK
R2: 1,0000000000	R2: 0,9846765334	R2: OK
R3: 0,9999199097	R3: OK	R3: 0,9999199097
R4: 0,9999199097	R4: OK	R4: 0,9999199097
R5: OK	R5: OK	R5: OK
R6: OK	R6: OK	R6: OK
R7: OK	R7: OK	R7: OK
R8: OK	R8: OK	R8: OK
R9: 1,0000000000	R9: 0,9846765334	R9: 1,0000000000

Time needed (all methods): 0.088745357 seconds.
Time needed by:
Moving window method: 0.001501587 seconds.
Calc at the end method: 0.086095859 seconds.
Calc 5 by 5 method: 8.0541E-4 seconds.

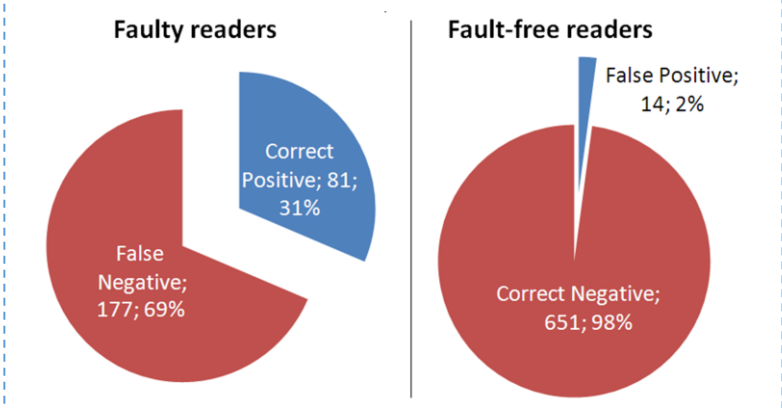
- ***Correct Negative***: un lecteur sain (valide, non défaillant) considéré comme tel par l'algorithme.
- ***Correct Positive***: un lecteur défaillant considéré défaillant (défaillance détectée).
- ***False Negative***: un lecteur défaillant considéré comme sain (défaillance non détectée).
- ***False Positive***: un lecteur valide considéré défaillant.

1.1. Scénario 1

Ce cas représente la simulation d'un environnement peu perturbé (*i.e.*, il y a moins de lecteurs défaillants que de lecteurs valides). Il est représenté par 44 cas de test (matrice de test) qui constituent au total 923 lecteurs comme indiqué précédemment. Ce scénario est représenté par 665 lecteurs sains et 258 lecteurs défaillants. Le résultat de la simulation est donné par la figure 76.



(a) *Moving Window Diagnosis*

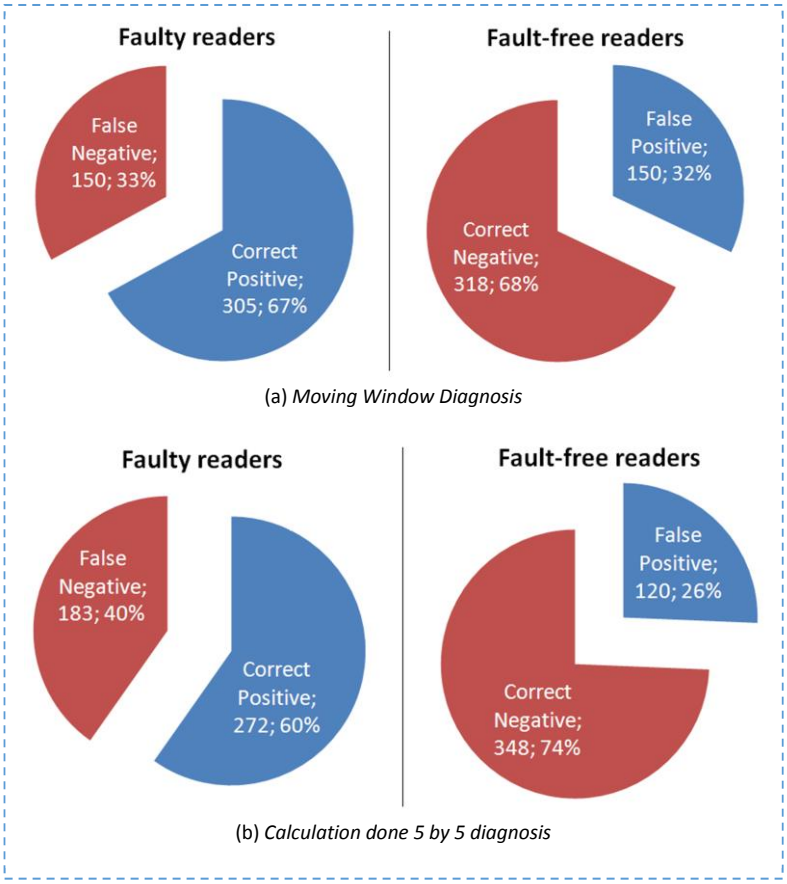


(b) *Calculation done 5 by 5 diagnosis*

Figure 76. Scénario 1

1.2. Scénario 2

Ce cas représente une simulation d'un environnement équilibré en termes de nombre de lecteurs défaillants (455 lecteurs) et de lecteurs valides (468 lecteurs). Le résultat de cette simulation est donné en figure 77.



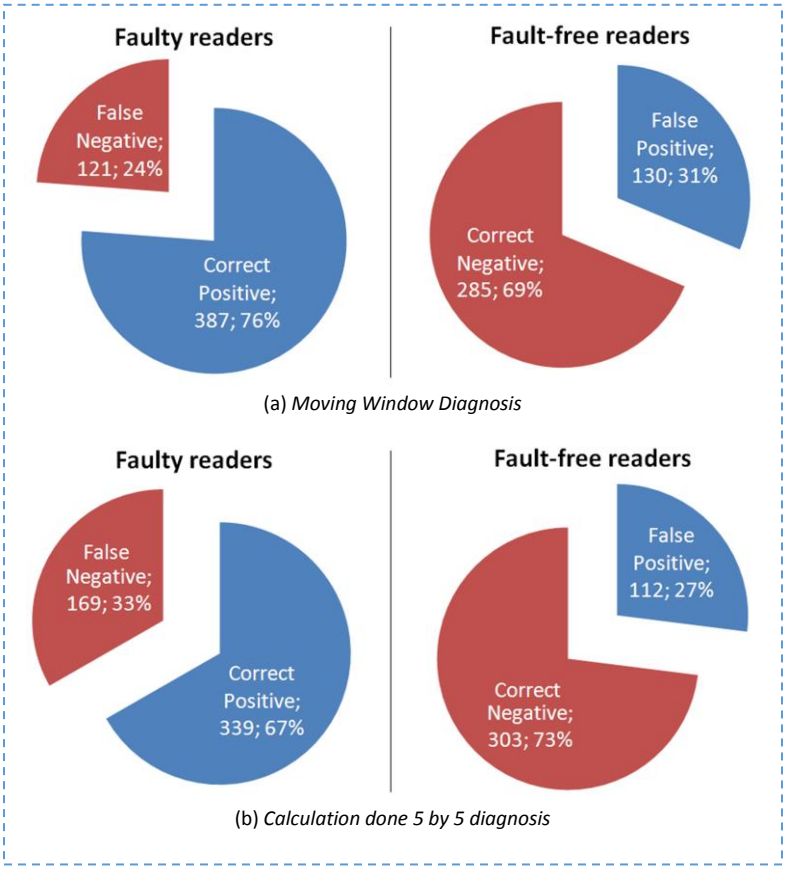
(a) *Moving Window Diagnosis*

(b) *Calculation done 5 by 5 diagnosis*

Figure 77. Scénario 2

1.3. Scénario 3

Ce cas représente une simulation d'un environnement dont le nombre de lecteurs défaillants est majoritaire. Ce cas est représenté par 508 lecteurs défaillants et 415 lecteurs valides. Les résultats de simulation de ce cas sont donnés par la figure 78.



(a) *Moving Window Diagnosis*

(b) *Calculation done 5 by 5 diagnosis*

Figure 78. Scénario 3

2. Discussion des résultats

Tous les résultats de simulation des différents scénarios montrent globalement la même tendance. L'approche « Moving Window » détecte plus de défaillances que l'approche « Calculation done 5 by 5 » indépendamment de l'environnement d'exécution. En contrepartie, elle génère plus de fausses alarmes (faux positifs). Ajouté à cela, cette approche est généralement plus rapide en termes de réactivité et de détection des défaillances. En effet, l'approche « Moving Window », après la première position de la fenêtre glissante permet d'analyser les lecteurs au fur et à mesure que leurs résultats arrivent (*i.e.*, le middleware n'attend que les résultats du prochain lecteur pour l'inclure dans la fenêtre d'analyse pour la prochaine itération), alors que l'approche « Calculation done 5 by 5 » oblige l'algorithme de diagnostic à attendre à chaque fois les résultats des 5 prochains lecteurs.

Les seuls cas où l'approche « Moving Window » est inefficace sont quand la majorité des lecteurs se trouvant dans la fenêtre d'analyse sont défaillants. Cela devient encore plus grave, si ce phénomène se produit dans les premières positions de la fenêtre d'analyse. En effet, étant donné que la majorité des lecteurs (au moins 3 lecteurs sur les 5) sont défaillants et que l'approche se base sur une comparaison statistique des résultats de lecture, le diagnostic est complètement faussé ; les lecteurs qui sont normalement défaillants sont considérés comme valides et les lecteurs valides comme défaillants et donc éjectés de la fenêtre glissante. Ce phénomène se poursuit vu que les lecteurs défaillants qui constituent la majorité sont toujours gardés dans la fenêtre.

Ainsi, le choix entre ces trois implémentations dépend essentiellement de la politique que le client ou l'utilisateur final a adoptée. Si le client veut détecter la défaillance des composants RFID au plus tôt

avec un taux de détection des défaillances élevé, l'approche « Moving Window » est la plus appropriée. Si le client veut un mécanisme de détection qui génère un minimum de faux positifs et qui fonctionne avec des temps raisonnables, l'approche « Calculation done 5 by 5 » est meilleure. Enfin, Si le client veut un mécanisme de diagnostic le plus fiable, peu importe s'il prend beaucoup plus de temps, l'approche « Calculation done at the end » est la plus adaptée.

VI. Conclusion

Dans ce chapitre, nous avons présenté une évaluation du modèle probabiliste utilisé par l'algorithme RFID diagAlgo afin d'analyser son comportement suivant différents paramètres tels que le nombre de lecteurs RFID ou le nombre de groupes de tags. Ensuite, nous avons présenté trois différentes implémentations de l'algorithme RFID diagAlgo que nous avons évaluées à travers plusieurs scénarios de test.

Le chapitre suivant concerne la conclusion générale de ce travail sur le test et le diagnostic des systèmes RFID. Dans cette conclusion, nous présenterons différentes pistes et perspectives de recherche pour poursuivre l'amélioration de la fiabilité et l'efficacité des systèmes RFID.

Conclusion générale

I. Contribution

Ce document a présenté les travaux de recherche effectués dans le cadre ma thèse de Doctorat en informatique au sein de l'équipe CTSYS du Laboratoire de Conception et d'Intégration des Systèmes. L'objectif de la thèse était de proposer des approches logicielles de test et de diagnostic des systèmes RFID au niveau de la couche middleware dans le but d'améliorer la fiabilité globale des systèmes RFID.

Dans un premier temps, nous avons étudié les systèmes RFID et les protocoles de communication utilisés afin de poser les connaissances de base nécessaires pour entamer la problématique de tolérance aux fautes dans les systèmes RFID. Dans un second temps, nous nous sommes penchés sur les techniques de monitoring existantes afin de démontrer pourquoi elles ne sont pas adaptées aux nouveaux usages des systèmes RFID. Enfin, afin de proposer des solutions bien adaptées, nous avons appliqué une AMDE (Analyse des Modes de Défaillance et de leurs Effets) sur les middlewares RFID pour identifier les défaillances à prendre en compte et de quelles façons.

Les solutions proposées dans ce mémoire sont au nombre de trois.

- Un algorithme de diagnostic probabiliste « RFID diagAlgo » qui se base sur analyse statistique des résultats de lecture afin d'identifier les composants défaillants. Il se compose de 3 étapes.
    - Regroupement des lecteurs à analyser suivant les flux des données RFID. Cela s'illustre par exemple, dans les systèmes de manutention des bagages dans les aéroports où les lecteurs sont disposés tout le long des chemins que les bagages empruntent.
    - Comparaison des données des lecteurs afin d'identifier ceux dont les résultats sont aberrants.
    - Estimation de la précision du diagnostic grâce à un modèle probabiliste adapté du diagnostic des systèmes multiprocesseurs.
- Une extension du protocole de communication entre les middlewares et les lecteurs RFID dont le but est de prendre en compte au niveau du protocole certaines défaillances. Ce mécanisme est présenté comme un processus de vérification hiérarchique d'un ensemble de causes matérielles (*e.g.*, antenne abîmée) ou dues aux conditions d'exécution (vitesse de déplacement des tags, présence de perturbations électromagnétiques, *etc.*).
- Un analyseur de fichiers log LLRP qui permet une fois la défaillance détectée de remonter dans la trace d'exécution grâce aux fichiers log et ainsi identifier les causes de cette défaillance. Dans cette partie, nous avons aussi vu pourquoi les « Distinguishing Sequences » ne sont pas adaptées au diagnostic des systèmes RFID, ce qui motive l'approche d'analyse des événements RFID.

Afin d'évaluer les mécanismes proposés dans des conditions d'exécution réelles, une solution middleware a été implémentée pour accueillir ces mécanismes. Ce middleware se base sur le protocole LLRP pour la communication avec les lecteurs à travers des commandes XML pour la récupération des données RFID et la gestion des différentes sources RFID. Enfin, nous avons présenté au chapitre 5 deux implémentations alternatives de l'algorithme RFID diagAlgo que nous avons évalué afin de réduire le temps de diagnostic et augmenter la réactivité de l'approche, tout en gardant sa précision et sa fiabilité.

II. Perspectives

La sûreté de fonctionnement des systèmes RFID est une problématique très complexe. L'étude menée tout au long de cette thèse a essayé de répondre à cette problématique. Néanmoins, de nouveaux travaux de recherche sont toujours nécessaires afin d'améliorer la fiabilité et l'efficacité de la technologie RFID. Parmi ces travaux, nous pouvons citer comme perspectives de recherche :

- Améliorer les implémentations de l'algorithme RFID diagAlgo. En effet, les implémentations alternatives, même si elles améliorent la vitesse d'exécution, présentent encore des faux positifs (fausses alertes : lecteur valide considéré défaillant) et des faux négatifs (alerte non déclenchée : lecteur défaillant considéré valide) qu'il faut réduire. Pour cela, nous pouvons envisager de s'intéresser à surveiller les lecteurs RFID sur plusieurs paramètres de performances (*e.g.*, RETR + vitesse de lecture « *nombre de lecture réussi / seconde* »).
- Améliorer le calcul du paramètre *Rational Behavior*. Nous avons présenté une approche permettant d'estimer ce paramètre. Elle se base sur les chemins d'exécutions du programme et grâce aux mesures d'Halstead [75], nous pouvons calculer la probabilité pour qu'une erreur se manifeste dans le système. Néanmoins, dans un déploiement RFID réel, les lecteurs RFID sont souvent des boîtes noires ; *i.e.*, nous n'avons pas accès à l'architecture détaillée ou au code source du lecteur. Il devient alors impossible d'estimer le paramètre *Rational Behavior*. Comme perspective d'amélioration de l'estimation de ce paramètre, nous pouvons nous pencher sur l'estimation du MTBF (Mean Time Between Failures) qui permet de déterminer la durée moyenne entre deux défaillances successives du système [76] [77]. En effet, la définition de *Rational Behavior* en est très proche. De ce fait, nous pouvons donc utiliser une des méthodes qui permettent d'estimer le MTBF ou directement prendre la valeur du MTBF du composant comme valeur de *Rational Behavior*.
- L'inconvénient de l'approche d'analyse des logs LLRP présentée au chapitre 4 est qu'il s'agit d'un mécanisme de surveillance hors-ligne. Il ne permet pas un diagnostic du système en temps réel. Nous pouvons modifier ce mécanisme de façon à ce que le flux de données à analyser soit récupéré directement du système surveillé sans passer par un fichier log. De cette façon, l'approche proposée pourra effectuer un diagnostic en ligne des défaillances détectées. Néanmoins, il serait important d'effectuer une évaluation de l'impact de cette solution sur les performances du système surveillé ; le système doit faire face à une charge supplémentaire qui se résume à l'envoi en temps réel du flux de données au mécanisme d'analyse des événements LLRP, ce qui risque de réduire ses performances.
- Les approches de tolérance aux fautes proposées reposent sur la fiabilité du middleware, ce qui constitue un inconvénient majeur. Il serait préférable de pouvoir distribuer le monitoring du système vers les applications des utilisateurs finaux et ainsi bénéficier de la redondance naturelle des applications d'exploitation les données RFID tout en évitant de gêner le fonctionnement normal de ces applications.
- Etudier la relation entre le paramètre « TagPopulation» du protocole LLRP et le paramètre Q du protocole EPC C1 Gen2. En effet, comme indiqué précédemment le paramètre TagPopulation qui est fixé dans une ROSpec par l'utilisateur influe sur la valeur de Q que le lecteur choisit pour gérer les collisions des signaux des tags comme évoqué dans le chapitre 4. Si l'utilisateur indique une valeur non appropriée de TagPopulation, cela engendrera plus de collisions et ainsi, affectera les performances du lecteur. Il serait important de conduire une étude approfondie sur le lien entre ces deux paramètres et qui permettra par la suite de mieux contrôler la configuration de LLRP en surveillant les valeurs saisies par l'utilisateur.

Bibliographie

[1] L. Jerry, «Shrouds of Time - The history of RFID,» The Association for Automatic Identification and Mobility, 01 October 2001. Available: http://www.aimglobal.org. [Accès le December 2010].

[2] M. Roberti, «The History of RFID Technology,». Available: http://www.rfidjournal.com/article/view/1338.

[3] S. Agusti, J. Domingo-Ferrer, A. Martinez-Balleste et V. Daza, «A distributed architecture for scalable private RFID tag identification,» Elsevier , Catalonia, Spain, 2007.

[4] D. Donsez, «Vers des intergiciels RFID,» 06 05 2008. Available: http://membres-liglab.imag.fr/donsez/cours/rfidmiddleware.pdf.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] R. Kheddam, O.-E.-K. Aktouf et I. Parissis, «Algorithme de diagnostic probabiliste pour la détection des défaillances dans les systèmes RFID,» 9ème Conférence Internationale Jeunes Chercheurs (MajecSTIC), Lille, France, 2012, pp. 1-8.

[11] 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.

[12] R. KHEDDAM, O.-E.-K. AKTOUF and I. PARISSIS, "An extended LLRP model for RFID system test and diagnosis," *5th IEEE International Conference on Software Testing, Verification and Validation. Montreal, Canada,* pp. 529-538, 2012.

[13] R. KHEDDAM, O.-E.-K. AKTOUF, I. PARISSIS et S. BOUGHAZI, «Monitoring of RFID Failures Resulting from LLRP Misconfigurations,» *21th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM). Split, Croatia,* pp. 1-6, 2013.

[14] H. Lehpamer, RFID Design Principles, Norwood, MA: ARTECH HOUSE, INC., 2008.

[15] CNRFID - Centre national de référence RFID, «RFID CNRFID - Introduction à la RFID,» [En ligne]. Available: http://www.centrenational-rfid.com/la-rfid/introduction-à-la-rfid/article/15/fr.html.

[16] Aspire RFID Project, «RFID History and Standards,» décembre 2009. Available: http://www.fp7-aspire.eu/RFID/RFID_history_standards_french/.

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

[18] Pôle National de Traçabilité, "Composants du système RFID,". Available: http://www.polenationaldetracabilite.com/composants-système-rfid,p-46,fr.html.

[19] EPCglobal Inc., "EPC Radio-Frequency Identification Protocols Class-1 Generation-2 UHF RFID (Version 1.2)," 23 October 2008. Available: http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.

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

[22] Institut National de Recherche et de Sécurité, «Champs électromagnétiques - La RFID - ED 4217,» Mars 2010.

[23] D. W. Engels, "The Reader Collision Problem, Technical Report. MIT-AUTOID-WH-007," 2001. Available: http://www.autoidlabs.org/uploads/media/MIT-AUTOID-WH-007.pdf.

[24] S. E. Sarma, S. A. Weis et D. W. Engels, «white paper - RFID Systems, Security & Privacy Implications,» Massachusetts, USA, November 2002.

[25] RFID Journal, «http://www.rfidjournal.com/glossary,». Available: http://www.rfidjournal.com.

[26] Manianguisse Mond'industriel, «La RFID : Radio Frequency IDentification,» [En ligne]. Available: http://manianguisse.ifrance.com/Main/ProdStragIndust/Ingenierie/La RFID.html.

[27] GS1 France, «Introduction à la RFID et à EPCglobal,». Available: http://www.gs1.fr/gs1_fr/standards_gs1__1/epc_rfid/les_standards_epc/informations_sur_la_rfid_et_sur_les_standards_d_epcglobal/introduction_a_la_rfid_et_a_epcglobal.

[28] NEPC (National Electronic Product Catalogue), "EAN International,". Available:

http://www.nepc.gs1.org.sg/epcglobal/main.htm.

[29] S. Lahiri, RFID Sourcebook, 1 ed., IBM Press, August 2005, pp. 1-304.

[30] GS1 EPCglobal, "Tag class definitions," November 2007. Available: http://www.gs1.org/docs/epcglobal/TagClassDefinitions_1_0-whitepaper-20071101.pdf.

[31] J.-L. Chen, M.-C. Chen, C.-W. Chen and Y.-C. Chang, "Architecture design and performance evaluation of RFID object tracking systems," *Computer Communications,* vol. 30, no. 9, pp. 2070-2086, June 2007.

[32] D. C. Ranasinghe, Q. Z. Sheng and S. Zeadally, Unique Radio Innovation for the 21st Century - Building Scalable and Global RFID Networks, Springer, 2010.

[33] GS1 Belgilux, "Architecture réseau EPC,". Available: http://www.gs1belu.org/fr/architecture-reseau-epc.

[34] M. C. O'Connor, "Gen 2 EPC Protocol Approved as ISO 18000-6C," RFID JOURNAL, 11 July 2006. Available: http://www.rfidjournal.com/article/articleview/2481.

[35] P. MORREALE and K. TERPLAN, CRC Handbook of MODERN TELECOMMUNICATIONS, New York: CRC Press, Taylor & Francis Group, 2010.

[36] EPCglobal Inc., "Low Level Reader Protocol (LLRP) Version 1.1 Ratified Standard," October, 2010.

[37] Agence Wallonne des Télécommunications, «Fonctionnement et applications du RFID,» 22 02 2005. Available: http://www.awt.be/web/res/index.aspx?page=res,fr,fic,130,002.

[38] Press Releases, "HKIA Boosts Baggage Handling Efficiency with RFID Technology," 15 January 2008. Available: http://www.hongkongairport.com/eng/media/press-releases/pr_914.html.

[39] 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.

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

[41] R. Derakhshan, M. E. Orlowska and X. Li, "RFID Data Management: Challenges and Opportunities," IEEE International Conference on RFID, USA, 2007.

[42] B. Prabhu, X. Su, C. Qiu, H. Ramamurthy, P. Chu and R. Gadh, "WinRFID – Middleware for Distributed RFID Infrastructure," International Workshop on Radio Frequency Identification (RFID) and Wireless Sensors, Kanpur, India, 11-13 November 2005.

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

[44] 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.

[45] F. Liu, Y. Jie and W. Hu, "Distributed ALE in RFID Middleware," 4th IEEE International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM '08., 2008.

[46] N. Ahmed and R. Umakishore, "A Path Based Reliable Middleware Framework for RFID Devices," IEEE, Atlanta, USA, 2010.

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

[48] 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.

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

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

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

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

[53] Sun Microsystems, Inc., "Sun Java™ System RFID Software 3.0 Administration Guide," February 2006.

[54] Sun Microsystems, Inc., "Sun Java™ System RFID Software 3.0 Developer's Guide," February 2006.

[55] J. Chamberlain, C. Blanchard, S. Burlingame, S. Chandramohan, E. Forestier, G. Griffith, M. L. Mazzara, S. Musti, S.-I. Son, G. Stump and C. Weiss, IBM WebSphere RFID Handbook : A Solution Guide, IBM, 2006.

[56] 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.

[57] 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.

[58] D. K. Pradhan, «Fault-tolerant computer system design,» *Prentice Hall,* February 1996. ISBN: 0-13-057887-8.

[59] Schneider Electric, «La Sûreté de Fonctionnement,» Novembre 2004. Available:

http://www.intersections.schneider-electric.fr/stock_images/telec/1/n3/GT12.pdf.

[60] National Aeronautics and Space Administration, «NASA Software Safety Guidebook,» March 2004. Available: http://www.hq.nasa.gov/office/codeq/doctree/871913.pdf.

[61] M. L. Shooman, "Reliability of computer systems and networks : Fault Tolerance, Analysis, and Design," *Wiley-Interscience Publication,* 2002. ISBN: 0-471-29342-3.

[62] L. Cauffriez, J. Ciccotelli, B. Conrard et M. Bay, «Design of intelligent distributed control systems: a dependability point of view,» *Elsevier, Inc., Journal of Reliability Engineering and System Safety,* pp. 19-32, April 2004.

[63] 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.

[64] 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.

[65] G. Fritz, B. Maaloul, V. Beroulle, O.-E.-K. Aktouf and D. Hély, "Read rate profile monitoring for defect detection in RFID Systems," pp. 89-94, Sitges, Espagne, 2011.

[66] R. Kheddam, O.-E.-K. Aktouf and I. Parissis, "SafeRFID-MW: a RFID Middleware with runtime fault diagnosis," *Journal of Communications Software and Systems,* vol. 9, no. 1, pp. pp. 57-73, 2013.

[67] 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.

[68] 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.

[69] 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, Syngress Publishing, Inc. Elsevier, Inc., 2007.

[70] 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.

[71] 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.

[72] F.C. Hennie, «Fault detecting experiments for sequential circuits,» *in Proc. FOCS,* pp. 95-110, 1964.

[73] "LLRP Toolkit," Available: http://www.llrp.org.

[74] Apache Inc., "Logging Services - Log4j Guide," Apache Inc. Available: http://logging.apache.org/log4j.

[75] 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.

[76] W. Torell and V. Avelar, "Mean Time Between Failure: Explanation and Standards," *Schneider Electric's, White Paper 78,* pp. 1-10, 2011.

[77] S. Stanley, "MTBF, MTTR, MTTF & FIT - Explanation of Terms," *IMC Networks,* 2011.

[78] D. J. Glasser, K. W. Goodman and N. G. Einspruch, "Chips, tags and scanners: Ethical challenges for radio frequency identification," *Ethics and Information Technology,* vol. 9, no. 2, pp. 101-109, 2007.

[79] EPCglobal Inc., "Low Level Reader Protocol (Version 1.1)," October 2010. Available: http://www.gs1.org/gsmp/kc/epcglobal/llrp.

[80] 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.

Annexes

Annexe A : Exemple d'une AccessSpec

```
<?xml version="1.0" encoding="UTF-8"?>
<llrp:ADD_ACCESSSPEC xmlns:llrp="http://www.llrp.org/ltk/schema/core/encoding/xml/1.0"
Version="1" MessageID="5">
  <llrp:AccessSpec>
    <llrp:AccessSpecID>22</llrp:AccessSpecID>
    <llrp:AntennaID>0</llrp:AntennaID>
    <llrp:ProtocolID>EPCGlobalClass1Gen2</llrp:ProtocolID>
    <llrp:CurrentState>Disabled</llrp:CurrentState>
    <llrp:ROSpecID>1</llrp:ROSpecID>
    <llrp:AccessSpecStopTrigger>
      <llrp:AccessSpecStopTrigger>Operation_Count</llrp:AccessSpecStopTrigger>
      <llrp:OperationCountValue>3</llrp:OperationCountValue>
    </llrp:AccessSpecStopTrigger>
    <llrp:AccessCommand>
      <llrp:C1G2TagSpec>
        <llrp:C1G2TargetTag>
          <llrp:MB>1</llrp:MB>
          <llrp:Match>1</llrp:Match>
          <llrp:Pointer>0</llrp:Pointer>
          <llrp:TagMask>ffffffffffffffff</llrp:TagMask>
          <llrp:TagData>abcd123456aacc56</llrp:TagData>
        </llrp:C1G2TargetTag>
      </llrp:C1G2TagSpec>
      <llrp:C1G2Write>
        <llrp:OpSpecID>1111</llrp:OpSpecID>
        <llrp:AccessPassword>0</llrp:AccessPassword>
        <llrp:MB>1</llrp:MB>
        <llrp:WordPointer>3</llrp:WordPointer>
        <llrp:WriteData>0000</llrp:WriteData>
      </llrp:C1G2Write>
      <llrp:C1G2Write>
        <llrp:OpSpecID>2222</llrp:OpSpecID>
        <llrp:AccessPassword>0</llrp:AccessPassword>
        <llrp:MB>1</llrp:MB>
        <llrp:WordPointer>7</llrp:WordPointer>
        <llrp:WriteData>ccdd</llrp:WriteData>
      </llrp:C1G2Write>
      <llrp:C1G2Read>
        <llrp:OpSpecID>3333</llrp:OpSpecID>
        <llrp:AccessPassword>0</llrp:AccessPassword>
        <llrp:MB>1</llrp:MB>
        <llrp:WordPointer>0</llrp:WordPointer>
        <llrp:WordCount>8</llrp:WordCount>
      </llrp:C1G2Read>
    </llrp:AccessCommand>
    <llrp:AccessReportSpec>
      <llrp:AccessReportTrigger>Whenever_ROReport_Is_Generated</llrp:AccessReportTrigger>
    </llrp:AccessReportSpec>
  </llrp:AccessSpec>
</llrp:ADD_ACCESSSPEC>
```

Une spécification AccessSpec est l'un des messages principaux du protocole LLRP. Elle est utilisée pour spécifier des opérations de lecture et/ou d'écriture sur les tags RFID. Ce message envoyé par le middleware au lecteur, permet d'indiquer à ce dernier quelle interface air utiliser et quel message ROSpec servira à lancer les opérations définies dans cette AccessSpec. Le nombre d'opérations de lecture et d'écriture dépend du lecteur RFID utilisé. Ces opérations d'entrée/sortie sont définies grâce aux balises C1G2Read et C1G2Write dans un block XML global AccessCommand. L'exécution des opérations définies est conditionnée par un filtre défini avec la balise C1G2TargetTag que les tags doivent passer pour être sélectionnés.

Annexe B : Exemple d'une ROSpecSpec

```
<?xml version="1.0" encoding="UTF-8"?>
<llrp:ADD_ROSPEC xmlns:llrp="http://www.llrp.org/ltk/schema/core/encoding/xml/1.0"
Version="1" MessageID="4">
  <llrp:ROSpec>
    <llrp:ROSpecID>1</llrp:ROSpecID>
    <llrp:Priority>0</llrp:Priority>
    <llrp:CurrentState>Disabled</llrp:CurrentState>
    <llrp:ROBoundarySpec>
      <llrp:ROSpecStartTrigger>
        <llrp:ROSpecStartTriggerType>Null</llrp:ROSpecStartTriggerType>
      </llrp:ROSpecStartTrigger>
      <llrp:ROSpecStopTrigger>
        <llrp:ROSpecStopTriggerType>Null</llrp:ROSpecStopTriggerType>
        <llrp:DurationTriggerValue>0</llrp:DurationTriggerValue>
      </llrp:ROSpecStopTrigger>
    </llrp:ROBoundarySpec>
    <llrp:AISpec>
      <llrp:AntennaIDs>0</llrp:AntennaIDs>
      <llrp:AISpecStopTrigger>
        <llrp:AISpecStopTriggerType>Null</llrp:AISpecStopTriggerType>
        <llrp:DurationTrigger>0</llrp:DurationTrigger>
      </llrp:AISpecStopTrigger>
      <llrp:InventoryParameterSpec>
        <llrp:InventoryParameterSpecID>9</llrp:InventoryParameterSpecID>
        <llrp:ProtocolID>EPCGlobalClass1Gen2</llrp:ProtocolID>
        <llrp:AntennaConfiguration>
          <llrp:AntennaID>1</llrp:AntennaID>
          <llrp:C1G2InventoryCommand>
            <llrp:TagInventoryStateAware>1</llrp:TagInventoryStateAware>
          </llrp:C1G2InventoryCommand>
        </llrp:AntennaConfiguration>
      </llrp:InventoryParameterSpec>
    </llrp:AISpec>
    <llrp:ROReportSpec>
      <llrp:ROReportTrigger>Upon_N_Tags_Or_End_Of_ROSpec</llrp:ROReportTrigger>
      <llrp:N>5</llrp:N>
      <llrp:TagReportContentSelector>
        <llrp:EnableROSpecID>1</llrp:EnableROSpecID>
        <llrp:EnableSpecIndex>1</llrp:EnableSpecIndex>
        <llrp:EnableInventoryParameterSpecID>0</llrp:EnableInventoryParameterSpecID>
        <llrp:EnableAntennaID>1</llrp:EnableAntennaID>
        <llrp:EnableChannelIndex>0</llrp:EnableChannelIndex>
        <llrp:EnablePeakRSSI>1</llrp:EnablePeakRSSI>
        <llrp:EnableFirstSeenTimestamp>1</llrp:EnableFirstSeenTimestamp>
        <llrp:EnableLastSeenTimestamp>1</llrp:EnableLastSeenTimestamp>
        <llrp:EnableTagSeenCount>1</llrp:EnableTagSeenCount>
        <llrp:EnableAccessSpecID>0</llrp:EnableAccessSpecID>
        <llrp:C1G2EPCMemorySelector>
          <llrp:EnableCRC>1</llrp:EnableCRC>
          <llrp:EnablePCBits>1</llrp:EnablePCBits>
        </llrp:C1G2EPCMemorySelector>
      </llrp:TagReportContentSelector>
    </llrp:ROReportSpec>
  </llrp:ROSpec>
</llrp:ADD_ROSPEC>
```

Une ROSpec est un message XML utilisé par le middleware pour indiquer au lecteur RFID quels paramètres utiliser lors de la phase d'inventaire des tags. Ces paramètres peuvent être la spécification de l'interface air entre le lecteur et les tags, l'antenne à utiliser par le lecteur, le nombre de tags et le type d'information que chaque rapport doit contenir.

Annexe C : Failure Mode and Effects Analysis on RFID middleware's components

In this FMEA, we assume that the middleware is implemented as a set of layers and "independent" modules (modular architecture), and the list of DPL (Data Processing Layer) components is non-exhaustive. Indeed, the middleware's components may differ from one implementation to another. Nevertheless, considered components represent the main services and functions that the middleware is expected to provide. These components were summarized in Figure 26 of chapter 3. For convenience purposes, the same figure is also reported in this annex (Figure 79).

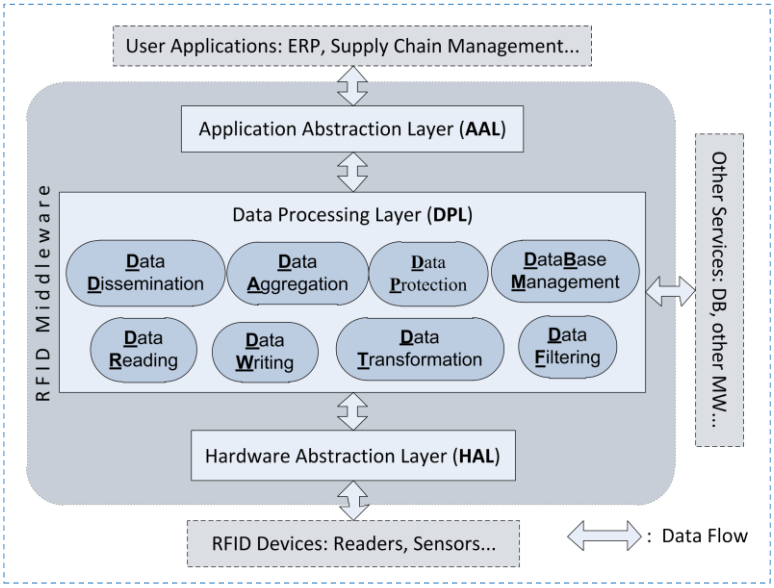


Figure 79. RFID Middleware Architecture

The FMEA is not easy to perform because there are errors hard to identify or to predict. Indeed, RFID system behaviour is very dependent on its runtime environment (presence of metal obstacles, electromagnetic disturbances, etc.) and the occurrence of an error is often the result of successive events (programming error, misconfiguration, bad use, overuse or component wear, *etc*.).

In our study, FMEA focuses on the middleware, which is the heart of a RFID system. The application of this technique on the three layers of the middleware (Application Abstraction Layer, Data Processing Layer and Hardware Abstraction Layer) is presented in Table 10 to Table 17.

The processing layer is based on modular architecture. Its functions are performed by the cooperation of a set of components. The main advantage of this type of architecture lies in the fact that the crash of a component does not break down the whole system. But this is not always true. In our case, some components of the DPL are strongly connected, and the crash of one of them triggers a chain reaction (*i.e.*, domino effect) that leads to the crash of the system.

Example: The failure of DF (Data Filtering) such as "the unnecessary data or the redundant data are not filtered" can lead to an overload of DT (Data Transformation); i.e. the data flow is blocked or the transmitted data are no longer in the correct format. So, the other components such as DA (Data Aggregation), DD (Data Dissemination) or DP (Data Protection) are not able to process the received data and the system will crash.

The failure modes that we have identified are not detailed in some cases such as "inconsistent data aggregation" that we can divide in:

- Grouping data types that should not be together or non-grouping of data types that must be together.
- No data aggregation.
- *etc*.

The system effects may differ depending on the component implementation and configuration. We can illustrate this in the effects of the "DP (Data Protection) crash". Indeed, the consequences of this failure mode are related to the enabled security policy. An example of policies is:

- "Anything not explicitly permitted is prohibited"; consequences: any data flows are blocked (System crash).
- "Anything not explicitly prohibited is permitted"; consequences: the system is still operational but there is a problem of data privacy (any user can access any data). We can see that this policy is more fault tolerant (high availability) than the first one, but it violates the security of the data. Then we can say that the security policy must be chosen carefully and according to our priorities. This introduces the notion of effects severity. If we consider the availability of this system, then the first policy is more severe than the second one, but if we take into account the security of data, the opposite is true.

The physical layer and the application layer have almost the same errors but with different consequences and severities. The failure "The action performed is different from that requested" is an example of error with severe consequences and hardly detectable. This is because the system is still operational but with transmission of data different from the requested one.

Table 10. FMEA of Application Layer

Failure modes	Possible causes	Effects on the system	Proposed Solutions
No access to MW services.	- Misconfiguration of DP. - Programming error in the AAL. - Crash of AAL.	System failure: impossible or incoherent product management.	- Reconfiguration of DP (Access rights modification). - AAL code review. - Regression testing.
Data format or application queries not recognized by AAL.	- Programming error in the AAL. - Incompatibility of the application with the middleware. - Communications link failure between MW and reader, or disturbance of transmission on the medium (external aggression, *e.g.*, interferences).	System failure: impossible or incoherent product management.	- AAL code review. - Regression testing. - Improvement of HAL layer to support application commands. - Application replacement by a compatible one. - Replacement of the communication link. - Suppression of disturbances when possible.
Slow execution / AAL overload.	- Low performance host machine. - AAL source code not optimized. - Too many readers or user applications (too many requests and huge amount of data to process). - Host machine slowed down by other processes.	System performances slowed down / System crash.	- Replacement of the host machine by another more powerful one. - Code review for optimization. - Regression testing. - Reduction of the number of applications or readers. - Reduction of the number of processes on the host machine. - Load / Stress Testing.

Table 11. FMEA of Physical Layer

Failure modes	Possible causes	Effects on the system	Proposed Solutions
HAL indicates that the received data is erroneous (but the data is correct).	- Errors in HAL code. - HAL misconfiguration (a reader driver is not loaded when needed). - Communications link failure between MW and reader or disturbance of transmission on the medium. - Unsupported hardware or protocol.	- System performances slowed down / System crash. - Loss of information.	- HAL source code review. - Regression testing. - HAL Reconfiguration.
HAL crash.	- Error in the HAL source code. - Too much data to process.	- Impossible read/write operations. - System crash. - Loss of information.	- HAL code review for errors fixing and improving of its performances. - Regression testing. - Reduction of the number of applications / readers.
HAL receives data but does not provide any output.	- Unsupported received data fror the DPL (components incompatibility). - Unrecognized communication protocol. - Erroneous translation of the DPL command for the reader.	- System crash.	- Integration and compatibility testing. - Improvement of the MW to support the communication protocol. - HAL code review. - Regression testing.
The performed action is not that requested.	- Erroneous translation of the DPL command for the reader.	- Inconsistent product management. - System crash.	- HAL code review. - Regression testing.
Slow execution / HAL overload.	- Low performance machine. - HAL source code not optimized. - Huge amount of data to process (too many applications or readers). - Host machine slowed down by other processes.	- System performances slowed down / System crash. - Loss of information.	- Replacement of the host machine by a high performance one. - HAL Code review for optimization. - Regression testing. - Reduction of the number of entities that request HAL services. - Reduction of the number of processes on the host machine.

Table 12. FMEA of Data Aggregation component

Failure modes	Possible causes	Effects on the system	Proposed Solutions
Inconsistent data aggregation.	- DA misconfiguration. - Errors in DA program.	System performances slowed down / System crash.	- DA reconfiguration. - DA code review. - Regression testing.
Data format unrecognized	- Errors in DA source code. - Incompatibility of DA and the data transmitter component (DR, DT, DF...).	System performances slowed down / System crash.	- DA code review. - Regression testing. - Integration and compatibility testing.
Slow execution / DA overload.	- Low performance machine. - DA source code not optimized. - Huge amount of data to process (too many applications or readers). - Host machine slowed down by other processes.	System performances slowed down / System crash.	- Replacement of the host machine by a high performance one. - Code review for optimization. - Regression testing. - Reduction of the number of applications or readers. - Reduction of the number of processes on the host machine. - Load / Stress Testing.

Table 13. FMEA of Data Dissemination component

Failure modes	Possible causes	Effects on the system	Proposed Solutions
Data transmission to applications other than the allowed ones.	- DD misconfiguration. - Programming error of DD module.	System failure: - Incoherent product management - Problem of data privacy.	- Reconfiguration of the DD or DP module. - DD code review. - Regression testing.
Non-reception of the needed data by an application.	- DD misconfiguration. - Errors in DD program.	System failure: - Incoherent or impossible product management.	- DD or DP module reconfiguration. - DD source code review. - Regression testing.
No data dissemination.	- DD misconfiguration. - Errors in DD program. - DD crash.	System failure: - Inconsistency (incoherence) in product management. - Real-time tracking of objects is impossible. - Loss of information.	- DD or DP reconfiguration. - DD code review. - Regression testing. - Retry the data dissemination.
Slow execution / DD overload.	- Low performance host machine. - DD source code not optimized. - Too many applications or readers (huge amount of data to process). - Host machine slowed down by other processes.	System performances slowed down / System crash.	- Replacement of the host machine by a high performance one. - Code review for optimization. - Regression testing. - Reduction of the number of applications or readers. - Reduction of the number of processes on the host machine. - Load / Stress Testing.

Table 14. FMEA of Data Filtering component

Failure modes	Possible causes	Effects on the system	Proposed Solutions
Masking of useful data.	- DF misconfiguration. - Errors in DF program.	System failure: - Incoherent product management. - Loss of information.	- DF reconfiguration. - DF code review. - Regression testing.
Non-filtering of unnecessary data.	- DF misconfiguration. - Errors in DF program.	System failure: - Incoherent product management. - System performances slowed down by the additional data management.	- DF reconfiguration. - DF code review. - Regression testing.
Blocking all data.	- DF or DP misconfiguration. - Errors in DF program.	System crash.	- DF or DP reconfiguration. - DF code review / Regression testing.
Data redundancy.	- Errors in DF program.	System failure : - Incoherent product management. - System slowed down by the additional data management.	- DF Code review. - Regression testing.
Slow execution / DF overload.	- Low performance host machine. - DF source code not optimized. - Too many applications or readers (huge amount of data). - Host machine slowed down by other processes.	System performances slowed down / System crash.	- Replacement of the host machine by a high performance one. - Code review for optimization. - Regression testing / Stress Testing. - Reduction of the number of applications or readers. - Reduction of the number of processes on the host machine.

Table 15. FMEA of DataBase Management component

Failure modes	Possible causes	Effects on the system	Proposed Solutions
The Database is inaccessible.	- Errors in DBM program. - Failing storage device. - DP misconfiguration. - Disturbance of transmission on the medium.	System failure : - Incoherent product management. - Loss of information.	- DBM code review. - Regression testing. - Replacement of the storage medium. - DB duplication to avoid such errors. - DP reconfiguration (Access rights modification).
Presence of errors in the DB (inconsistent data organization and storage).	- Errors in DBM program. - DB updating problems.	System failure : - Incoherent product management. - Loss of information.	- DBM code review. - Regression testing.
Data redundancy in the DB.	- Errors in the DBM program. - Problems in the DF module.	System failure: - Incoherent product management. - System slowed down by the additional data management.	- DBM code review. - DF code review. - Regression testing.
Data retrieved from the DB is not that expected.	- Query generated by the DBM does not match with the received command. - Data is badly stored in the DB	System failure: - Incoherent product management.	- DBM code review. - Regression testing.
DB connection failed.	- Database inaccessible. - Errors in the DBM program. - Communication link failure between the DB and the MW. - DB is overloaded.	- Incoherent product management. - System performances slowed down. - Loss of information.	- Solutions related to an inaccessible DB. - Replacement of the communication link. - Repeat the connection request after a while.
Slow execution / DBM overload.	- The answer to the query corresponds to a large amount of data. - Low performance machine. - DBM source code not optimized. - Too much data to process. - Host machine slowed down by other processes. - The interrogation of the DB is too frequent. - Low performance storage medium (R/W Speed, memory size).	- Incoherent product management. - System performances slowed down. - Loss of information.	- DBM source code optimization for handling large amounts of data. - Regression testing. - Replacement of the host machine by a high performance one. - Replacement of the storage medium. - Reduction of the number of processes on the host machine. - Load / Stress Testing.

Table 16. FMEA of Data Transformation component

Failure modes	Possible causes	Effects on the system	Proposed Solutions
DT incorrectly states that the received data seems to be erroneous.	- Error in DT source code. - Incompatibility of DT and the data transmitter component (DR, DF...). - In case where DT receives directly the captured data : ● Error in HAL code. ● External disturbances. ● Communication link failure.	- Incoherent product management. - System performances slowed down. - Loss of information.	- DT code review. - Regression testing. - Integration and compatibility testing. - HAL code review. - Interference removing when possible. - Replacement of the communication link.
No output data (DT receives data but does not provide any output).	- Errors in DT code (crash). - Unsupported received data (components incompatibility). - Unrecognized communication protocol.	- Incoherent product management. - System performances slowed down. - Loss of information.	- DT code review for errors fixing and improvement of its performances. - Regression testing. - Integration and compatibility testing.
Mismatch between the requested and applied format.	- Error in DT source code. - Problem of components compatibility.	- Incoherent product management. - System performances slowed down. - Loss of information.	- DT code review for errors fixing and improving its performances. - Regression testing. - Integration and compatibility testing.
Slow execution / DT overload.	- DT source code not optimized. - Low performance machine. - Host machine slowed down by other processes. - Huge amount of data to process (too many applications or readers).	-System performances slowed down / System crash. - Loss of information.	- DT Code review for optimization. - Regression testing. - Replacement of the host machine by a high performance one. - Reduction of the number of processes on the host machine. - Reduction of the number of applications or readers.

Table 17. FMEA of Data Protection component

Failure modes	Possible causes	Effects on the system	Proposed Solutions
No access to data and MW services by an application.	- DP misconfiguration. - Errors in DP source code.	- System performances slowed down - Loss of information.	- DP reconfiguration. - DP code review. - Regression testing.
Access control failure (an application accesses prohibited data and services).	- DP misconfiguration. - Errors in DP source code.	- Problem of data privacy.	- DP reconfiguration. - DP code review. - Regression testing.
DP crash.	- Errors in DP source code.	- System performances slowed down / System crash. - Loss of information	- DP source code review. - Regression testing.
Slow execution / DP overload.	- Low performance host machine. - DP source code not optimized. - Host machine slowed down by other processes. - Huge amount of data to process.	- System performances slowed down / System crash. - Loss of information.	- Replacement of the host machine by a high performance one. - DP Code review for optimization. - Regression testing. - Reduction of the number of processes on the host machine. - Reduction of the number of entities that request DP services. - Load / Stress Testing.

Table 18. FMEA of Data Reading component

Failure modes	Possible causes	Effects on the system	Proposed Solutions
Mismatch between the received and the expected data.	- Triggered command by DR does not match the demand of the AAL or the action performed by the reader. - Improper command interpretation / adaptation by the HAL.	- Inconsistent product management.	- DR, HAL or reader driver code review (with regression testing).
Erroneous received data.	- Error in HAL code. - Error in DR code. - Recent addition of not supported hardware. - The communication link is failing (e.g. disturbance due to the environment).	- System crash.	- DR or HAL code review. - MW code review to support the unsupported reader. - Regression testing. - Interference suppression. - Replacement of the communication link.
No data capture.	- Capture devices, DR or HAL crash. - Unsupported reader protocol. - Application queries unrecognized (new application, commands that are not implemented, etc.).	- System crash.	- DR or HAL code review for error correction and to support the new hardware components, the new application or the reader protocol. - Regression testing. - Replacement of faulty equipment or the one not supported by the MW.
Slow execution / DR overload.	- DR source code not optimized. - Low performance machine. - Host machine slowed down by other processes. - Huge amount of data to process (too many applications or readers).	- System performances slowed down / System crash.	- DR Code review for optimization with regression testing. - Replacement of the host machine by a high performance one. - Reduction of the number of processes on the host machine. - Reduction of the number of applications or readers. - Load / Stress Testing.

Table 19. FMEA of Data Writing component

Failure modes	Possible causes	Effects on the system	Proposed Solutions
Failed or impossible tag writing.	- DW tries to write data whose size exceeds the storage capacity of the tag. - Errors in DW source code. - Unrecognized data Format (request). - Improper command interpretation / adaptation by the HAL. - Too many writing requests in the same time (overload).	- Incoherent product management. - System performances slowed down. - Loss of information.	· DW code review for errors fixing and improvement of its performances (more control). · HAL code review. · Regression testing. · Support of the new hardware components, the new application or the reader protocol. · Reduction of the number of applications. · Load / Stress Testing.

Annexe D : Algorithme de diagnostic probabiliste des systèmes RFID (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.

begin
  $n = |R|;\ t = |G|;$
  $FT \coloneqq \emptyset;$ *// The set of Faulty Tags or groups of tags.*
  $FR \coloneqq \emptyset;$ *// The set of Faulty Readers.*
  $T[n] \coloneqq \{0,0,\dots,0\};$ *// T[u]: number of groups of tags read by reader u.*
  $r \coloneqq 0.99;$ *// Most failures of the faulty readers have visible effects on the system.*
  $nbFailR[n] \coloneqq \{0,0,\dots,0\};$ *// Failure counters for each reader, (array of n counters).*
  $nbFailT[t] \coloneqq \{0,0,\dots,0\};$ *// Failure counters for each group of tags, (array of t counters).*
  $failureProba[n] = \{0,0,\dots 0\};$ *// Failure probabilities of all readers.*
  $x \coloneqq 0;$ *// number of groups of tags already processed.*

  for $(g \in G)\{$
    $x{+}{+};$
    for $(u \in R_g)\ \{$ *// R_g is the set of readers that process the group g with $R_g \subset R$.*
      computeProfile(u,g); [51]
      $T[u] \coloneqq T[u] + 1;$ *// Counting the number of groups of tags processed by u.*
      if $(D_u(g) = 0)\ \{$
      // $D_u(g)$=0 means "g does not match its original profile according to the reader u".
        $nbFailT[g] \coloneqq nbFailT[g] + 1;$ *// number of times g is declared faulty.*
      }
    }
    compute(S(g)) ; *// S(g) contains the faulty components (faulty readers and/or tags).*
    for $(u \in R)\ \{$
      if $(u \in S(g))\ \{$*// u is faulty on g.*
        $nbFailR[u] \coloneqq nbFailR[u] + 1;$
        $failureProba[u] \coloneqq nbFailR[u]/T[u]\,;$
      }
    }
    if $(g \in S(g))\ \{$ *// g is a faulty group.*
      $FT \coloneqq FT \cup \{g\};$
      print("g is faulty with a probability", $nbFailT[g]/n \times I(n, x, failureProba, r)$);
    }
  }
  compute(**SF**); *// SF contains the real faulty readers.*
  for $(u \in SF)\ \{$
    $FR \coloneqq FR \cup \{u\};$
    print("u is faulty with a probability", $I(n, t, failureProba, r)$);
  }
end

[51] *computeProfile(u,g)* : est une fonction qui permet de calculer le profil du groupe de tags *g* suivant le lecteur *u*. Cette approche est présentée brièvement dans le chapitre 2 et fait l'objet d'un article [65].

Annexe E : Explication du modèle probabiliste

La fiabilité du diagnostic de diagAlgo (chapitre 3) réside dans sa capacité à identifier les lecteurs défaillants comme étant défaillants et les lecteurs non défaillants comme étant non défaillants. Nous utiliserons le terme « Identifiabilité » ou « précision du diagnostic » pour parler de la capacité de l'algorithme diagAlgo à effectuer un diagnostic correct des lecteurs.

***p** = la probabilité de défaillance d'un lecteur ; **r** = la probabilité qu'un lecteur défaillant fasse un jugement incorrect ; **t** = nombre de groupes de tags traités. n = le nombre total des lecteurs participants à l'analyse.*

Soit CN (Correct Negative) la probabilité qu'un lecteur R_1 **non défaillant** soit considéré comme tel, après le traitement de ***t*** groupes de tags.
Ce lecteur R_1 non défaillant est jugé correctement si

- Au moins la moitié du nombre des lecteurs sont **non défaillants** et dont les résultats sur les groupes de tags concordent avec ceux de R_1. Ce cas est représenté par :

$$
\begin{aligned}
&1 \quad C(n/2, n-1).(1-p)^{n/2}.p^{(n-1)-n/2} + \\
&2 \quad C(n/2+1, n-1)(1-p)^{n/2+1}.p^{(n-1)-n/2-1} + \\
&3 \quad ...+ \\
&4 \quad C(n-2, n-1)(1-p)^{n-2}. p + \\
&5 \quad C(n-1, n-1)(1-p)^{n-1}
\end{aligned}
\qquad (A)
$$

$$(A) <=> \sum_{i=\lfloor \frac{n}{2} \rfloor}^{n-1} C(i, n-1) \times (1-p)^i \times p^{n-1-i}$$

- Au moins la moitié du nombre des lecteurs sont **défaillants**, mais pour chaque groupe de tags, il y a au moins la moitié des lecteurs qui font un jugement correct sur ledit groupe et qui concorde avec celui de R_1. Ce cas est représenté par :

$$
\begin{aligned}
&1 \quad C(n/2,n-1)p^{n/2}(1-p)^{(n-1)-n/2}.(1-r^{n/2})^t + \\
&2 \quad C(n/2+1, n-1)p^{n/2+1}(1-p)^{(n-1)-n/2-1}.[(1-r^{n/2+1})-C(1, n/2+1)(1-r).r^{n/2}]^t + \\
&3 \quad C(n/2+2, n-1)p^{n/2+2}(1-p)^{(n-1)-n/2-2}[C(3,n/2+2)(1-r)^3.r^{n/2-1} + C(4,n/2+2)(1-r)^4.r^{n/2-2} + ...+ \\
&\quad\quad C(n/2+2, n/2+2)(1-r)^{n/2+2}]^t + \\
&4 \quad ...+ \\
&5 \quad C(n-1, n-1)p^{n-1}.[C(n/2,n-1)(1-r)^{n/2}.r^{n-1-n/2} + ... + C(n-2,n-1)(1-r)^{n-2}.r + C(n-1,n-1)(1-r)^{n-1}]^t
\end{aligned}
\qquad (B)
$$

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

Explication:

- Formule (A) :
  Le fragment de la formule en ligne 1 correspond au cas où la moitié des lecteurs est défaillante et l'autre moitié est non défaillante. La deuxième ligne correspond au cas où la

moitié des lecteurs plus un sont non défaillants, ainsi de suite jusqu'à avoir tous les lecteurs voisins du lecteur en cours de diagnostic non défaillants (ligne 5).

- Formule (B) :
  Le fragment de la formule en ligne 1 correspond au cas où la moitié des lecteurs est défaillante et l'autre moitié est non défaillante. Dans ce cas, il suffit qu'un lecteur défaillant fasse un jugement correct de chaque tag ou groupe de tags pour que l'ensemble des lecteurs qui ont fait un jugement correct constitue la majorité. La ligne 2 correspond au cas où la moitié des lecteurs plus un sont défaillants. Dans ce cas, il suffit qu'au moins deux lecteurs défaillants fassent un jugement correct. Ainsi de suite, jusqu'à avoir tous les voisins du lecteur en cours de diagnostic défaillants. Dans ce cas, pour chaque tag ou groupe de tags analysé, il faut qu'au moins la moitié des lecteurs voisins fassent un jugement correct du tag ou du groupe de tags analysé.

NB : La division utilisée dans les formules est une division entière.

Ce qui fait que la probabilité d'avoir un correct négatif est

$$CN(t,n,p,r) = (A) + (B)$$

Il nous reste maintenant à définir la probabilité pour qu'un lecteur défaillant soit considéré défaillant. Cela revient à s'intéresser au complémentaire de la probabilité qu'un lecteur défaillant soit considéré comme non défaillant.

Soit FN (False Negative) la probabilité qu'un lecteur **défaillant** soit considéré **non défaillant.**

- Un lecteur R_1 **défaillant** est jugé **non défaillant** si
  - R_1 traite **correctement** les r groupes de tags (il a un comportement d'un lecteur non défaillant lors du traitement de tous les r groupes de tags). Ce cas est représenté par la probabilité

$$CN(t, n, p, r) \times (1 - r)^t \quad \text{(C)}$$

$$(C) <=> CN(t,n,p,r) \times (1-r)^t$$

  - R_1 traite **incorrectement** un sous-ensemble de i groupes de tags ($1 \leq i \leq t$), mais pour **chaque** résultat sur ces i groupes, il y a **au moins la moitié** du nombre des lecteurs voisins de R_1 qui sont **défaillants** et qui ont **le même résultat** que R_1.

1 $[C(n/2, n-1).(p.r)^{n/2}.((1-p)+ p.(1-r))^{(n-1)-n/2} + C(n/2+1, n-1).(p.r)^{n/2+1}.((1-p)+p.(1-r))^{(n-1)-n/2-1} +...+(p.r)^{n-1}]^1 .$
$C(1, r).r.(1-r)^{t-1}.L(_{n,...,}t-1) +$
2 $[C(n/2, n-1).(p.r)^{n/2}.((1-p)+ p.(1-r))^{(n-1)-n/2} + C(n/2+1, n-1).(p.r)^{n/2+1}.((1-p)+p.(1-r))^{(n-1)-n/2-1} +...+(p.r)^{n-1}]^2 .$
$C(2, r).r^2.(1-r)^{t-2}.L(t-2) +$
3 ...+
4 $[C(n/2, n-1).(p.r)^{n/2}.((1-p)+ p.(1-r))^{(n-1)-n/2} + C(n/2+1, n-1).(p.r)^{n/2+1}.((1-p)+p.(1-r))^{(n-1)-n/2-1} +...+(p.r)^{n-1}]^t.r^t$

(D)

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

Explication :

CN(t-i,n,p,r) avec ($1 \leq i \leq t$) dans la formule (D), est la probabilité pour que tous les n lecteurs soient considérés comme non défaillants sur le traitement des "t-i" groupes de tags. Cela revient à dire que c'est la probabilité pour que les "t-i" groupes de tags soient analysés correctement.

La ligne 1 de la formule (D) représente la probabilité pour qu'au moins la moitié (*i.e.*, la majorité) des lecteurs soient défaillants et ces lecteurs traitent incorrectement un seul et même groupe de tags. La ligne deux concerne le traitement incorrect de deux groupes de tags par au moins la moitié des lecteurs. Ainsi de suite, jusqu'au traitement incorrect de tous les t groupes de tags par au moins la moitié des lecteurs (ligne 4).

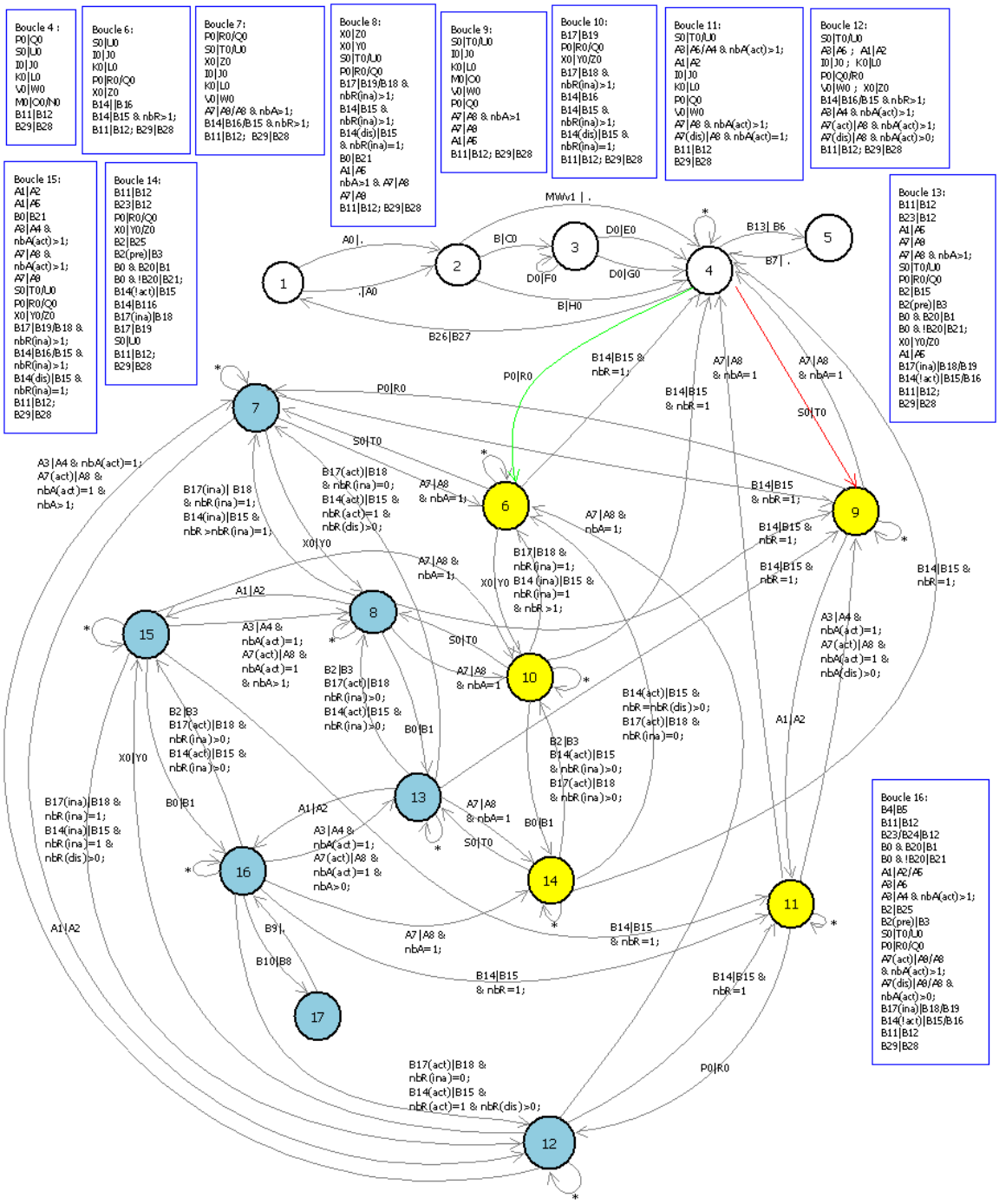
Ce qui fait que la probabilité d'avoir un faux négatif est

$$\mathbf{FN(t,n,p,r) = (C) + (D)}$$

- Soit **I(t,n,p,r)** la probabilité que les deux types de lecteurs (défaillant et non défaillant) soient correctement diagnostiqués. Alors

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

Annexe F : Machine à états finis du protocole LLRP



Le protocole LLRP se base sur des messages XML pour véhiculer les requêtes et leurs réponses entre le middleware et les lecteurs RFID. Dans cette annexe, nous représentons le comportement du protocole LLRP en 17 états et plus de 200 transitions. Les états 1 à 4 représentent la phase de négociation de la version du protocole à utiliser et les états restants représentent la phase d'échange de messages proprement dite. Les états coloriés en jaune représentent des états du protocole LLRP où seul un type de spécifications (ROSpec ou AccessSpec) est défini sur le lecteur RFID. Les états coloriés en bleus représentent les états où les deux types de spécifications (ROSpec et AccessSpec) sont définis dans le lecteur RFID.

Etat	Description
1	Lecteur non connecté
2	Lecteur connecté
3	Même état que 2 (état interne inchangé)
4	Accord sur la version du protocole nbR++ (si P0|R0) ; nbA++ (si S0|T0) ;
5	En attente des commandes du client
6	ROSpec = disabled nbR++ (si P0|R0) ; nbA -- (si A7|A8) ;
7	ROSpec = disabled; AccessSpec = disabled nbR++ (si P0|R0) ; nbA++ (si S0|T0) ; nbA -- (si A7|A8) ;
8	ROSpec = inactive AccessSpec = disabled nbR++ (si P0|R0) ; nbA++ (si S0|T0) ; nbA -- (si A7|A8) ;
9	AccessSpec = disabled nbA++ (si S0|T0) ; nbA -- (si A7|A8) ; nbR++ (si P0|R0) ;
10	ROSpec = inactive nbA ++ (si S0|T0) ; nbR -- (si B14|B15) ; nbR++(si P0|R0) ;
11	AccessSpec = active nbA++ (si S0|T0) ; nbR++ (P0|R0) ; nbA-- (si A7|A8) ;
12	ROSpec = disabled AccessSpec = active nbA++ (si S0|T0) ; nbA -- (si A7|A8) ; nbR++ (si P0|R0) ; nbR -- (si B14|B15) ;
13	Exécution de ROSpec (ROSpec=active) AccessSpec = disabled nbA++ (si S0|T0) ; nbA -- (si A7|A8) ; nbR++ (si P0|R0) ; nbR -- (si B14|B15) ;
14	Exécution de ROSpec nbR++ (si P0|R0) ; nbR -- (si B14|B15) ; nbA++ (si S0|T0) ;
15	ROSpec = inactive AccessSpec = active nbA++ (si S0|T0) ; nbA -- (si A7|A8) ; nbR++ (si P0|R0) ; nbR -- (si B14|B15) ;
16	Exécution de ROSpec (ROSpec=active) Exécution de AccessSpec nbA++ (si S0|T0) ; nbA -- (si A7|A8) ; nbR++ (si P0|R0) ; nbR -- (si B14|B15) ;
17	En attente des commandes du client

Transition	Description
A0	LLRPConnection(V1)
B	GET_SUPPORTED_VERSION(V2)
C0	GET_SUPPORTED_VERSION_RESP(liste versions)
D0	SET_PROCOL_VERSION (X∈liste versions)
E0	SET_PROCOL_VERSION_RESPONSE(M_Success)
F0	ERROR_MESSAGE(M_UnsupportedVersion)
G0	ERROR_MESSAGE(M_UnexpectedMessage) (*e.g.*, Négociation de versions déjà effectuée)
H0	ERROR_MESSAGE(M_UnsupportedVersion) (La suite de la communication s'effectuera en V1)
I0	GET_READER_CAPABILITIES
J0	GET_READER_CAPABILITIES_RESP(LLRPStatus)
K0	GET_READER_CONFIG
L0	GET_READER_CONFIG_RESP(LLRPStatus)
M0	SET_READER_CONFIG
N0	SET_READER_CONFIG_RESP(M_Success)
O0	SET_READER_CONFIG_RESP(ErrorCode)
P0	ADD_ROSpec
Q0	ADD_ROSpecResponse(ErrorCode)
R0	ADD_ROSpecResponse(M_Success)
S0	ADD_AccessSpec
T0	ADD_AccessSpecResponse(M_Success)
U0	ADD_AccessSpecResponse(ErrorCode)
V0	GET_ROSpec
W0	GET_ROSpecResponse(LLPRStatus)
X0	Enable_ROSpec(ROSpecID)
Y0	Enable_ROSpecResponse(M_Success)
Z0	Enable_ROSpecResponse(ErrorCode)
A1	Enable_AccessSpec (AccessSpecID)
A2	Enable_AccessSpecResponse(M_Success)
A3	Disable_AccessSpec(AccessSpecID)
A4	Disable_AccessSpecResponse(M_Success)
A5	Enable_AccessSpecResponse(ErrorCode)
A6	Disable_AccessSpecResponse(ErrorCode)
A7	Delete_AccessSpec(AccessSpecID) ou AccessSpecStopTrigger
A8	Delete_AccessSpecResponse(M_success)
A9	Delete_AccessSpecResponse(ErrorCode)
B0	Start_ROSpec(ROSpecID) **ou** ROSpecStartTrigger
B1	Start_ROSpecResponse(M_Success)
B2	Stop_ROSpec(ROSpecID) **ou** AllSpecDone **ou** ROSpecStopTrigger **ou** *Preempted*
B3	Stop_ROSpecResponse(M_Success)
B4	TagSpec Matched
B5	Exécution de AccessSpec
B6	KEEPALIVE
B7	KEEPALIVE_ACK
B8	CLIENT_REQUEST_OP **+** TagDataReport
B9	CLIENT_REQUEST_OP_RESPONSE
B10	ClientRequestOpSpec ∈ OpSpec de AccessSpec
B11	GET_REPORT ou B30
B12	RO_ACCESS_REPORT
B13	KeepaliveTrigger = periodic (grâce à Set_Reader_Config)
B14	Delete_ROSpec(ROSpecID)
B15	Delete_ROSpecResponse (M_Success)
B16	Delete_ROSpecResponse(ErrorCode)
B17	Disable_ ROSpec(ROSpecID)
B18	Disable_ ROSpecResponse (M_Success)
B19	Disable_ ROSpecResponse(ErrorCode)
B20	Priorité de la ROSpec demandante > à la priorité de la ROSpec en cours
B21	Start_ROSpecResponse(ErrorCode)
B22	A7 **ou** AccessStopCondition
B23	ROReportTrigger
B24	AccessReportTrigger
B25	Stop_ROSpecResponse(ErrorCode)
B26	CLOSE_CONNECTION
B27	CLOSE_CONNECTION_RESPONSE
B28	READER_EVENT_NOTIFICATION
B29	Le client est inscrit à un ou plusieurs types d'événements par Set_Reader_Config et (B30 ou HoldEventAndReportsUponReconnect = true)
B30	ENABLE_EVENTS_AND_REPORTS
MWv1	Le client (middleware) implémente juste LLRP v1 (La communication s'effectuera en V1)

Remarques :

- **nbR** = nombre de ROSpec ;
- **nbA** : nombre de AccessSpec ;
- **nbR(act)** = nombre de ROSpec dans l'état "active" ;
- **ina** = inactive ;
- **act** = active ;
- **dis** = disabled ;
- **B2(pre)** = ROSpec is preempted
- **S0|T0/U0** = soit S0|T0 ou bien S0|U0 ;
- Les transitions en boucle des états sont représentées dans leur rectangle respectif.

Résumé :

On assiste de nos jours à une utilisation croissante des systèmes RFID (Radio Frequency IDentification systems) dans divers domaines d'application (logistique, systèmes de production, inventaires, traçabilité, etc.). Certaines de ces applications présentent un caractère critique à l'image du respect de la chaîne de froid lors de l'acheminement de denrées alimentaires ou dans le cas de systèmes de manutention de bagages dans les aéroports. Or, la sensibilité des systèmes RFID vis-à-vis de leur environnement, notamment des perturbations électromagnétiques ou de la présence d'obstacles, les rend vulnérables. De même, de par le nombre important d'éléments (étiquettes, lecteurs) mis en œuvre dans de tels systèmes, des comportements erronés peuvent survenir en raison de fautes dans les divers éléments constituant le système. D'où l'importance et la nécessité de traiter le problème de la sûreté de fonctionnement et de la tolérance aux fautes dans le but de rendre ces systèmes plus robustes.

L'objectif de cette thèse concerne la proposition d'approches logicielles de test et de diagnostic en ligne adaptées aux systèmes RFID en vue d'améliorer leur robustesse. Depuis quelques années, une exploitation efficace des systèmes RFID a vu le développement d'intergiciels ou de middlewares RFID, dont le rôle est de proposer des services permettant la gestion des quantités de données importantes en provenance des lecteurs RFID. L'utilisation de tels intergiciels est d'un grand intérêt pour la sûreté de fonctionnement des systèmes RFID en raison de la nature distribuée de ces systèmes ; en particulier, grâce à l'intégration des mécanismes de sûreté de fonctionnement, plus précisément le test et le diagnostic en ligne, au niveau du middleware. Dans cette thématique, nous avons proposé plusieurs solutions pour couvrir les deux couches centrales du système à savoir la couche middleware et son interface de communication avec les sources de données, le protocole LLRP (Low Level Reader Protocol). Nous avons proposé une solution middleware compatible avec le standard de communication des systèmes RFID, et utilisée comme un réceptacle pour une solution algorithmique de diagnostic probabiliste qui permet de détecter les défaillances potentielles des composants du système sur la base d'un modèle probabiliste qui tient compte de l'environnement d'exécution. Ensuite, nous avons proposé un mécanisme d'analyse des fichiers log de l'interface de communication LLRP, complémentaire à l'algorithme probabiliste et qui permet d'approfondir le diagnostic en recherchant les causes de la défaillance détectée sur la base de différentes signatures de défaillances déjà établies. Enfin, nous avons proposé une extension du standard de communication LLRP qui tient compte de plusieurs comportements défaillants dans le but de rendre ce dernier plus fiable.

Mots-clés : Systèmes RFID, Middleware RFID, Protocole LLRP, test, diagnostic, sûreté de fonctionnement.

Abstract:

We are witnessing today a growing use of RFID (Radio Frequency IDentification) systems in various application areas (logistics, production systems, product traceability, etc.). Some of these applications are critical such as food-related cold chain logistics or baggage handling systems in airports. Nevertheless, RFID are very sensitive to their environment, including electromagnetic disturbances or presence of obstacles, making them error-prone. Also, because of the large number of elements (tags, readers, and sensors) constituting current RFID systems, erroneous behaviors are more frequent. Hence, it is important to address all the problems related to RFID system dependability and deal with them in order to make these systems more robust.

The goal of this thesis is the development of software test and online diagnosis facilities for RFID systems to improve their robustness. In recent years, the effective use of RFID systems has seen the development of RFID middleware solutions, whose role is to provide services for the management of large amounts of raw data of the various RFID sources. Due to the distributed nature of current RFID systems, the use of such solutions is of great interest regarding the improvement of RFID system dependability. In particular, thanks to the integration of dependability mechanisms, specifically the online test and diagnosis approaches in the RFID middleware solution. In addition, because of the middleware is considered as the backbone of an RFID system, whereby the whole RFID dataflow passes; all the needed information will be availabe to our proposed approaches to perform a correct diagnosis. We proposed several solutions to cover the two main layers of RFID systems; namely, the middleware layer and the communication layer between the middleware and the data sources, the Low Level Reader Protocol (LLRP). We have proposed a LLRP compliant middleware solution, used to accommodate a probabilistic diagnosis algorithm to detect potential failures of the RFID system components on the basis of a probabilistic model that takes into account the execution conditions. Then, we proposed a complementary mechanism to the previous algorithm for analyzing the log files of the LLRP communication interface allowing further analysis by looking for the causes of the detected failures on the basis of an already defined set of failure signatures. Finally, we proposed an extension of the LLRP standard to make it more reliable by taking into account several RFID failures.

Keywords: RFID systems, RFID middleware, LLRP protocol, test, diagnosis, dependability.