Simulation de fautes pour l'évaluation du test en ligne de systèmes RFID

Gilles Fritz

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Nano Electronique Nano Technologies**

Arrêté ministériel : 7 août 2006

Présentée par

« Gilles FRITZ »

Thèse dirigée par **« Vincent Beroulle »**
coencadrée par **« Oum-El-Kheir AKTOUF »**et **« David HELY »**

préparée au sein du **Laboratoire de Conception et d'intégration des Systèmes**
dans **l'École Doctorale Electronique, Electrotechnique, Automatique et Traitement du Signal**

Simulation de fautes pour l'évaluation du test en ligne de systèmes RFID

Thèse soutenue publiquement le **10 décembre 2012**,
devant le jury composé de :

M. Olivier ROMAIN
Professeur à l'université de Cergy-Pontoise, Président
M. Jean ARLAT
Directeur de recherche au LAAS-CNRS de Toulouse, Rapporteur
M. Thierry VAL
Professeur à l'université de Toulouse, Rapporteur
Mme. Nathalie MITTON
Chargée de recherche à l'INRIA de Lille, Examinateur
M. Ioannis PARISSIS
Professeur à Grenoble INP, Examinateur
M. Vincent BEROULLE
Maitre de conférences à Grenoble INP, Directeur de thèse
Mme. Oum-El-Kheir AKTOUF
Maitre de conférences à Grenoble INP, Co-encadrant
M. David HÉLY
Maitre de conférences à Grenoble INP, Co-encadrant

Remerciements

Je tenais à remercier en premier lieu les peronnes m'ayant proposer cette aventure qu'est la thèse : M. Vincent Beroulle, Mme Oum-El-Kheir Aktouf et M. David Hély. Je les remercie vivement de leur accompagnement tout au long de cette thèse.

Je suis très reconnaissant à M. Jean Arlat, directeur de recherche du LAAS-CNRS de Toulouse, et M. Thierry Val, Professeur à l'université de Toulouse, d'avoir accepté le rôle de rapporteur.

Je remercie aussi M. Olivier Romain, Professeur à l'université de Cergy-Pontoise, pour avoir accepté de présider le jury de thèse de doctorat et Mme Nathalie Mitton, chargée de recherche à l'INRIA de Lille, pour avoir accepté d'examnier mon travail.

Je remercie également M. Ioannis Parissis, Professeur à Grenoble-INP, pour m'avoir intégré au groupe de recherche ainsi qu'à l'équipe pédagogique et lui suis reconnaissant d'avoir accepté de participé au jury de thèse.

Je remercie M. Christophe Chantepy, responsable du RFTLab, pour sa collaboration aux expérimentations et son expertises. Je remercie également M. Christophe Medina, du RFTLab, pour le temps qu'il nous a consacré tout au long de ces 3 années.

Je n'oublie pas bien sûr tout le personnel du LCIS qui m'a accompagné pendant ma thèse, ni tout le personnel de l'ESISAR qui m'a accompagné durant mes études d'ingénieur.

Je remercie évidemment toute ma famille pour son irremplaçable et inconditionnel soutien tout au long de ces interminables années d'étude.

Enfin, je remercie Marie, ma femme, pour son soutien quotidien et son incroyable patience. Merci d'avoir écouté mes doutes, calmé mes inquiétudes et de m'avoir permis de réaliser cette thèse dans des conditions exceptionnelles. Et merci pour le petit bonheur que nous avons eu pendant cette thèse et celui qui arrive.

Table des matières

Table des figures

Table des tableaux

I Introduction

L'identification des personnes et des objets est une question primordiale depuis toujours. Elle permet de contrôler les entrées et sorties des zones à accès restreint ou encore de suivre la vie d'un produit, de la production jusqu'à la vente voire jusqu'au recyclage de ce dernier. De nombreuses technologies permettant cette identification de manière plus ou moins automatisée existent. Une des technologies les plus connues est le code à barres. Le code à barres permet principalement l'identification des objets, grâce à un code imprimé sur ces derniers. Ce code est alors scanné pour en extraire l'identifiant de l'objet. Cette technologie, peu chère et largement utilisée, présente néanmoins quelques inconvénients : il est impossible de récupérer l'identifiant de l'objet si le code barre est masqué ou encore il est difficile d'identifier un objet unique en raison du nombre limité d'identifiants disponibles. Différentes solutions ont été proposées pour permettre l'identification unique des objets. Celle qui nous intéresse est la RFID, pour « *Radio FrequencyIDentification* » ou Identification par Radiofréquence. Cette technologie en plein essor permet l'identification sans contact ni vision directe grâce à une étiquette intelligente, ou tag, associée à l'objet à identifier. L'identifiant contenu dans l'étiquette est extrait par un lecteur puis transféré à un logiciel spécifique, le middleware, facilitant la gestion de l'ensemble des identifiants reçus.

La RFID promet de nombreuses améliorations dans toutes les applications nécessitant l'identification et le suivi de personnes ou d'objets. L'automatisation de l'identification, la possibilité d'obtenir les informations relatives à un objet sans contact ni vision directe et la possibilité d'obtenir ces informations pour plusieurs objets simultanément permettent un gain de temps et d'argent important ainsi qu'un gain au niveau de la sécurité. Ces gains sont relativement importants notamment pour les chaînes logistiques.

Depuis son apparition durant la seconde guerre mondiale, la RFID a connu une croissance explosive. Le marché global de la RFID – étiquette, lecteur et middleware – devrait augmenter rapidement, passant d'un chiffre d'affaire de 5,56 milliards de dollars en 2006 à un chiffre d'affaire de 26 milliards de dollars en 2016 [DAS09]. Elle est utilisée dans de nombreuses applications de notre quotidien : du passeport électronique à l'identification des animaux domestiques, en passant par les forfaits de ski ou encore les tickets de transport. On retrouve historiquement la RFID dans les applications de contrôle d'accès : les badges d'accès qui, en stockant les informations relatives au porteur, lui permettent d'entrer dans les zones sécurisées ; les télépéages, les cartes de transport ou encore les forfaits de ski permettant de vérifier le bon acquittement des autorisations par les utilisateurs ; et finalement les billets d'évènements sportifs ou culturels.

On retrouve aussi la RFID dans des applications médicales [ZDN05] : des hôpitaux marseillais ont ainsi utilisé la RFID pour marquer les tubes contenant des prélèvements humains afin d'assurer un suivi optimal. D'autres utilisations de la RFID sont en cours d'étude : pour le suivi des dons de sang, pour le suivi des patients afin de prévenir les embouteillages au niveau des équipements d'examens ou encore afin de prévenir les enlèvements de nourrissons dans les maternités.

Une autre utilisation importante de la RFID est le suivi des objets durant leur cycle de vie. Un exemple connu est le cas de Mark & Spencer [GUI06]. Mark & Spencer a commencé en 2003 à équiper ses magasins d'un système de gestion par RFID. En 2007, la totalité des 120 magasins ont été équipés de ce système, gérant alors 120 millions d'articles avec 320 millions de tags. Les tags sont attachés à chaque produit au centre de distribution. Ainsi, lorsqu'ils arrivent au magasin, ils peuvent être contrôlés automatiquement et le stock est ainsi mis à jour automatiquement. Les erreurs de livraison sont ainsi repérées plus facilement. De plus,

l'inventaire exact peut être réalisé facilement chaque semaine. Ainsi, l'utilisation de la RFID dans ce contexte permet la diminution des coûts liés à la gestion des stocks et des inventaires ainsi que l'augmentation des gains en facilitant la gestion du stock, permettant par exemple de s'assurer de la disponibilité des produits en rayon, et de détecter la perte ou le vol d'article. Un autre exemple est l'airbus A380 : il est équipé de plus de 10 000 étiquettes RFID qui permettent d'accélérer la maintenance et d'améliorer la sécurité de l'appareil. Ainsi, il est possible de s'assurer de la présence des éléments de sécurité tels que les gilets de sauvetages ou encore d'assurer le suivi des pièces mécaniques critiques : date d'installation, de dernière maintenance, … La société Systerel, au travers du projet BARFID, utilise la RFID pour réaliser des barrières ferroviaires permettant de vérifier la présence de trains, d'optimiser la gestion des équipements, d'en faciliter la maintenance et d'améliorer la sécurité. Il est aussi envisagé d'utiliser des tags RFID associés à des capteurs au sein de tableaux électriques. L'objectif est alors de vérifier le bon fonctionnement des équipements de sécurité afin d'en accélérer la maintenance et de garantir un niveau de sécurité optimal.

Néanmoins, en cas de pannes du système RFID, les conséquences peuvent être importantes : pour les applications critiques telles que les applications médicales, nucléaires, ferroviaires ou aéronautiques, les incidents peuvent avoir des répercutions catastrophiques en termes de vie humaine ; pour les applications logistiques, l'apparition de défaillances implique une intervention humaine dans la chaîne automatisée. Or, l'automatisation de la chaîne introduisant une diminution de la présence d'opérateur, le coût d'une intervention devient alors important et le bénéfice lié à l'utilisation de la RFID est alors perdu. Il est alors important d'étudier la sûreté de fonctionnement des systèmes RFID, afin de connaître la confiance que l'on peut avoir en ces systèmes, ainsi que les méthodes permettant d'améliorer cette confiance. Or cette étude est particulièrement délicate car les systèmes RFID sont des systèmes fortement hétérogènes et complexes.

Cette thèse s'intéresse à l'étude de la sûreté de fonctionnement des systèmes RFID, principalement au niveau d'un unique point de lecture (lecteur), en vue de l'amélioration de sa fiabilité et de sa disponibilité. Pour cela un simulateur de fautes spécifique aux systèmes RFID est proposé afin de permettre l'analyse précise du fonctionnement du système, l'évaluation des attributs de la sûreté de fonctionnement et l'évaluation des méthodes permettant d'améliorer la robustesse du système. Cette thèse s'inscrit dans un projet de l'Agence Nationale de la Recherche appelé **SAFERFID** dont le but est l'amélioration les différents attributs de la sûreté de fonctionnement des systèmes RFID, y compris la sécurité des données, à tous les niveaux : de l'étiquette intelligente jusqu'au système d'information gérant la totalité de l'infrastructure [SafeRFID].

Le chapitre II de ce mémoire a pour objectif de présenter le fonctionnement des différents éléments de la RFID, ainsi que leurs interactions. Le principe de la sûreté de fonctionnement et les méthodes permettant d'assurer le niveau de sûreté souhaité sont présentés ainsi que leur application aux systèmes RFID. Finalement, ce chapitre présente les simulateurs RFID disponibles, permettant l'observation du système, l'évaluation de son niveau de sûreté et l'évaluation de la qualité des méthodes de test.

Le chapitre III présente le simulateur de fautes développé au cours de la thèse. Ce simulateur s'appuie sur une étude préalable des défaillances pouvant apparaître au sein du système. Cette étude a été réalisée sous forme d'une AMDE, ou Analyse des Modes de Défaillance et de leurs Effets. Le simulateur développé permet ainsi la simulation de systèmes RFID sains et fautifs afin d'optimiser les paramètres du système et d'observer l'impact des fautes sur celui-ci.

Le chapitre IV présente une méthode de test en ligne permettant de détecter la présence de dysfonctionnements apparaissant dans le système. Cette méthode de test en ligne non intrusive est comparée à d'autres méthodes de test en ligne classiques, par expérimentation ainsi que par simulation.

Finalement, en conclusion, une synthèse du travail présenté dans ce mémoire est faite et les perspectives pour ce travail et pour le projet **SAFERFID** global sont présentées.

II Etat de l'art

La RFID est une technologie en plein essor. Cette technologie, de plus en plus utilisée, permet l'identification sans contact ni vision directe des personnes ou objets. Pour cela, elle s'appuie sur des ondes radiofréquences pour échanger des données entre étiquettes intelligentes et lecteurs. Une fois les données, associées à un objet ou à une personne, récupérées par le lecteur, elles sont envoyées vers le système d'information (middleware) chargé de les gérer.

La première partie de ce chapitre présente les concepts généraux de la RFID : comment la RFID fonctionne-t-elle ? De nombreux éléments de natures différentes – matérielles, logicielles – interviennent dans le fonctionnement des systèmes RFID : quels sont-ils ? Quels sont les spécificités permettant à ce type de système l'identification des objets à moindre coût ?

La seconde partie de ce chapitre s'intéresse à la sûreté de fonctionnement, et particulièrement à celle de la RFID. L'objectif est l'étude de la robustesse des systèmes RFID : quelles sont les défaillances matérielles ou logicielles pouvant apparaître dans un système RFID ? Quelles sont les méthodes permettant de prévenir ou de détecter ces défaillances ?

Finalement, dans la dernière partie de ce chapitre, les simulateurs de systèmes RFID sont présentés. Ces simulateurs ont pour objectifs l'analyse des paramètres des systèmes RFID et l'évaluation de l'impact de ces paramètres en fonction des systèmes.

II.1 Qu'est-ce que la RFID ?

La RFID est une technologie permettant l'identification de personnes ou d'objets sans contact ni vision directe grâce aux radiofréquences [KRI07].

La RFID existe depuis longtemps [AFS09]. Elle fut notamment utilisée durant la seconde guerre mondiale par les américains, durant les années 40, pour la reconnaissance à distance des avions amis ou ennemis. Le premier brevet est apparu en 1969 et décrivait l'utilisation de la RFID pour l'identification des locomotives. Simultanément, cette technologie est utilisée pour garantir la sécurité de zones critiques, telles que les sites nucléaires, en contrôlant les accès. Dix ans plus tard, dans les années 80, l'utilisation de la RFID devient un peu plus intense : elle est notamment utilisée en Europe pour l'identification du bétail ainsi que sur les chaînes de fabrication des constructeurs automobiles. De nombreuses recherches commencent alors permettant, dans les années 90, la miniaturisation du système RFID, diminuant son coût et favorisant sa diffusion. Suite aux événements du 11 septembre 2001 aux Etats-Unis, le passeport électronique, utilisant la RFID, est utilisé à travers le monde. Les recherches continuent et, en 2003, grâce aux différentes avancées technologiques, le coût des étiquettes intelligentes est divisé par 3. Divers grandes chaînes de distribution, comme Carrefour, Auchan ou Wal-Mart, utilisent alors la RFID pour gérer la traçabilité de leurs produits. Devant l'utilisation de plus en plus massive de la RFID, et à des fins d'interopérabilité, la société GS1 – EPCglobal, leader mondial des codes à barres, décide dès lors d'utiliser la RFID pour la gestion des chaînes logistiques.

Elle s'articule autour de trois composants principaux :

1. Le *tag* – ou *étiquette*, *transpondeur*, *smart label*, … – qui est l'équipement associé à l'objet ou la personne à identifier. Il stocke les données relatives à cet objet ;

2. Le *lecteur*– ou *station de base, interrogateur*, … – qui est l'équipement capable de venir interroger le tag à distance et de récupérer (ou d'y inscrire) les données relatives à l'objet tracé ;
3. et finalement le *middleware* – ou *intergiciel* – qui est le logiciel chargé de gérer, de collecter et de mettre en forme les données pour l'application finale, application métier dédiée à l'entreprise. Il sera décrit au paragraphe II.1.4.

La Figure II-1 illustre ces différents composants et leurs interconnexions. Plusieurs lecteurs interconnectés via le réseau informatique d'une entreprise lisent les données présentes sur les tags associées à différents objets. Le middleware est en charge de gérer l'infrastructure RFID ainsi que les données issues des lecteurs, afin de fournir à l'application métier les données dont elle a besoin.

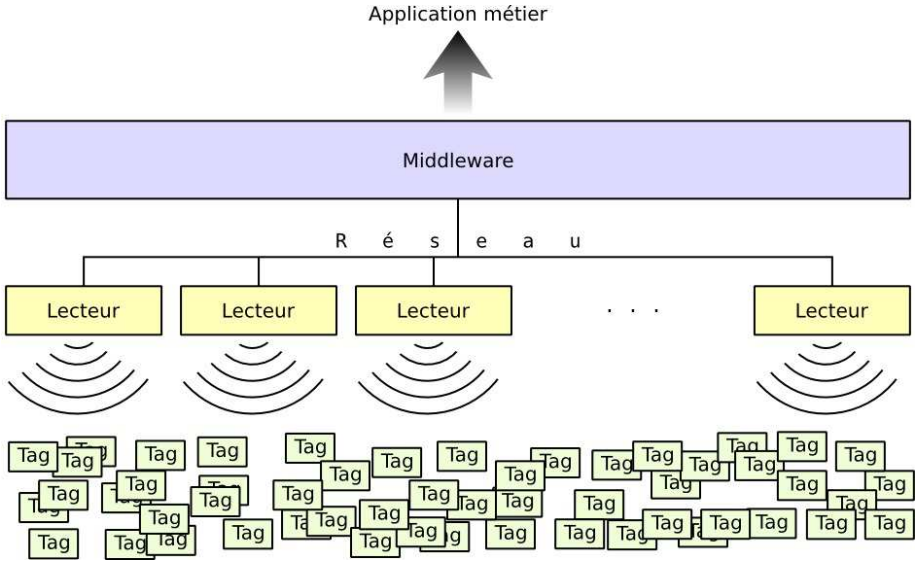


Figure II-1 : Vue globale d'un système RFID

Il existe plusieurs manières de classifier les systèmes RFID. La plus répandue et communément admise est la classification en fonction de la fréquence de l'onde utilisée pour les communications lecteurs/tags. En effet, cette fréquence est directement liée aux phénomènes physiques mis en jeu pour réaliser la communication [PAR08]. Le Tableau II-I donne les différentes fréquences utilisées en RFID.

Tableau II-I : Fréquences retenues et/ou autorisées en RFID

Ondes radiofréquences			Fréquences retenues et/ou autorisées en RFID
De 30 à 300 kHz	LF	Basses fréquences	< à 135 kHz
De 3 à 30 MHz	HF	Hautes fréquences	13,56 MHz
De 300 à 3 000 MHz	UHF	Ultra-hautes fréquences	433 MHz & de 860 à 960 MHz & 2,45GHz
De 3 à 30 GHz	SHF	Super-hautes fréquences	5,8GHz

La fréquence utilisée par un système RFID va directement impacter les distances de lectures possibles [FIN99]. Ainsi, les basses fréquences – LF – et les hautes fréquences – HF – seront utilisées principalement pour des applications de courte distance – <1cm –, de proximité –quelques centimètres – et de voisinage – quelques dizaines de centimètre – alors que les ultra-hautes fréquences – UHF – et les super-hautes fréquences seront utilisées principalement pour des applications de longue distance – quelques mètres – voire de très longue distance – plusieurs mètres, comme le montre la Figure II-2.

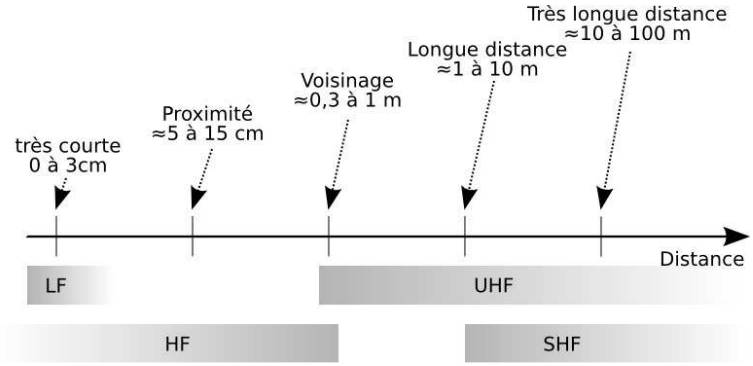


Figure II-2 : Distances de fonctionnement des systèmes RFID

Dans ce manuscrit, nous nous intéressons particulièrement à une technologie : la technologie UHF. En effet, les utilisations actuelles de la RFID qui nécessitent l'étude de la sûreté de fonctionnement des systèmes sont principalement dans ce domaine de fréquences. Les applications utilisant ces fréquences sont soit des applications critiques liées au nucléaire, au ferroviaire ou encore à l'aéronautique ; soit des applications de logistique, mettant en œuvre une grande quantité de tags et dont le but est de minimiser les interventions humaines. Une défaillance dans le système RFID d'une application logistique va bloquer une partie des produits et donc l'activité et nécessiter une intervention humaine coûteuse en temps et en argent.

Dans la technologie UHF, se distinguent plusieurs catégories de tags [PAR08, PRE08]. En effet, il est possible de classer les tags en fonction de plusieurs critères qu'illustre la Figure II-3 : alimentation, communication, lecture seule ou lecture et écriture[1]. L'objectif du travail présenté ici est d'étudier la sûreté de fonctionnement des systèmes RFID les plus couramment utilisés (systèmes dont les caractéristiques des tags sont entourées en pointillés rouge dans la Figure II-3). Nous nous intéressons donc dans la suite aux tags UHF passifs télé-alimentés, permettant des écritures et lectures multiples grâce à une mémoire EEPROM. Ces tags ne possèdent pas leur propre source d'énergie : c'est le lecteur qui envoie l'énergie sous forme d'onde électromagnétique. L'énergie disponible étant faible, les tags n'embarquent pas de système de communication, trop gourmand en énergie : la communication est alors dite passive. Ces deux caractéristiques sont décrites plus précisément dans la suite du document. Ces tags ont une mémoire contenant principalement un identifiant spécifique à l'objet suivi. Cette mémoire est en « Écriture et lecture multiple ».

La particularité principale des systèmes RFID réside dans l'utilisation de l'onde transmise par le lecteur : les tags reçoivent de l'énergie de la part du lecteur pour fonctionner et utilisent l'onde reçue du lecteur pour communiquer. Les paragraphes suivant expliquent en détail le fonctionnement des systèmes RFID du point de vue de la télé-alimentation et de la communication spécifique des systèmes RFID, dans laquelle un des équipements – le tag – ne possède pas d'émetteur à proprement parler.

II.1.1 Principe physique de fonctionnement entre tag et lecteur

Dans le cas de la RFID UHF, les systèmes fonctionnent généralement en champ lointain car les distances entre tags et lecteurs sont importantes (de l'ordre du mètre). Le couplage entre le tag et le lecteur est radiatif. La technique principalement utilisée en champ lointain est la technique dite de *rétro-modulation*, ou *backscattering* en anglais : le tag va réfléchir l'onde émise par le lecteur comme le montre la Figure II-4.

[1] La classification des tags est détaillée en annexe A.

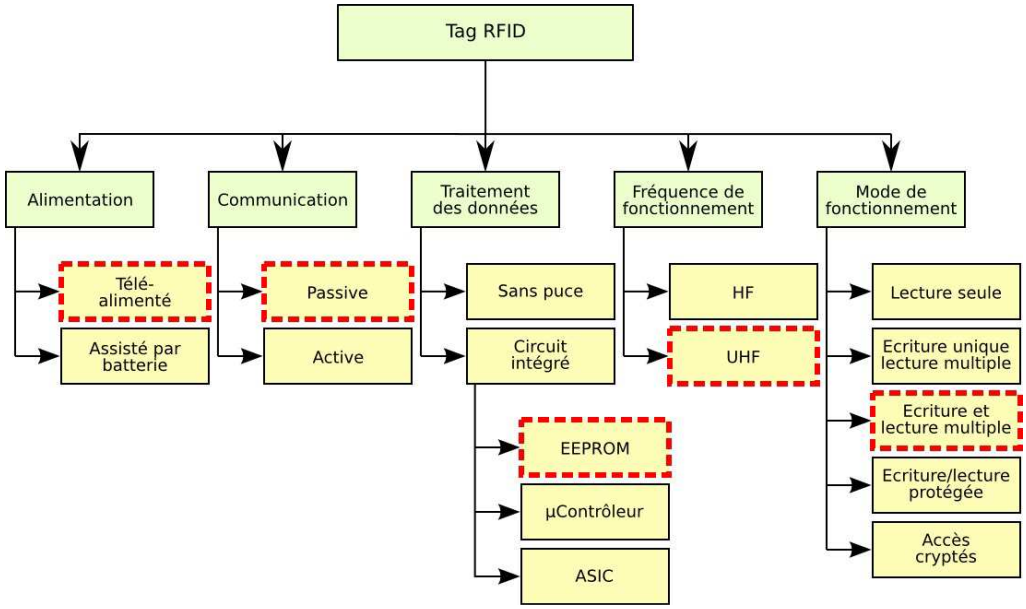


Figure II-3 : Classification des tags RFID

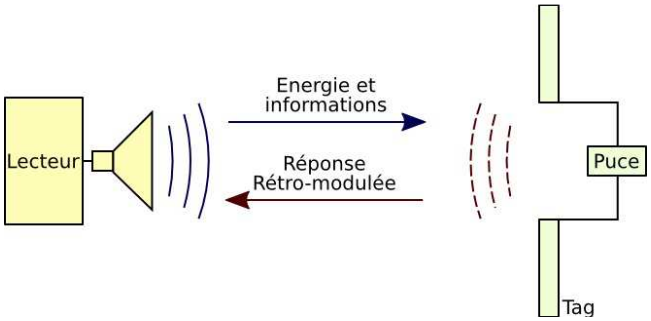


Figure II-4 : Technique de rétro-modulation ou « backscattering »

Afin de se faire comprendre par le lecteur, le tag modifie la quantité d'énergie réfléchie en faisant varier la charge aux bornes de son antenne. La Figure II-5 illustre la rétro-modulation pour 3 charges différentes. En fonctionnement normal (a), l'impédance de la puce et de l'antenne sont adaptées. Ainsi, une partie de l'onde absorbée par l'antenne est transmise à la puce et une autre partie est réfléchie. Cette réflexion est minimale, car c'est dans cette configuration que l'alimentation de la puce est optimale. Dans le cas d'un circuit ouvert (b), il n'y a pas de charge connectée à l'antenne : c'est un circuit ouvert. Dans ce cas-là, il n'y a pas de puissance transmise à la puce qui est déconnectée, et l'onde réfléchie est négligeable dans la majorité des cas. Dans le cas d'un circuit fermé (c), l'antenne est court-circuitée. Ici, la puissance réfléchie est très importante mais aucune puissance n'est transmise à la puce.

Nous venons de voir comment, malgré l'absence d'émetteur, le tag est capable de transmettre un signal au lecteur. Le paragraphe suivant explicite les techniques utilisées en RFID afin de coder l'information.

II.1.2 Transmission de l'information

L'information échangée par les différents équipements des systèmes RFID est représentée par un signal numérique (codage bit) ; signal numérique qui est modulé pour être transmis sous forme analogique.

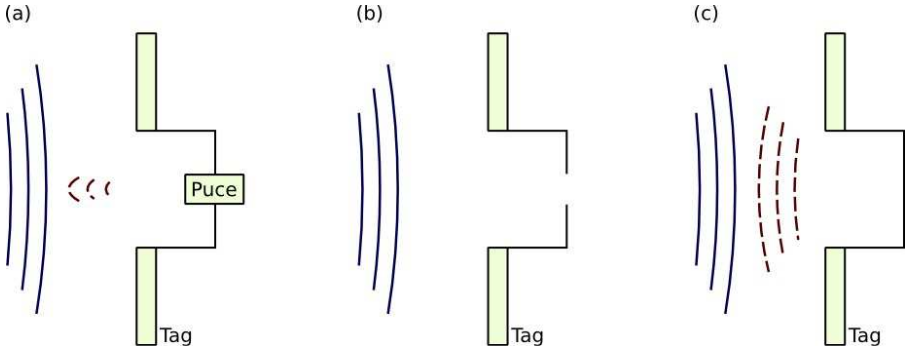


Figure II-5: Modulation de la puissance réfléchie
(a)charge adaptée, (b) circuit ouvert et (c) court-circuit

II.1.2.1 **Aspect numérique (codage bit)**

Les données binaires sont dans un premier temps représentées par un signal physique numérique. Il existe pour cela plusieurs *codages bit* possibles. Le codage bit revêt une importance cruciale dans les systèmes sans contact. En effet, le choix du codage bit influe sur la qualité du transfert d'énergie ainsi que sur la récupération des données. Ainsi, le choix du codage bit est fait en fonction des besoins de la communication, besoins qui sont différents en fonction du sens de communication – montante, du lecteur au tag ou descendante du tag au lecteur.

La liaison montante doit garantir une bonne alimentation du tag : le signal physique contenant les données doit être le plus longtemps possible à l'état haut afin d'assurer une bonne fourniture d'énergie pour la télé-alimentation. De plus, il faut un maximum de transitions permettant ainsi au tag de se synchroniser pour décoder les informations.

Pour la liaison descendante, il faut aussi garantir un maximum de transitions, ce qui permettra au lecteur d'identifier et d'extraire facilement les données même en présence de bruit important. Il faut aussi que ce codage bit minimise la consommation du tag pendant sa phase de réponse. En effet, pendant la phase de réponse, on a vu que la rétro-modulation diminue la quantité d'énergie transmise à la puce. Il faut donc que le codage bit consomme un minimum d'énergie et dure le moins longtemps possible. Le codage bit doit aussi permettre de détecter la transmission d'information par plusieurs tags simultanément, engendrant une collision. Une section spéciale est dédiée à la détection et la gestion des collisions – voir Section II.1.3.

Par exemple, pour les communications descendantes, le codage *FM0* peut être utilisé. Ce codage possède une transition en début de symbole quel que soit le bit codé. Il possède en plus une transition en milieu de symbole pour le bit « **0** ». Ainsi, chaque bit a deux représentations possibles, dépendant du symbole précédent. La même séquence a alors deux alternatives possibles, dépendant de la séquence précédente comme le montre la Figure II-6. Ce codage bit offre beaucoup de transitions, facilitant la synchronisation entre le tag et le lecteur. Néanmoins, il y a environ 50% d'états hauts permettant l'alimentation du tag, ce qui est assez faible. Il est donc principalement utilisé pour la liaison descendante, du tag vers le lecteur.

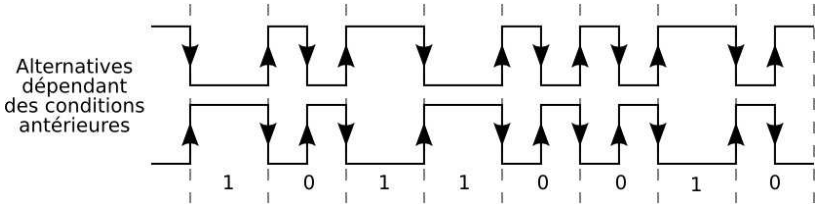


Figure II-6 : Codage bit « FM0 »

D'autres codages bit, leurs caractéristiques et leur utilisation dans le domaine de la RFID sont présentés en annexe B, notamment des codages bits permettant une transmission optimale de la puissance du lecteur vers le tag.

II.1.2.2 **Aspect analogique (modulation)**

Après avoir représenté les bits, il faut les transmettre à travers le canal, c'est-à-dire l'air. Pour cela, on utilise une onde appelée porteuse qui est modulée par le signal binaire. A la réception du signal modulé, le récepteur démodule ce qu'il a reçu pour retrouver le signal binaire initial.

La modulation la plus utilisée en RFID est la modulation d'amplitude, noté *ASK*, pour « Amplitude Shift Keying ». Elle consiste à modifier l'amplitude du signal en fonction de la représentation binaire – voir Figure II-7. Cette modulation est relativement simple à démoduler, nécessitant peu d'électronique. C'est pour cette raison qu'elle est particulièrement utilisée en RFID.

L'autre modulation utilisée en RFID est la modulation *PSK*, pour « Phase Shift Keying ». Elle consiste à modifier la phase du signal comme le montre la Figure II-7. Elle est notamment utilisée dans la communication du tag vers le lecteur car elle est simple à mettre en œuvre.

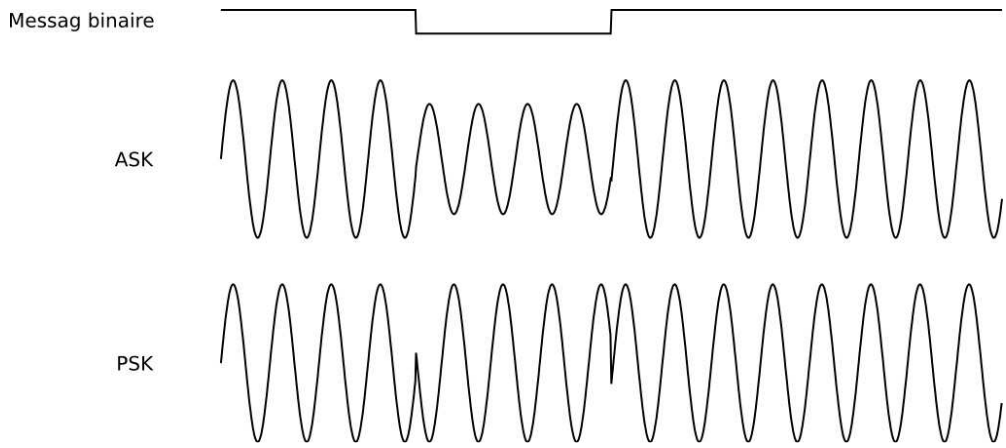


Figure II-7 : Exemples de modulations utilisées par les systèmes RFID

II.1.3 Algorithmes anticollisions

Lorsque plusieurs équipements souhaitent envoyer un message sur le même canal, les différents signaux se mélangent et en résulte un signal incorrect. C'est ce que l'on appelle une collision. Les paragraphes suivants expliquent l'origine de ces collisions, la manière de détecter la présence d'une collision et enfin les algorithmes anticollisions permettant de limiter la présence de collisions durant les différents échanges entre tous les équipements RFID.

II.1.3.1 **Origine et détection des collisions**

Les collisions sont provoquées par l'émission simultanée de différents signaux provenant de différents équipements. Il est important de comprendre le phénomène de collisions, omniprésent dans les systèmes communiquant par radiofréquence. Il est aussi important pour le système de détecter la présence de ces collisions. En effet, en fonction des paramètres de communication utilisés, il arrive que des collisions soient invisibles ; le système prend alors en compte des données erronées. Les paragraphes suivants ont pour objectifs d'expliquer

l'origine des collisions et les différentes techniques existantes permettant au système de détecter la présence de collisions.

Origine des collisions

Dans les systèmes RFID UHF, plusieurs tags sont souvent présents dans le champ du lecteur. On peut ainsi différencier deux formes de communication [FIN03].

La première forme consiste en l'envoi de données du lecteur vers les tags (Figure II-8). Les données sont reçues simultanément par tous les tags ; ce mode est aussi connu comme mode *broadcast*.

La seconde forme consiste en l'envoi de données de plusieurs tags vers un lecteur (Figure II-9). Ce mode de communication est appelé mode *accès multiple*. Le canal de communication doit alors être partagé par tous les tags afin qu'ils puissent transmettre leurs informations au lecteur sans interférence mutuelle ou *collision*.

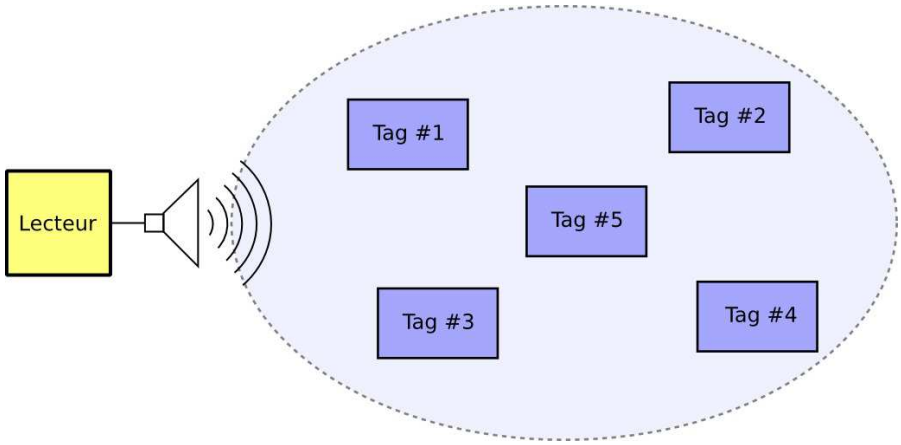


Figure II-8 : Mode broadcast. Les données envoyées par le lecteur sont à destination de tous les tags

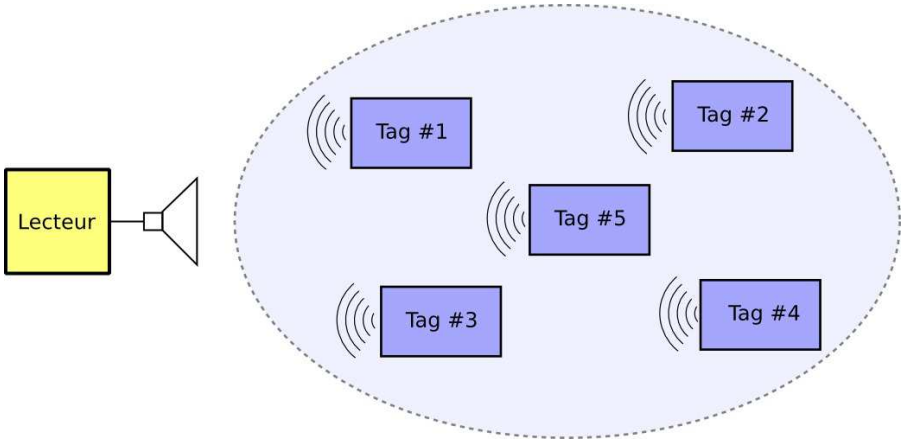


Figure II-9 : Mode multi accès : plusieurs tags envoient leurs informations au lecteur

Ce problème d'accès multiple, engendrant la communication de plusieurs équipements et donc la collision au niveau bit des messages, est connu depuis bien longtemps dans l'utilisation de la technologie RF. De nombreuses solutions ont d'ores et déjà été proposées afin de permettre à chaque participant de transmettre les données qu'il a à envoyer. Ces méthodes ont été proposées pour répondre à la problématique de systèmes RF tels que les systèmes de communication satellitaire et les réseaux de téléphone mobile. Ces systèmes ont la particularité d'avoir un flux constant de données entre les récepteurs et les émetteurs. Or,

les systèmes RFID n'ont pas cette particularité. En effet, les communications entre les tags et le lecteur sont courtes et les périodes d'attente peuvent être assez longues. Par exemple, prenons le cas du suivi d'un objet : le lecteur va récupérer l'identifiant de l'objet stocké dans le tag ainsi que diverses autres informations, puis va y ajouter ses informations. S'ensuit alors pour le lecteur une longue attente d'un autre objet. Ici, la nécessité de l'accès multiple n'est pas flagrante. Imaginons toujours le même objet à identifier. Cette fois-ci, il est stocké sur une palette ou dans un chariot Plusieurs tags sont alors dans le champ d'interrogation de la borne d'identification et une procédure de sélection du bon tag doit être mise en œuvre. Une fois le tag identifié, le système se retrouve alors dans le cas d'un seul tag à gérer. Le partage du canal ne doit donc avoir lieu que pendant cette phase de sélection. Les collisions possibles ne représentent alors qu'une faible partie des communications.

Les techniques de gestion des accès multiples pour la RFID posent différents problèmes : ils doivent permettre l'échange de données entre le tag et le lecteur en évitant les collisions, mais aussi ne pas augmenter de façon irraisonnable les délais de détection. Ces techniques d'accès multiple pour la RFID sont appelées *protocole anticollision*.

Détection des collisions

Il est important de détecter la présence de collision. En effet, détecter la présence de collision va permettre d'activer les protocoles anticollisions, mais aussi donner des informations à ces derniers afin que leur exécution soit optimisée [FIN03, PAR01, PAR08].

La détection de collision est directement dépendante du codage bit utilisé.

La Figure II-10 montre la détection de collision avec le codage bit Manchester – les **1** sont codés par une transition descendante et les **0** par une transition montante en milieu de symbole. La durée d'un bit, pout la norme UHF EPC Classe 1 Génération 2 [EPCC1G2], qui sera décrite et utilisée par la suite, varie de 1,5 µs à 200µs. Dans un souci de clarté, les données envoyées par les tags 1 et 2 sont considérés ici synchrones.

Deux tags émettent un message simultanément dont les 4^{e} et 5^{e} bits sont différents. On peut ainsi voir que la composition d'un signal représentant un **1** et un signal représentant un **0** donne un état haut durant toute la durée du bit. Ce symbole n'a aucune signification dans le codage Manchester. En effet, lorsque le lecteur reçoit le signal composé des deux autres signaux, il décode un état haut et ne détecte donc aucun changement d'état durant le bit. En RFID, ce symbole considéré comme impossible dans le codage bit Manchester signifie la présence d'une collision sur le bit. Il permet donc de *détecter* et *localiser* une collision au sein d'un signal reçu.

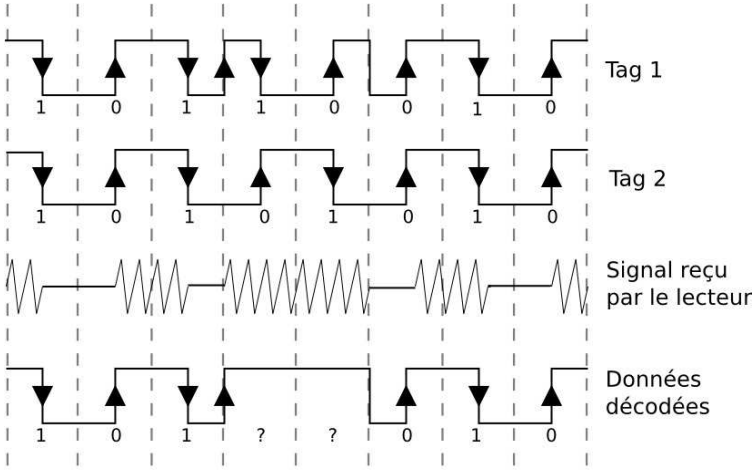


Figure II-10 : Collision avec un codage Manchester

Cas particulier : le tag « masqueur »

Dans certains cas particuliers, alors que plusieurs équipements émettent un message simultanément, il n'y a pas de collision [ZOR94]. En effet, la présence de collisions implique que chaque signal est perturbé par les autres signaux. Ainsi, si un signal est vraiment plus fort que les autres, il n'est pas perturbé par les autres signaux, et l'équipement destinataire est capable de traiter ce signal fort comme s'il était seul : il est capable de le *capturer correctement*. Afin de déterminer la présence ou non de collision dans les modèles parmi *N* signaux envoyés, il est possible d'utiliser cette équation [ZOR94] :

$$P_0 > c \sum_{j=1}^{N-1} P_j \qquad \text{(II.1)}$$

où P_0 représente la puissance du signal dont on souhaite vérifier s'il est capturable ; P_j la puissance du signal j, un des $N-1$ autres signaux ; et c le ratio de capture. Ainsi, si un signal a sa puissance c fois plus grande que la somme des puissances des autres signaux, alors ce signal masque les autres signaux et devient capturable. Il apparaît alors au destinataire comme s'il était le seul et unique signal présent dans le canal.

II.1.3.2 **Gestion des collisions**

Le paragraphe 0 ci-dessus introduit la notion de collision. Ce paragraphe présente l'importance pour les systèmes RFID de gérer correctement ces collisions [FIN03]. Afin de gérer ces collisions, il est nécessaire de partager le canal pour s'assurer que chacun des émetteurs ait la possibilité d'émettre son message. En RFID, l'accès multiple par domaine temporel (TDMA : Time Domain Multiple Access) est principalement utilisé : le canal de communication est utilisé à tour de rôle par chacun des participants. C'est une technique peu coûteuse en équipement car la gestion des temps de parole est faite au niveau « logiciel ». Ensuite, comme indiqué plus haut, les périodes de communication en accès multiple sont courtes et servent uniquement à identifier les tags. Les échanges d'information sont ensuite simplement faits avec un émetteur et un récepteur uniquement, il n'y a donc plus besoin de partager le canal de transmission.

Il existe principalement deux catégories d'algorithmes anticollision : les algorithmes déterministes et les algorithmes probabilistes [CHE07]. Les algorithmes de la première catégorie sont principalement réalisés à base d'arbre : les tags sont alors séparés en sous-groupes en fonction de leur identifiant. Les sous-groupes sont alors séparés en de nouveaux sous-groupes récursivement jusqu'à obtenir des sous-groupes contenant un identifiant unique. Ces algorithmes ont des temps d'exécution directement dépendant du nombre de tags et de leurs identifiants : les temps d'identification des tags présents dans le champ du lecteur sont bornés et constant quel que soit l'instant de leurs exécutions. Ce type d'algorithmes est notamment utilisé par la norme ISO-15696 [ISO15693-3] décrite dans l'annexe D. Néanmoins, en présence de nombreux tags, ces algorithmes sont particulièrement lents.

Afin de diminuer les temps d'inventaire, un autre algorithme est proposé : *Slotted ALOHA* [PIR08]. Cet algorithme divise le temps en intervalle appelé *slot*. Les tags devant être identifiés choisissent alors un slot au hasard en espérant ne pas choisir le même qu'un autre tag, ce qui engendrerait alors une collision : *Slotted ALOHA* est donc un algorithme probabiliste. Le temps n'est alors plus borné, mais, pour de nombreux tags, il est en moyenne beaucoup plus rapide que celui des algorithmes déterministes. Afin de diminuer le nombre de collision, les slots sont organisés en *frame*. Les tags ne sont autorisés à choisir qu'un seul slot par frame. La probabilité de collision et le temps d'inventaire des tags sont alors dépendants de la taille du frame, c'est-à-dire du nombre de slots dans un frame. Tout le problème de la configuration du système est alors dans le choix de cette taille. Mais pour cela, il faut connaître le nombre de tags à inventorier, ce qui n'est pas toujours le cas.

Pour contrer ce problème de taille de frame, un autre protocole est apparu : le *Dynamic Framed Slotted ALOHA* [SCH83]. Dans ce protocole, le lecteur choisit dynamiquement la taille des frames en fonction du nombre de collisions et du nombre de slots vides. C'est le protocole utilisé dans la norme *ISO-18000-6C*, appelé aussi *standard EPC Classe 1 Génération 2* et présenté dans la suite du document.

Norme ISO-18000-6C / EPC Classe 1 Génération 2

Le standard EPC Classe 1 Génération 2 est défini pour des applications logistiques utilisant la RFID UHF [EPCC1G2]. Il doit donc pouvoir fonctionner de manière optimale pour des systèmes pouvant être composés aussi bien de 10 tags que de 10 000 tags. Il utilise alors le protocole *Dynamic Framed Slotted ALOHA* (DFSA) pour gérer les collisions, permettant ainsi l'adaptation de la taille des frames au nombre de tags présents devant le lecteur à l'instant *t*.

Avant de parler de l'algorithme anticollision propre à cette norme, il est important de connaître les données contenues dans un tag. Un tag contient *quatre banques* de données. Chacune de ces banques a une fonction particulière bien définie : stocker les mots de passe (banque « **RESERVED** »), stocker l'identifiant unique **TID** du tag (banque « **TID** ») ou encore stocker l'identifiant EPC (banque « **EPC** ») et diverses autres informations concernant l'objet suivi spécifiques à l'utilisation qu'il en est faite (banque « **USER** »).

Au vu de la complexité et de l'hétérogénéité des identifiants uniques du tag ou de l'objet suivi, rendant alors la singularisation des tags, grâce à ces identifiants, complexe et longue, la norme a prévu d'ajouter à chaque tag un identifiant temporaire de 16 bits. A chaque début de tour d'inventaire, les tags choisissent chacun un *identifiant temporaire* de 16 bits appelé **RN16** ou **HANDLE**. Ainsi, la singularisation des tags se fera sur cet identifiant simple et homogène. Une fois la singularisation des tags faite, le lecteur pourra alors demander à chacun les informations dont il a besoin (identifiant EPC, identifiant TID, informations contenues dans la banque « **USER** »).

Afin de permettre à plusieurs lecteurs d'effectuer parallèlement des inventaires sur un même groupe de tags[1], ces derniers proposent *quatre sessions*, nommées **S0**, **S1**, **S2** et **S3**. Durant un tour d'inventaire, c'est-à-dire durant le parcours des slots d'un frame, les tags ne peuvent participer qu'à une seule et unique session : les tags ne peuvent donc pas être identifiés simultanément par deux lecteurs. Pour chaque session, les tags maintiennent un drapeau « **INVENTORIE** ». Chacun de ces quatre drapeaux peut prendre *deux valeurs* distinctes, notées **A** et **B**. Au début d'un tour d'inventaire, le lecteur choisit d'inventorier les tags **A** ou **B** dans une des quatre sessions. Les tags participant à un tour d'inventaire n'utiliseront ni ne modifieront les drapeaux « **INVENTORIE** » des autres sessions. Après une identification d'un tag par le lecteur, le tag inverse le drapeau « **INVENTORIE** » de la session d'inventaire en cours. Ainsi, un lecteur pourra faire plusieurs tours d'inventaire avec les tags **A**, jusqu'à ce que tous les tags soient passés en **B**, et ainsi auront tous été identifiés.

En plus des sessions et indépendamment de celles-ci, les tags ont un drapeau « **SELECTIONNE** », égal soit à **VRAI** soit à **FAUX**. Le lecteur pourra alors affiner son choix de tags participant au tour d'inventaire.

L'algorithme utilisé pour gérer les collisions est le *Dynamic Framed Slotted Aloha*. Le temps est donc divisé en intervalles de temps appelés *slots*. On a vu précédemment qu'au démarrage d'un frame le lecteur informe les tags du nombre de slots disponibles et les tags choisissent chacun un slot parmi les slots disponibles durant lequel ils vont communiquer. Le

[1] Ceci est utile notamment vis-à-vis de la sûreté de fonctionnement, comme il sera expliqué au chapitre II.2.2, page 40.

nombre de slots disponibles *L* est caractérisé par la variable *Q* dont la relation est donnée par la formule :

$$L = 2^Q \tag{II.2}$$

Ce slot est choisi à partir du **RN16** du tag et est compris entre *0* et 2^Q *- 1*. Les tags stockent alors ce numéro de slots dans leur compteur de slots. À chaque changement de slot, ce compteur de slots est décrémenté. Lorsqu'il est égal à **0**, le tag peut alors communiquer avec le lecteur et envoyer son **RN16**. Si le lecteur reçoit correctement le **RN16** – *i.e.* qu'il n'y a pas de collision, il informe le tag par acquittement de la bonne réception de ce **RN16** et donc de la bonne identification de ce tag. Le tag inverse alors le drapeau « **INVENTORIE** » de la session en cours et transmet son identifiant **EPC**.

Afin de réaliser un inventaire, le lecteur réalise 5 actions principales, chacune étant associée à une commande spécifique : (1) sélectionner un sous-groupe de tags avec la commande « **SELECT** » ; (2) démarrer un inventaire avec la commande « **QUERY** » ; (3) passer au slot suivant avec la commande « **QUERYREP** » ; (4) passer au tour d'inventaire suivant, *i.e.* au frame suivant, avec la commande « **QUERYADJUST** » ; et finalement (5) demander les informations utiles aux tags lorsqu'ils sont singularisés avec la commande « **ACK** ». Ainsi, le lecteur est capable de sélectionner un sous-groupe de tags présents dans son champ à partir des informations qu'ils contiennent, démarrer un inventaire ou encore passer au slot ou au frame suivant.

La Figure II-11 donne un exemple d'exécution de cet algorithme lorsque 3 tags sont présents dans le champ d'un lecteur. A l'état initial, le lecteur n'a vu aucun tag, et les tags sont dans leur configuration par défaut : toutes les sessions, de **S0** à **S3** sont à **A,** le drapeau « **SELECTIONNE** », nommé **SF** pour « Selected Flag », est égal à **FAUX** et ils n'ont choisi ni *d'identifiant temporaire* **RN16** ni de numéro de slot, sauvegardé dans le *compteur de slot*, marqué **SC** pour « Slot Counter ».

Lors de la phase de sélection, le lecteur informe tous les tags afin que tous mettent leur drapeau « **SELECTIONNE** » à **VRAI** avec la commande « **SELECT** ». Il démarre ensuite le tour d'inventaire dans la session **S0** en sélectionnant tous les tags dont la session **S0** est à **A** et dont le drapeau « **SELECTIONNE** » est égal à **VRAI** avec la commande « **QUERY** ». Il précise aussi, lors de l'envoi de cette commande, la taille des frames en envoyant le paramètre *Q*. Ici, *Q* est égal à 2, et donc, d'après l'équation (II.2) vue précédemment, il y a 4 slots, numérotés de 0 à 3. Chaque tag choisit alors un slot aléatoirement et stocke ce choix dans son compteur de slot **SC** : sur la Figure II-11, le tag #1 choisit le slot 1 et les tags #2 et #3 choisissent le slot 3. Cette commande démarre enfin le tour d'inventaire. Le système se retrouve donc dans le slot 0 du frame 0. Aucun tag n'a son compteur de slot **SC** égal à 0 : il n'y a donc pas de communication et le slot 0 est un *slot vide*.

Le lecteur peut alors informer les tags qu'ils doivent passer au slot suivant grâce à la commande « **QUERYREP** ». À réception de cette information, le système se trouve dans le slot 1 du frame 0. Les tags décrémentent alors de 1 leur compteur de slot **SC**. Les compteurs de slot **SC** des tags #2 et #3 deviennent alors égaux à 2 et le compteur de slot **SC** du tag #1 devient alors égal à 0, ce qui permet au tag #1 de communiquer son numéro temporaire **RN16** au lecteur. Comme seul le tag #1 envoie une réponse dans ce slot, il n'y a pas de collision et le lecteur est capable de lire les données contenues dans le signal envoyé. Il en informe les tags en acquittant avec la commande « **ACK** » la bonne réception de l'identifiant temporaire **RN16** du tag ayant communiqué – le tag #1. En recevant cet acquittement, le tag dont le **RN16** correspond au sien comprend qu'il est le seul à avoir choisi ce slot et peut donc envoyer son identifiant EPC. Il passe aussi son drapeau « **INVENTORIE** » de la session **S0** à **B** et sort du tour d'inventaire. Le slot 1 est un *slot singleton*.

Le lecteur informe alors les tags qu'ils doivent passer au slot suivant grâce à la commande « **QUERYREP** » et les tags restants décrémentent alors leur **SC** de 1. Tous les compteurs de slot **SC** sont différents de 0, il n'y a donc pas de tag autorisé à communiquer avec le lecteur. Le slot 2 est aussi un *slot vide*.

Le lecteur informe enfin les tags qu'ils doivent passer au dernier slot. Les tags restants diminuent leur compteur de slot **SC** de 1. Les tags #2 et #3 ont leur **SC** égal à 0 et sont donc autorisés tous les deux à communiquer avec le lecteur. Ils envoient alors en même temps au lecteur leur **RN16**. Ce double envoi génère alors une collision, empêchant le lecteur d'identifier la présence du tag #2 et du tag #3. Celui-ci ne répond pas, les tags comprennent alors qu'il y a eu une collision et ne changent pas leur drapeau « **INVENTORIE** », restant ainsi dans le tour d'inventaire. Le frame 0 se termine alors sur ce *slot collision*.

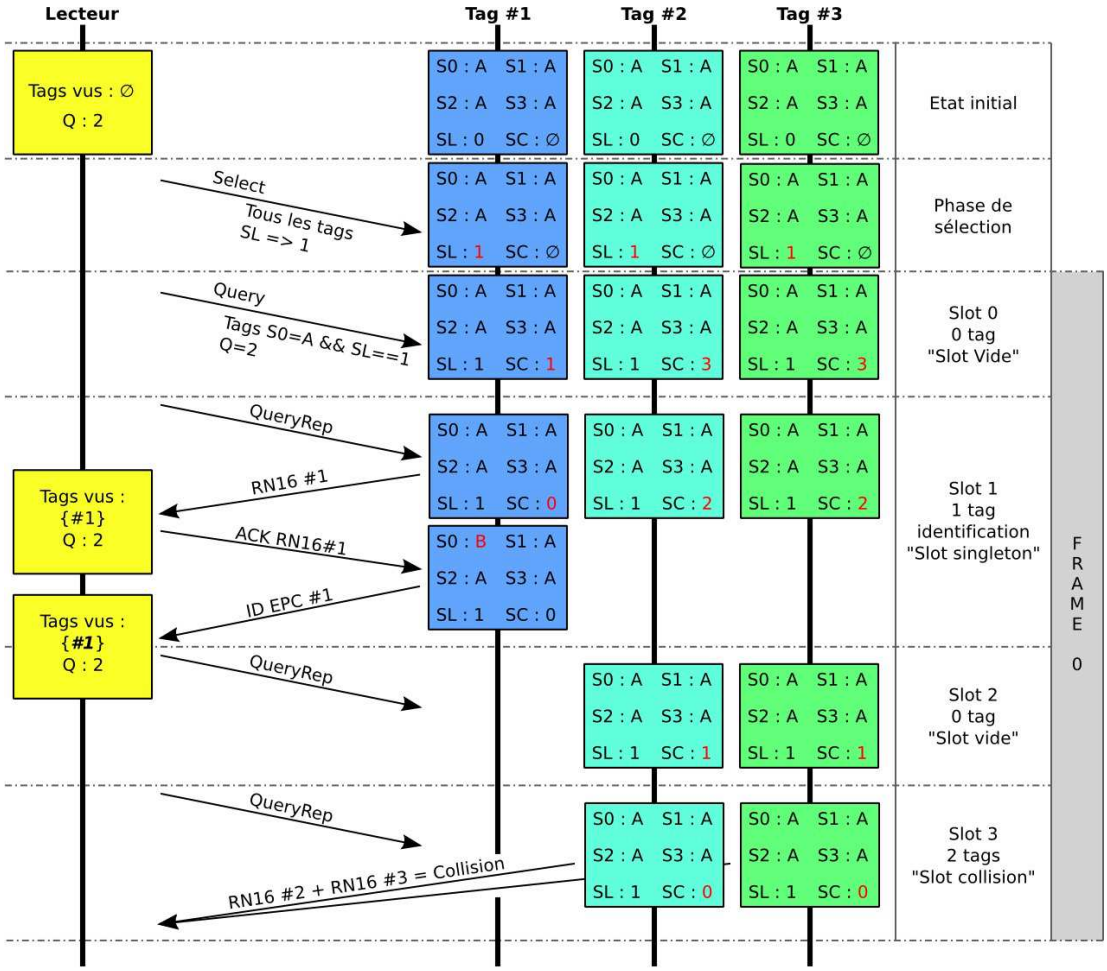


Figure II-11 : Exemple d'exécution de l'algorithme EPC Classe 1 Génération 2

Pour inventorier les tags restants, le lecteur passe au frame suivant avec la commande « **QUERYADJUST** ». Il profite de cette commande pour informer les tags d'un éventuel changement de taille de frame. À réception de cette commande, les tags encore dans le tour d'inventaire choisissent alors un nouveau slot parmi les slots disponibles et le sauvegardent dans leur compteur de slot **SC**. Ceux ayant choisi le slot 0 sont alors autorisés à communiquer. Ce slot est alors soit *vide*, *singleton* ou *collision*. Le lecteur parcourt alors les slots de ce frame un à un afin d'offrir la possibilité aux tags restants de communiquer. Une fois les slots parcourus, et s'il reste encore des collisions, le lecteur peut alors ajouter autant de frames que nécessaire et offrir d'autres slots aux tags.

A la fin du tour d'inventaire, le lecteur a alors résolu les collisions et identifié tous les tags présents dans son champ respectant les conditions imposées au départ.

Il est à noter qu'il est possible de prédire le nombre de slots *vides V*, *singletons S* ou *collisions C* durant un frame, connaissant le nombre de tags *N* à inventorier durant un frame et le nombre de slots *L* contenus dans le frame [LIU06]. En effet, le nombre de tags *t* présents dans un des slots suit une loi binomiale :

$$P(X = t) = \binom{N}{t}\left(\frac{1}{L}\right)^{t}\left(1 - \frac{1}{L}\right)^{N-t} \quad \text{(II.3)}$$

Le nombre de slots *m* ayant *t* tags peut donc être exprimé comme suit :

$$m = L\binom{N}{t}\left(\frac{1}{L}\right)^{t}\left(1 - \frac{1}{L}\right)^{N-t} \quad \text{(II.4)}$$

Donc, d'après l'équation (II.4), on peut exprimer le nombre de slots ayant 1 seul et unique tag – slots singletons –, le nombre de slots n'ayant aucun tag – slots vides – et le nombre de slots ayant au moins 2 tags – slots collisions :

$$S(N, L) = N\left(1 - \frac{1}{L}\right)^{N-1} \quad \text{(II.5)}$$

$$V(N, L) = L\left(1 - \frac{1}{L}\right)^{N} \quad \text{(II.6)}$$

$$C(N, L) = L - S(N, L) - V(N, L) \quad \text{(II.7)}$$

Par exemple, d'après ces équations, et, dans le cas de 10 tags à inventorier avec 16 slots dans le frame, il y aura en moyenne

- $S(10{,}16) = 5{,}6$ slots singletons
- $V(10{,}16) = 8{,}4$ slots vides
- $C(10{,}16) = 2$ slots collisions

II.1.4 Middleware

Les middlewares RFID sont une nouvelle classe de *programmes* faisant *l'interface* entre les équipements physiques – lecteurs et tags – et les applications métiers – SAP, gestion de stock, contrôle d'accès, … [KRI07]

L'existence de cette nouvelle classe de programme est due à trois motivations principales [GLO06] :

1. *Simplifier l'accès aux lecteurs* en encapsulant les programmes d'interfaces des lecteurs de plus en plus complexes, couplés parfois à des capteurs ;
2. *Gérer la grande quantité de données brutes* issues des lecteurs et des capteurs pour fournir aux applications métiers seulement les données utiles ;
3. *Offrir une interface applicative* permettant à l'utilisateur de gérer les lecteurs.

Pour répondre à ces besoins, le middleware est souvent un programme distribué : il s'exécute sur plusieurs machines différentes en réseau. Ainsi, les différents composants s'exécutent à l'endroit le plus pertinent.

Le middleware peut donc être divisé en trois composants principaux, représentés sur la Figure II-12, et répondant à ces trois motivations principales : le composant de *gestion des équipements*, le composant de *gestion des données*, et le composant de *gestion des services*.

II.1.4.1 **Composant de « *gestion des équipements* »**

Le premier composant est le service de gestion des équipements, équivalent à un service d'abstraction matérielle. Il est là pour permettre à l'utilisateur final de ne pas avoir à gérer la complexité et l'hétérogénéité de ses équipements. En effet, il n'est pas rare qu'une demi-

douzaine de types différents de lecteurs provenant de différents fournisseurs soit présente dans une entreprise. Chaque lecteur ayant une interface spécifique, il devient difficile d'interconnecter tous ces équipements. En effet, en plus d'avoir une connectique différente – RS232, Ethernet, USB, Wifi, … – les commandes sont souvent différentes et les capacités et options sont très différentes. Le middleware doit donc être dans la capacité de configurer et gérer chacun des équipements pour donner l'illusion à l'utilisateur qu'ils sont tous identiques.

Cette partie du middleware est souvent installée proche des équipements. Elle est donc localisée dans le lecteur.

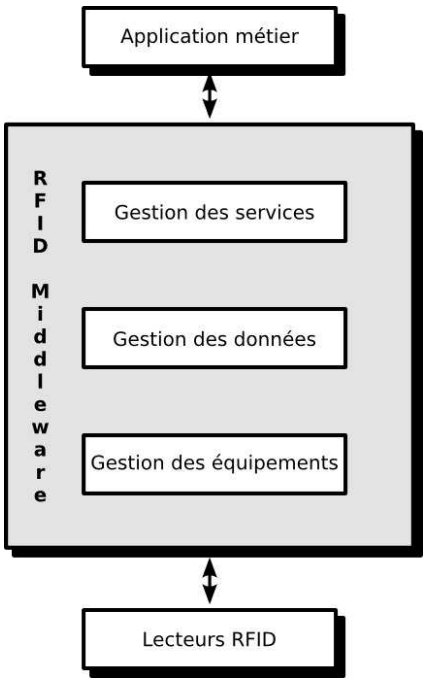


Figure II-12 : Composants d'un middleware RFID

Le middleware peut aussi offrir des solutions de monitoring des lecteurs. Ce monitoring a pour objectif de s'assurer du bon fonctionnement du système, améliorant ainsi la sûreté de fonctionnement du système RFID. Des exemples de monitoring seront donnés dans la section II.2.2 dédiée à la sûreté des systèmes RFID.

Ainsi, le middleware doit pouvoir gérer ces équipements, et masquer à l'utilisateur final la complexité de l'infrastructure RFID. Il doit donc déclencher et configurer spécifiquement chaque équipement en fonction des informations haut-niveau fournies par l'utilisateur grâce au composant de *gestion des services* présenté au paragraphe II.1.4.3. De plus, il fournit à l'utilisateur une vue homogène des différents équipements.

Les données récupérées par ce composant sont brutes et abondantes. En effet, il n'est pas rare d'avoir plusieurs lecteurs à un point de lecture, ou des lecteurs effectuant des inventaires à répétition, générant énormément d'observations de tags par seconde[1]. Toutes ces observations n'ont pas forcément beaucoup de signification pour l'utilisateur, en particulier à cause du nombre d'observations identiques. Les observations effectuées par ce composant sont donc transmises au composant de « *gestion des données* » afin que ce dernier les mette en forme et ne retienne que les observations utiles pour l'utilisateur final.

[1] Pouvant aller jusqu'à des millions dans le cas d'applications logistiques de grande envergure.

II.1.4.2 **Composant de « *gestion des données* »**

Comme expliqué précédemment, ce composant a pour objectif la récupération et le tri des observations RFID brutes afin de les transformer en observations utiles à l'utilisateur final. Ces observations sont organisées, triées, filtrées et enregistrées dans des rapports en fonction des informations données par l'utilisateur via le composant de « *gestion des services* » décrit au paragraphe II.1.4.3. Ces rapports seront ensuite envoyés directement aux applications métiers qui en ont besoin. L'objectif est de réduire le nombre d'observations au strict nécessaire. La Figure II-13 montre l'utilité de ce composant : ce composant réduit le nombre d'observations brutes pour que les observations applicatives soient les plus pertinentes possibles.

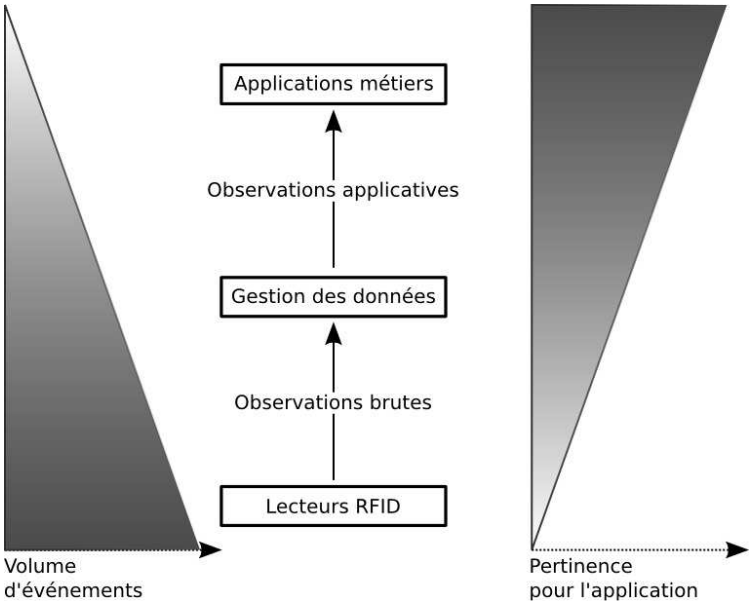


Figure II-13 : Volume des événements et pertinence des données en fonction des différents niveaux d'un système RFID

La première étape est la *récupération* et *l'agrégation* des données générées par les lecteurs. Ceci va permettre au middleware de travailler avec la totalité des données dont il a besoin. Un premier tri va permettre de supprimer les données redondantes. Cette agrégation et ce premier tri permettent, entre autres, de regrouper et trier les données issues de plusieurs lecteurs couvrant la même zone de lecture. En effet, il n'est pas rare en RFID d'avoir plusieurs lecteurs sur un même point de lecture afin d'avoir un taux de lecture optimal. Les données issues de ces lecteurs doivent être analysées comme étant issues d'un seul et même lecteur. Le middleware va donc regrouper les données de ces lecteurs et supprimer les lectures redondantes des tags présents devant plusieurs lecteurs.

La seconde étape va être *l'analyse* et la *transformation* des données. Les données brutes correspondent à des événements de lectures peu digestes pour l'application finale. Ces événements de lectures brutes vont donc être analysés puis transformés en événements applicatifs grâce aux règles établies par l'utilisateur. Par exemple, lorsqu'un tag reste présent devant un lecteur, cela génère une quantité d'événements de lectures. En effet, ce tag va être lu à chaque inventaire tant qu'il sera dans le champ du lecteur. Ces multiples observations vont alors être transformées en deux observations : le tag est entré dans le champ du lecteur à l'instant *t* et le tag est sorti du champ du lecteur à l'instant *t'*.

La dernière étape est la *création* et *l'envoi* de rapports aux différentes applications métiers. Pour cela, le lecteur va *filtrer* les événements applicatifs en fonction de règles établies par

l'utilisateur et ainsi créer les différents rapports qui seront envoyés automatiquement aux applications métiers.

Ainsi, par les étapes d'*agrégation*, de *transformation*, de *création* et d'*envoi*, le middleware va fournir aux applications métiers des informations en nombre restreint mais dont la pertinence est totale – Figure II-13.

II.1.4.3 Composant de « *gestion des services* »

Les deux composants précédents ont besoin de règles pour fonctionner. Ces règles sont spécifiques à chaque système RFID et doivent être définies par l'utilisateur. Ce dernier composant offre donc la possibilité à l'utilisateur de définir ces règles. Les règles sont énoncées en général vis-à-vis de l'application finale.

Le premier service offert par ce composant est la visualisation de l'infrastructure RFID. Afin de permettre à l'utilisateur d'exprimer des règles complexes, ce composant va différencier deux types de lecteurs : les lecteurs *physiques* et les lecteurs *logiques*. Les lecteurs physiques sont les lecteurs réellement connectés au réseau, alors que les lecteurs logiques sont un regroupement de lecteurs physiques. En effet, dans les systèmes RFID, afin de garantir un bon fonctionnement du système, plusieurs lecteurs sont souvent utilisés à un point de lecture. Ceci permet d'avoir une zone de lecture plus grande, garantissant des taux optimaux de lectures. Ces lecteurs physiques sont alors réunis ensemble et visibles par l'utilisateur comme un seul et unique lecteur : ce lecteur est un lecteur *logique*. Certains middlewares offrent même la possibilité de créer des lecteurs logiques à partir des antennes des lecteurs plutôt qu'à partir des lecteurs physiques eux-mêmes comme le montre la Figure II-14. Afin d'éviter que les lecteurs physiquement proches et partageant donc une partie de leur champ de lecture ne se perturbent entre eux, le middleware va réaliser un multiplexage temporel : ces lecteurs vont alors inventorier les tags présents dans leurs champs à tour de rôle.

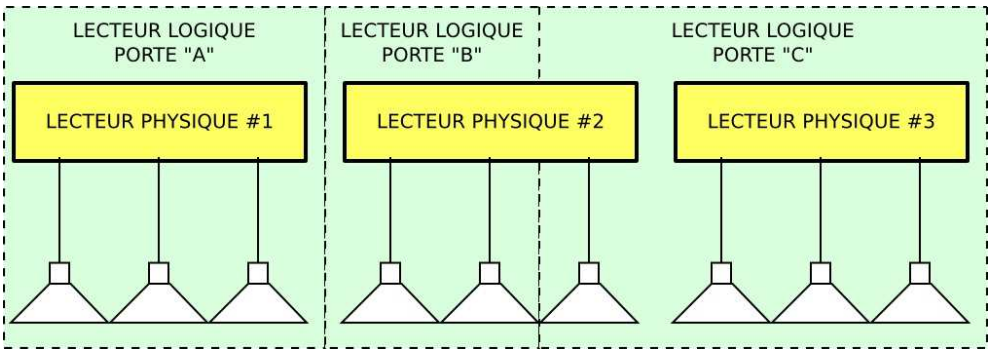


Figure II-14 : Lecteurs physiques et lecteurs logiques

II.1.4.4 Conclusion

Le middleware fournit à l'utilisateur les outils pour gérer et utiliser son infrastructure RFID. La Figure II-15 résume les différents services offerts par un middleware. Il y a un service de gestion des équipements permettant de masquer à l'utilisateur la complexité et l'hétérogénéité de l'infrastructure. Un autre service est en charge de collecter les données, de les mettre en forme puis de les transmettre aux applications métiers dans le format le plus approprié. Finalement, un dernier service met à disposition de l'utilisateur des outils de configuration et de gestion du système.

Le middleware est un logiciel distribué : les différentes fonctionnalités s'exécutent sur différentes machines en réseau, permettant ainsi d'être le plus efficace possible. En effet, toute la gestion d'un lecteur est plus efficace proche de celui-ci alors que la gestion des

données de l'ensemble des lecteurs doit être faite sur un serveur centralisant toutes les données issues des différents lecteurs.

II.1.4.5 Exemples de middleware RFID

Il existe actuellement bon nombre de middlewares RFID répondant à différents besoins. Certains sont Open Source, d'autres propriétaires. Ils peuvent être basés sur des interfaces et des mécanismes décrits par des standards tels que le standard *« Application Level Events (ALE)* » établi par la société EPCGlobalInc. [EPCALE1, EPCALE2]. On pourra citer par exemple les middlewares suivants :

1. *Aspire RFID* [KEF08] est un exemple de middleware Open Source respectant les spécifications ALE. Ce middleware est issu d'un projet européen visant au développement d'un middleware léger, gratuit, hautement personnalisable, facile d'utilisation, respectant les standards pour permettre un développement et un déploiement bas coût des solutions RFID.
2. *RF²ID* [AHM07] est un autre middleware fortement distribué développé par l'institut technologique de Géorgie à Atlanta. Il est construit autour du concept de lecteurs virtuels et de chemins virtuels. Chaque lecteur virtuel est associé à un lot de lecteurs physiques géographiquement proches. Chaque lecteur virtuel doit récupérer et filtrer les données issues de ses lecteurs physiques, les dater, gérer les chemins virtuels et gérer les demandes. Les chemins virtuels sont des canaux dynamiques créés par les lecteurs virtuels afin de fournir des fonctionnalités comme la rapidité de réponse aux requêtes et l'équilibrage de la charge de tous les lecteurs.

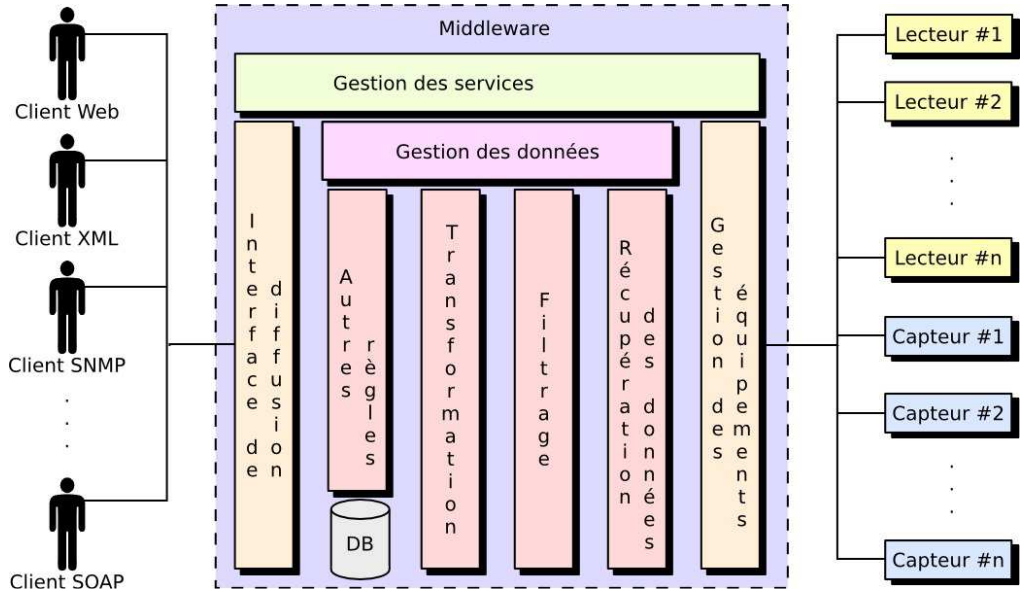


Figure II-15 : Architecture générique d'un middleware RFID

II.2 Sûreté de fonctionnement des systèmes RFID

La sûreté de fonctionnement d'un système est définie [LAP96] comme étant la « *propriété qui permet aux utilisateurs du système de placer une confiance justifiée dans le service qu'il leur délivr*e ». C'est donc une propriété importante à évaluer afin de déterminer si le système est suffisamment fiable pour être utilisé dans telle ou telle application. Cette section explique donc dans un premier temps ce qu'est la sûreté de fonctionnement et comment elle est évaluée

puis dans un second, quelles sont les techniques utilisées afin d'améliorer la confiance que l'on peut placer dans un système RFID.

II.2.1 Notions de base de la sûreté de fonctionnement

La sûreté de fonctionnement est caractérisée par cinq points principaux, appelés *attributs* de la sûreté de fonctionnement [ARL06, KNI02, LAP96] :

1. la **disponibilité** : c'est l'aptitude d'un composant ou d'un système à *délivrer* le service attendu à un *instant t* ;
2. la **fiabilité** : c'est l'aptitude d'un composant ou d'un système à *fonctionner* pendant un *intervalle de temps* $[t_0; t]$;
3. la **maintenabilité** : c'est l'aptitude d'un composant ou d'un système à être *réparé* ;
4. la **sécurité-innocuité** : c'est l'aptitude d'un composant ou d'un système à ne pas *conduire* à des *accidents catastrophiques*, en terme humain, matériel ou financier ;
5. la **sécurité-immunité** : c'est l'aptitude d'un composant ou d'un système à *garantir **l'intégrité*** des données qu'il contient et à en *empêcher l'accès* ou les *manipulations* non autorisées (c'est-à-dire à garantir la ***confidentialité*** de ces données).

La Figure II-16 illustre les relations existantes entre les différents attributs de la sûreté de fonctionnement [CAU04]. En effet, la fiabilité a un impact direct sur la disponibilité : si la fiabilité est basse, il va y avoir beaucoup de défauts, et ainsi diminuer la disponibilité. De plus, il est connu qu'il y a souvent une corrélation entre l'apparition d'un défaut et l'occurrence d'un accident : la fiabilité impacte donc aussi la sécurité-innocuité. Une maintenabilité inadaptée, dans le cas des systèmes réparables, peut compromettre la disponibilité du système car le nombre de défauts peut augmenter et le temps de réparation peut être long. En augmentant le nombre de défauts possibles, la maintenabilité augmente par conséquence le nombre possible d'accidents, diminuant par la même occasion la sécurité-innocuité. Enfin, en général, la sécurité-innocuité va avoir un impact néfaste sur la disponibilité. En effet, dans la majorité des cas, la sécurité-innocuité va imposer un service dégradé en présence de défauts pour minimiser le risque de panne, ce qui va avoir pour effet de diminuer la disponibilité – par exemple, l'arrêt d'un moteur suite à un défaut. Néanmoins, dans certains cas, la nécessité de la sécurité-innocuité impose une forte disponibilité comme dans le cas d'un système de freinage de voiture ou des systèmes de commandes d'un avion.

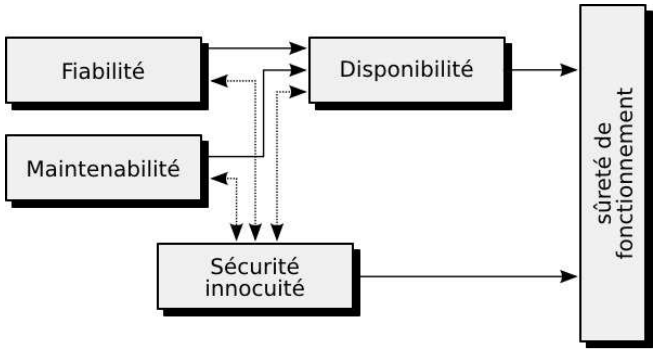


Figure II-16 : Relation entre les attributs de la sûreté de fonctionnement [CAU04]

Afin d'améliorer la sûreté de fonctionnement, plusieurs moyens sont à notre disposition. Ces moyens peuvent être classés en quatre catégories [LAP96] :

1. Elimination des fautes. Cette catégorie regroupe les techniques visant à réduire le nombre et la gravité des fautes :

- Validation de conception
- Analyse de testabilité
- Test logiciel
- Test de production

2. Tolérance aux fautes. Cette catégorie regroupe les mécanismes permettant au système de continuer à assurer un service malgré la présence de fautes :
   - Redondance
   - Test en ligne
   - Diagnostique en ligne
   - Reconfiguration
3. Prévision des fautes. Cette catégorie a pour objet l'estimation de la présence et des conséquences des fautes :
   - Simulation des fautes
   - Evaluation des attributs
4. Prévention des fautes. Cette catégorie regroupe les moyens dont le but est d'empêcher l'introduction de fautes dans le système :
   - Conception particulière
   - Redondance matérielle
   - Redondance logicielle
   - Contrôle qualité

L'étude de la sûreté de fonctionnement est dépendante des applications. En effet, toutes les applications n'attachent pas la même importance à ses différents attributs. Ainsi, les moyens pour améliorer la sûreté de fonctionnement doivent être étudiés pour une application particulière.

II.2.2 Approche utilisées pour la sûreté de fonctionnement des systèmes RFID

Les systèmes RFID n'échappent pas à une étude de la sûreté de fonctionnement dépendant de l'application, prenant en compte ses caractéristiques intrinsèques et les besoins de l'utilisateur. [LOD06] a identifié les différents paramètres permettant d'évaluer la fiabilité et la disponibilité des systèmes RFID. La principale est la *capacité des tags à être lus*. En effet, la fonction principale d'un système RFID est la lecture de l'identifiant contenu dans le tag. De plus, cette capacité est directement impactée par les principaux facteurs engendrant une baisse de fonctionnement : la capacité du tag à récupérer l'énergie envoyée par le lecteur et la capacité du lecteur à envoyer de l'énergie aux tags ; la géométrie et l'architecture des antennes et des tags ; les protocoles anticollisions …

De plus, les systèmes RFID sont des systèmes embarqués, contenant à la fois du matériel et du logiciel qui fonctionnent en étroite collaboration. Ainsi, comme l'indique [KNI02], il est important de prendre en compte les interactions existant entre le logiciel et le matériel : les défaillances de l'un doivent être gérées par l'autre. Ainsi, le logiciel peut venir aider au test et au diagnostic des équipements matériels pouvant tomber en panne.

Il existe beaucoup d'études concernant l'amélioration de la sécurité-immunité des systèmes RFID. En effet, les systèmes RFID manipulent une grande quantité de données pouvant porter atteinte à la vie privée [KAR07]. L'exemple le plus courant est le paiement sans contact. Les données échangées concernant les données bancaires, cette application demande

donc une grande sécurité. En effet, les transactions pouvant être espionnées, il est donc important de protéger ces informations car elles renseignent sur les habitudes des personnes. Des personnes mal intentionnées peuvent alors s'en servir pour tracer les utilisations des usagers. Cette sécurité apparaît clairement nécessaire pour ce genre de systèmes (contrôle d'accès, paiement sans contact, …) mais elle est tout aussi nécessaire pour la gestion de stock. En effet, lorsqu'un magasin utilise la RFID pour gérer son stock, chaque produit est associé à un tag contenant des informations permettant d'identifier le type de produit, le fabricant ou encore son prix. Ainsi, tous les produits que le client souhaite acheter peuvent être identifiés lors du passage en caisse. Mais, une fois hors du magasin, pour préserver la vie privée, il est important que les tags ne soient plus lisibles, afin d'empêcher quiconque de voir ce que le client a acheté sans son accord. De ce fait, de nombreuses études existent et proposent des solutions, aussi bien au niveau applicatif [MAL08], middleware [DON10, NGU11], lecteur [PAR06, MAJ07] ou bien tag [PAR07, PIA10].

Notre étude s'intéresse particulièrement à l'étude de la fiabilité et de la robustesse[1] des systèmes RFID. Afin d'améliorer leur qualité, plusieurs méthodes existent. Les tests de production sont la première étape et permettent d'éliminer, avant leur utilisation, les tags dont les caractéristiques dévient de celles assurant un bon fonctionnement de l'équipement. [NAT07, MUR04] proposent des conceptions de tags facilitant le test de production. Ils soulignent dans un premier temps les difficultés pour tester les tags, difficultés dues à leurs signaux mixtes analogiques/numériques, à leurs petites tailles (puce de surface inférieure à 1mm²) et donc au peu de place disponible pour du test intégré, à leurs fonctionnalités asynchrones, et à leur faible prix (environ 5 cents), ainsi qu'à la présence de mémoires non volatiles. De plus, du fait de la communication par modulation de charge ou par rétro-modulation [PAR01, FIN03], il est impossible d'utiliser des techniques classiques de bouclage pour les tests des transmetteurs RF [NEG07]. Néanmoins, des équipements ont été conçus afin de simuler le signal provenant de l'antenne du tag et sont capables d'analyser la modulation de charge effectuée par le tag [COS00]. Ainsi, la puce peut être testée complètement avant intégration avec l'antenne, garantissant ainsi aussi sa capacité à décoder le signal provenant du lecteur et à générer un signal de réponse.

Les tests de caractérisation sont une seconde étape possible. [LOD06] propose aussi, pour garantir la fiabilité et la disponibilité des systèmes RFID, d'effectuer des tests avant la mise en service du système. Pour cela, il caractérise différents paramètres comme la distance de lecture, la capacité du tag à fonctionner dans une mauvaise position, la vitesse d'inventaire ou encore le nombre de tags lisibles par unité de temps. Ainsi, après ces différents tests, les limites du système sont connues ainsi que les configurations permettant de garantir le bon fonctionnement du système.

Après les tests de caractérisation, il peut apparaître nécessaire de rajouter de la redondance dans le système. Des techniques existent ainsi pour améliorer la tolérance aux fautes, la fiabilité et la disponibilité. Ainsi, afin d'assurer la complète couverture d'une zone de lecture et garantir la détection de tous les tags présents à ce point de lecture, plusieurs lecteurs sont classiquement utilisés en parallèle [JAC09]. Le résultat de l'inventaire est par conséquence l'union des résultats des inventaires de chaque lecteur. Une autre redondance possible est la redondance des tags. [BOL07] propose ainsi d'équiper chaque objet avec deux tags à la place d'un seul, principalement pour les applications logistiques : lorsqu'au moins un des deux tags est détecté, l'objet est considéré comme détecté. Cela va permettre d'éviter différents problèmes liés à l'utilisation de la RFID :

[1] La robustesse est définie comme étant la sûreté de fonctionnement par rapport aux fautes externes [ARL06]

- la perte d'un tag. En effet, durant le transport, il arrive que quelques tags se détachent ;
- la dégradation d'un tag, suite à des chocs, un mauvais stockage ou encore tout simplement l'usure normale ;
- la mauvaise orientation du tag. En effet, les performances des tags sont directement impactées par leur positionnement face à l'antenne de lecture. En mettant plusieurs tags, il est alors possible de les placer différemment et donc d'avoir des bonnes performances pour au moins un des deux tags. Dans la même idée, [RAH07] étudie l'impact du nombre de tags et du nombre d'antennes par tag sur la fiabilité. Il met ainsi en évidence que, plus il y a de tags, et d'antennes par tag, plus le système est fiable. Néanmoins, la fiabilité dépend du nombre d'objets suivis en même temps.

Toutefois, l'utilisation de ces redondances présente des inconvénients : (1) une augmentation du coût du système, due à la multiplication des équipements et (2) une augmentation du temps de chaque inventaire et du traitement des données. En effet, dans le cas de plusieurs lecteurs redondants, les lecteurs devront effectuer leurs inventaires les uns après les autres afin de ne pas interférer les uns avec les autres. Dans le cas de plusieurs tags redondants, les temps d'inventaire étant directement dépendants du nombre de tags, ils dureront donc plus longtemps. En plus de cela, dans les deux cas (redondance de lecteurs ou de tags), il y aura une augmentation artificielle du nombre de données à gérer, demandant alors au middleware plus de travail, risquant de le surcharger et donc dégrader ses performances et sa capacité à répondre aux demandes des applications métiers.

Toutes ces méthodes permettent de limiter l'apparition de fautes ou de comportement non désiré durant la durée de vie du système. Néanmoins, elles n'empêchent absolument pas leur apparition. Il est donc nécessaire de vérifier, tout au long de sa vie et de préférence pendant son fonctionnement, le bon fonctionnement et le bon état du système. Pour cela, il est nécessaire d'utiliser des méthodes de test en ligne.

II.2.2.1 **Test en ligne**

Les méthodes de test en ligne permettent à l'utilisateur de vérifier le bon fonctionnement du système durant son utilisation. Ainsi, les différentes propriétés du système sont évaluées en continu pour vérifier leur maintien. Il existe deux types de test en ligne :

1. les méthodes non-intrusives : ces méthodes analysent les informations déjà disponibles présentes dans le système et ne génèrent donc aucun fonctionnement non productif ni ne modifient le fonctionnement du système ;
2. les méthodes intrusives : ces méthodes changent le fonctionnement du système afin de lui demander d'effectuer des actions leur permettant d'établir une décision quant à sa bonne santé.

Différentes techniques ont déjà été proposées afin de vérifier le bon fonctionnement des systèmes RFID. La plupart de ces techniques reposent sur l'étude de l'application elle-même et sont gérées au niveau middleware. En effet, cet élément logiciel du système possède une vue générale de toute les activités qui se passent au sein du système. Il est donc le plus apte à prendre des décisions quant à la santé des différents équipements.

Ainsi, dans l'objectif de déterminer si des fautes sont présentes ou pas dans le système, des techniques de test en ligne, appelé *monitoring*, ont été proposées [GLO06]. Cependant, ces techniques s'intéressent particulièrement à la santé des lecteurs et de l'infrastructure réseau. Une des techniques les plus simples est la demande régulière aux lecteurs s'ils sont toujours présents et bien allumés par de simples commandes *ICMP* [GLO06]. Ces commandes,

couramment utilisées en réseau, permettent de vérifier la présence d'un équipement réseau. Le middleware envoie un message de type « **ECHO** » et si le lecteur est en fonctionnement et correctement connecté au réseau, il répond un « **ECHOREPLY** ». Cette technique permet de vérifier facilement si un lecteur est allumé et si le réseau fonctionne correctement. Mais elle ne renseigne absolument pas sur la capacité du lecteur à remplir ses fonctions.

Pour vérifier le bon fonctionnement des systèmes RFID, de manière non intrusive, il est alors possible de mesurer les performances de ces systèmes et de les comparer à des valeurs attendues :

1. le nombre de tags lus au cours du temps
2. le nombre d'erreurs de lecture

Les deux paragraphes suivants expliquent en détails ces différentes techniques.

Volume moyen de trafic de tags

Cette première méthode de monitoring de lecteur, qui s'appelle *volume moyen de trafic de tags*, ou *ATTV*, pour « Average Tag Traffic Volume », s'attache au nombre de tags vus par les lecteurs par unité de temps. Cette unité de temps est définie en fonction de l'application et peut être égale à 1 minute comme à 1 heure. Une première étape est l'étude statistique du flux des tags. Ainsi, il sera possible d'avoir un graphique représentant le volume de tags lus par unité de temps, en fonction des différentes périodes de la journée.

La Figure II-17 donne un exemple du résultat d'une étude statistique du flux de tags à un point de lecture. On voit ainsi que de 0 à 4 heures, il n'y a pas de lecture de tags. Entre 4 et 8 heures et entre 20 heures et minuit, 50 tags par minute sont lus. Entre 8 et 16 heures, 100 tags par minute sont lus. Et enfin entre 16 et 20 heures, 150 tags par minute sont lus.

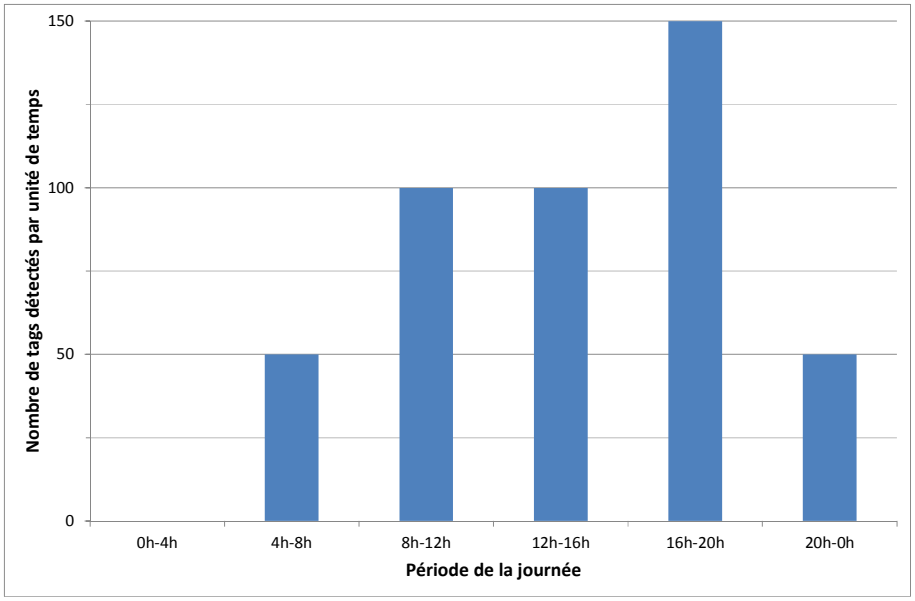


Figure II-17 : Exemple de volume de trafic de tags en fonction de la période de la journée

Ainsi, il sera possible de vérifier le bon fonctionnement d'un lecteur. En effet, si le lecteur a un ATTV vraiment différent de celui mesuré pendant la phase d'étude, c'est qu'il y a un problème soit dans le processus de l'application, soit dans le lecteur concerné.

Cette technique de monitoring impose une bonne connaissance du processus utilisant le système RFID. De plus, il faut un minimum d'homogénéité dans le processus : en effet, il faut que le flux de tags soit périodiquement identique. Ainsi, la Figure II-17 représente le flux des

tags sur une journée. Cette méthode est efficace si tous les jours se ressemblent. Il est encore pensable d'utiliser cette méthode lorsqu'il faut étudier les flux sur une semaine. Mais lorsque les flux varient au cours du mois, voire plus, ou si les flux ne sont pas constants, cette méthode devient inapplicable. De plus, il n'est pas possible de détecter des erreurs au sein des tags. Il est seulement possible, de cette manière, de détecter la présence de fautes graves présentes sur une longue durée – durée égale à l'intervalle d'observation.

En présence d'un processus constant, cette technique de monitoring va permettre de détecter la présence de fautes au sein d'un groupe de tags, de lecteurs ou de leur environnement. En effet, prenons l'exemple d'une entreprise traitant toujours le même type de produits et le même nombre de produits[1]. Ainsi, les lectures qu'il y aura à effectuer concerneront toujours le même type de population de tags : même nombre, même disposition par rapport aux antennes. La Figure II-18 donne un exemple d'application : l'application consiste en une palette de 110 tags. Cette palette est mise sur une plateforme tournante entre deux antennes afin d'effectuer l'inventaire de tous les tags présents. Après étude du système, on s'aperçoit en l'absence d'erreur, qu'à chaque inventaire, 108 tags sont détectés au minimum. Ainsi, lorsque le système détectera moins de tags – *i.e.* que le volume de tags durant un intervalle de temps correspondant au passage d'une palette est plus faible –alors il y a probablement une défaillance au niveau des tags, du lecteur ou de l'environnement. Il faudra alors effectuer une étude détaillée permettant de diagnostiquer l'origine de la défaillance.

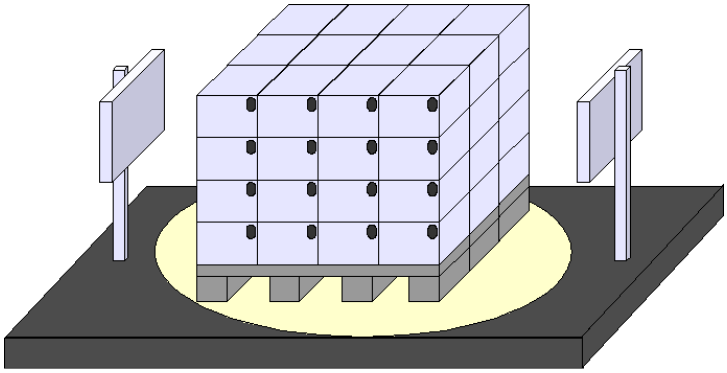


Figure II-18 : Exemple de système RFID pour la traçabilité sur une chaîne de distribution industrielle

Nombre d'erreurs de lectures par rapport au nombre de lectures total

Une autre technique de monitoring consiste à surveiller le nombre d'erreurs de lecture par rapport au nombre total de lectures pendant une période donnée. Cette technique est appelée *RETR* pour « Read Errors to Total Reads ». Une lecture réussie est une lecture dont le résultat est l'obtention de l'identifiant de l'objet taggé. Le nombre d'erreurs de lectures par rapport au nombre total de lectures est calculé à l'aide de l'équation suivante :

$$RETR = \frac{\sum_{i=1}^{N} e_i}{\sum_{i=1}^{N} e_i + \sum_{i=1}^{N} s_i} \quad \text{(II.8)}$$

Où N est le nombre d'inventaires, e_i est égal à 1 si une erreur de lecture lors du i-ème inventaire s'est produite et est égal à 0 sinon ; et s_i est égal à 1 si une lecture réussie lors du i-ème inventaire s'est produite et est égal à 0 sinon.

[1] Ceci est courant. Une entreprise ou une entité d'une entreprise dont le travail est de fabriquer un produit spécifique recevra en permanence les mêmes matières premières et sortira toujours les mêmes produits

En effet, ce paramètre est fortement caractéristique des systèmes RFID [DER07a] : bien que la précision des systèmes RFID soit croissante, leurs taux de lectures durant un inventaire simple sont en moyenne de 60 à 70%. Il y a donc en général 30% d'erreurs de lecture : 30% des tags ne sont pas identifiés au cours d'un inventaire simple. Ainsi, il est possible d'étudier l'évolution de ce paramètre, renseignant sur la santé du système.

Comme la méthode précédente, il faut effectuer une analyse statistique avant de pouvoir l'utiliser. Cette analyse sera donc dépendante de l'application utilisant la RFID. Elle consiste à effectuer une analyse du nombre d'erreurs de lecture sur le nombre total de lectures en fonction du temps. Ainsi, en fonction de l'application, on pourra avoir un RETR constant quel que soit le moment de la journée, ou alors, comme pour l'ATTV, dépendant du moment de la journée, voire de la semaine. Il n'est donc pas toujours possible de détecter des erreurs au sein des tags. Il est seulement possible, de cette manière, de détecter la présence de fautes graves présentes sur une longue durée – durée égale à l'intervalle d'observation.

En reprenant l'application décrite au paragraphe précédent et illustrée par la Figure II-18, il est alors possible d'effectuer le calcul du RETR par palette, permettant ainsi de détecter la présence d'erreurs au niveau d'une palette – erreurs présentes soit sur un ou plusieurs tags, sur le lecteur ou dans l'environnement.

Contraintes de route comme technique de monitoring

[INO04] propose ainsi de déclarer des contraintes de route. Ces contraintes de route correspondent aux différents chemins que peuvent prendre les tags dans le système. La Figure II-19 donne un exemple de contraintes de route pour une application de gestion de magasin. Dans le magasin se trouvent 8 points de lecture différents regroupés en 3 zones :

1. les étagères – ronds avec des points jaunes ;
2. les caissiers – ronds avec des vagues verts ;
3. et les sorties – ronds avec des traits bleues.

Les routes possibles permettent donc :

- des déplacements entre les étagères ;
- des déplacements depuis les étagères vers un des caissiers ;
- et des déplacements depuis les caissiers vers une des sorties.

Tous les autres déplacements ne sont pas autorisés. En effet, de par la construction de l'application, il est impossible qu'un tag se déplace d'une étagère vers une sortie, ou bien encore d'une sortie vers une étagère. Dans un cas comme dans l'autre, qu'il s'agisse d'une erreur humaine ou d'une erreur intrinsèque au système, le middleware lèvera une exception afin d'avertir les utilisateurs de l'apparition de l'erreur.

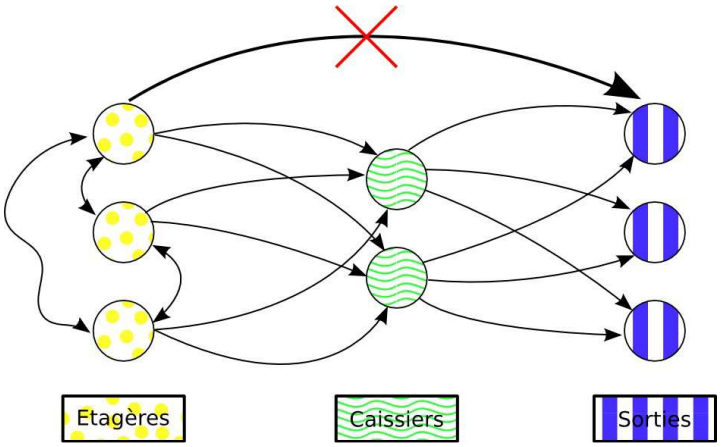


Figure II-19 : Exemple de contraintes de route [INO04]

L'inconvénient de cette méthode de test en ligne est qu'il faut bien connaître le système afin de permettre de définir les routes possibles. De plus, il faut aussi que les points de lectures soient géographiquement proches afin que le middleware puisse traiter les informations en temps réel.

II.3 Simulateurs RFID

Nous venons de voir, au travers de la section précédente, qu'il était nécessaire d'étudier la robustesse des systèmes RFID. En effet, ce sont des systèmes complexes qui sont utilisés à des fins de sécurisation ou d'optimisation de processus.

Afin d'étudier la sûreté de fonctionnement de ces systèmes, un simulateur est nécessaire pour :

1. faciliter l'analyse du système dans son ensemble,
2. étudier l'impact et l'effet des fautes locales sur le système complet,
3. évaluer l'efficacité des approches visant à la détection de ces erreurs.

Nous nous intéressons donc particulièrement aux simulateurs permettant la simulation des différents éléments du système RFID afin de simuler le comportement du système dans son ensemble.

Il existe de nombreux simulateurs RFID simulant une ou plusieurs parties des systèmes RFID : par exemple, simulant le comportement radiofréquence du tag, le fonctionnement interne de la puce du tag, la communication entre le tag et le lecteur ou le fonctionnement du lecteur. Nous ne nous intéresserons pas à la première catégorie de simulateur qui traite le comportement radiofréquence du tag. En effet, la simulation du comportement RF est longue et très exigeante en précision. Elle prend en compte beaucoup de paramètres de bas niveau tels que la géométrie de l'antenne du tag : ces simulations sont donc très coûteuses en temps. De plus, notre objectif est d'étudier la globalité du système. Ainsi ces simulateurs ne permettent pas l'observation des interactions entres tags, ou entre tags et lecteurs au niveau protocole. Il n'est pas non plus possible d'y injecter certains types de fautes tels qu'une erreur dans la logique du tag et d'observer leurs effets au niveau de chaque composant et au niveau du système complet.

Dans la suite du document, nous présentons donc les différents simulateurs déjà existants. En premier lieu, les simulateurs représentant le fonctionnement d'un tag sont présentés. Puis les simulateurs permettant l'évaluation du protocole de communication entre le tag et le lecteur sont étudiés. Enfin, nous présentons les simulateurs spécialisés dans la simulation du comportement du lecteur vis-à-vis du middleware.

II.3.1 Simulateurs de tags

Les simulateurs de tags reposent souvent sur des plateformes de prototypage rapide. Néanmoins, les tags sont décrits dans des langages permettant aussi la simulation.

[DER07b] propose une modélisation de tag UHF via une description sous Matlab Simulink®. Cette description est ensuite traduite en C/C++ afin d'être implantée sur carte DSP. Ainsi, il sera possible de proposer et valider de nouvelles fonctionnalités ou bien des modifications dans le protocole anticollision. La Figure II-20 montre cette plateforme.

[WAN09] quant à lui, propose une plateforme de simulation permettant la validation du fonctionnement d'un lecteur et d'un middleware en environnement réel. Pour cela, il intègre plusieurs tags sur une carte FPGA, permettant ainsi d'augmenter artificiellement le nombre de tags présents devant un lecteur et de valider en environnement réel de nouvelles fonctionnalités ou des modifications dans le protocole anticollision.

Néanmoins, ces descriptions ne permettent pas la simulation du système complet. En effet, elles ont été conçues dans un but précis : la validation de fonctionnalités intégrées aux tags. Les langages de descriptions utilisés et les architectures de ces simulateurs ont été choisis et conçus dans ce but.

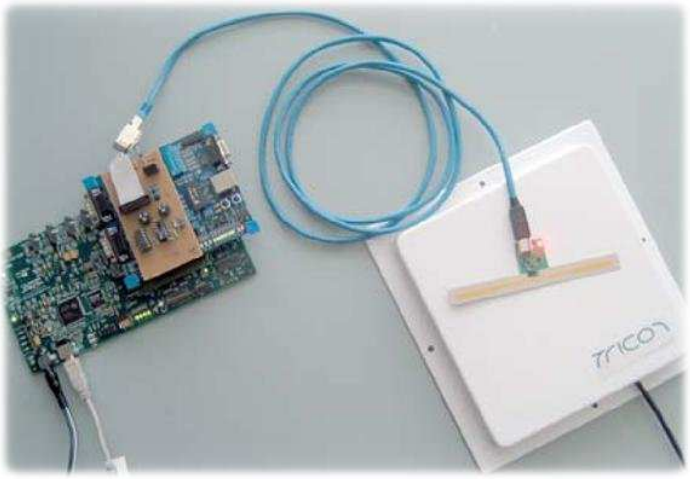


Figure II-20 : Plateforme de simulation de tags RFID [DER07b]

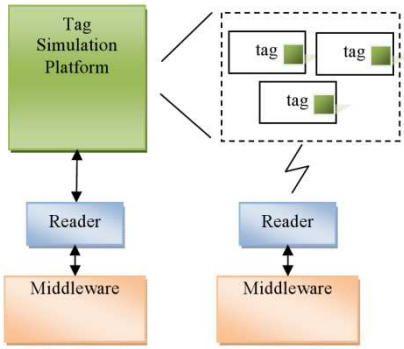


Figure II-21 : Plateforme de simulation [WAN09]

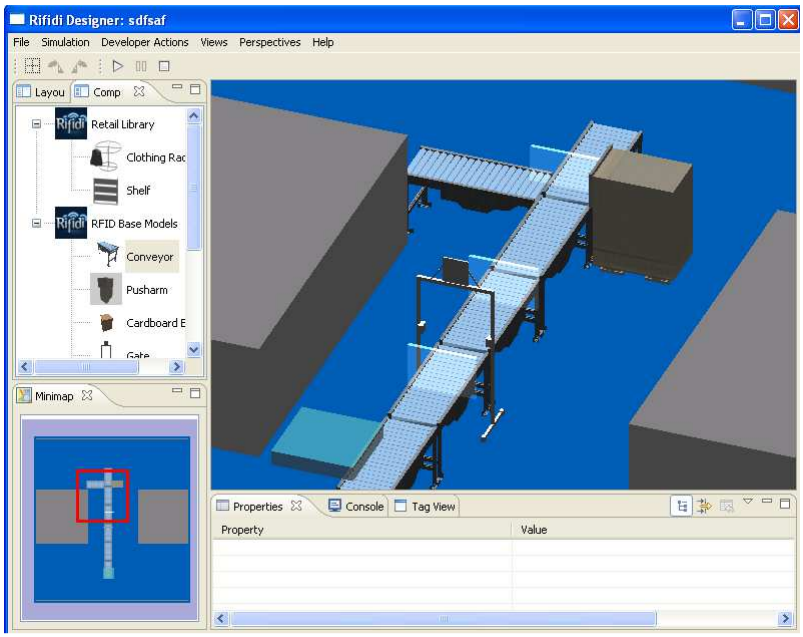


Figure II-22 : Exemple d'utilisation de RIFIDI [RIFIDI]

II.3.2 Simulateurs de lecteurs

Ces simulateurs ont pour objectif principal de générer des données à destination du middleware. Ces données sont principalement des listes de tags détectés associés à la date et l'heure de détection. [FOSSTRAK], [RIFIDI] et [MIR09] proposent ainsi de simuler des lecteurs. Ceux-ci vont alors générer une grande quantité de données à destination des middlewares. Leurs objectifs sont divers : valider le fonctionnement d'un middleware, tester la capacité de charge d'un système, vérifier la capacité à détecter des données erronées, … La Figure II-22 donne un exemple du simulateur **RIFIDI**. A l'aide des différents outils fournis, il est possible de modéliser les différents points de lecture et les chemins possibles des différents tags. Il est alors possible d'établir des scénarios de tests qui simuleront un système RFID.

Le fonctionnement des tags, la communication entre les lecteurs et les tags ainsi que le fonctionnement interne des tags ne sont pas modélisés ; seuls les flux de données générés par les lecteurs sont modélisés. Ces outils sont généralement réalisés avec des langages de programmation haut niveau tel que Java. Cela rend difficile l'ajout de modèles de composants matériels.

II.3.3 Simulateurs de protocole tag/lecteur

Ces simulateurs ont pour objectif la modélisation du lien existant entre un tag et un lecteur. Ils peuvent être divisés en deux catégories :

1. les simulateurs du protocole logique. Ces simulateurs vont s'attacher à simuler le fonctionnement logique du lien, c'est-à-dire les commandes échangées entre le tag et le lecteur et les différentes étapes de la communication – par exemple le protocole anticollision ;
2. les simulateurs du protocole physique. Ces simulateurs vont s'attacher à simuler le fonctionnement physique du lien, c'est-à-dire la création du signal analogique comportant les données, son envoi, son transport, sa réception et son décodage.

La première catégorie de simulateurs, dont [CHE07], [PETRA] et [WOE09] sont des représentants, va offrir à l'utilisateur la possibilité d'évaluer les différents paramètres logiques de la communication, principalement ceux touchant le protocole anticollision. Ils vont

permettre de simuler, en fonction de leurs spécifications, des algorithmes anticollisions utilisés en HF et/ou UHF. Ainsi, il sera possible d'évaluer la pertinence du choix d'un algorithme anticollision en fonction du nombre de tags ou encore de préconfigurer ces algorithmes par rapport à l'estimation faite du nombre de tags et de leurs identifiants. Ces outils sont développés avec des langages de programmation hauts niveau, tel que Java, C# ou encore Matlab Simulink®.

La seconde catégorie de simulateurs va s'attacher à simuler le lien physique permettant aux tags et aux lecteurs de communiquer. Il existe beaucoup de simulateurs de lien physique [ANG09, HAN05, JIN06, KHO07, SOF07]. Chacun de ces simulateurs permet de simuler avec précision un des éléments du lien physique entre le lecteur et le tag. Ainsi, certains simuleront le lien HF, d'autres le lien UHF. Certains s'attacheront particulièrement à la modélisation de la modulation de charge ou de la rétro-modulation, ou de la récupération et de l'analyse du signal, de sa démodulation ; d'autres encore s'attacheront au canal de transmission. On peut voir par exemple sur la Figure II-23 un exemple d'un de ces simulateurs. Sur cet exemple, toute la chaîne de transmission est modélisée dans le sens tag vers lecteur : l'envoi de l'onde porteuse par le lecteur, la modulation du signal et l'envoi par rétro-modulation, le parcours du canal et la récupération et la démodulation par le lecteur. Ce simulateur utilisant une modélisation fine du canal va aussi permettre d'étudier le taux d'erreur binaire en fonction du bruit et des paramètres du signal. Ces simulateurs sont en général développés avec des langages dédiés à la modélisation de signaux analogiques : Verilog-A, VHDL-AMS ou encore Matlab Simulink®.

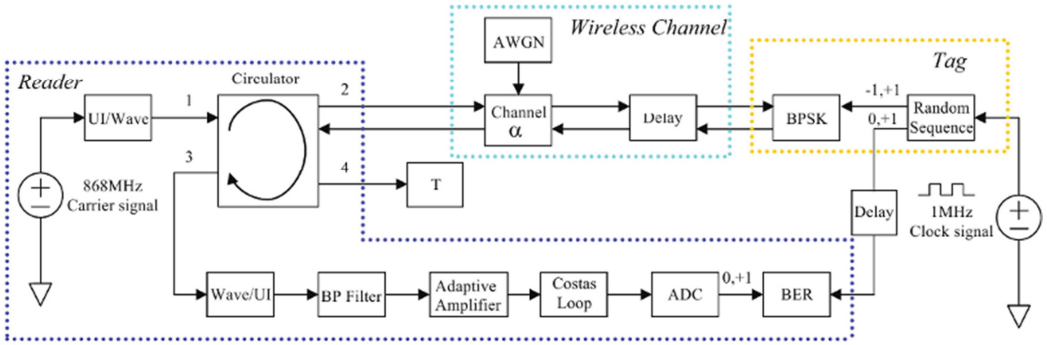


Figure II-23 : Diagramme blocs du simulateur [SOF07] simulant la chaîne d'émission, la chaîne de réception et le canal de transmission

II.3.4 Simulateurs de systèmes RFID

Ces simulateurs vont s'attacher à simuler la plus grande partie possible d'un système RFID. Ils sont peu nombreux. Deux simulateurs de systèmes RFID se distinguent particulièrement. Le premier, décrit dans [ANG07], va s'attacher à simuler la totalité du système, depuis les signaux analogiques de communication entre le tag et le lecteur, jusqu'au fonctionnement du lecteur. Néanmoins, ce simulateur souffre de trois principales limites : la première est l'absence de connexion avec un middleware, le résultat des simulations servant juste à la vérification du bon fonctionnement du système ; la seconde est la limitation du nombre de tags pouvant être simulé : en effet, les phénomènes de collision et de masquage décrit au paragraphe II.1.3.1 ne sont pas modélisés ; enfin, comme le montre la Figure II-24, la simulation se fait en deux étapes : la première partie de la simulation se fait dans un environnement donné, ici SystemC et C++, générant alors des traces d'exécution ; la seconde partie de la simulation se fait ensuite dans Matlab Simulink® à partir des traces d'exécution de la première partie. Cette double simulation empêche d'observer les interactions existantes entres les deux parties.

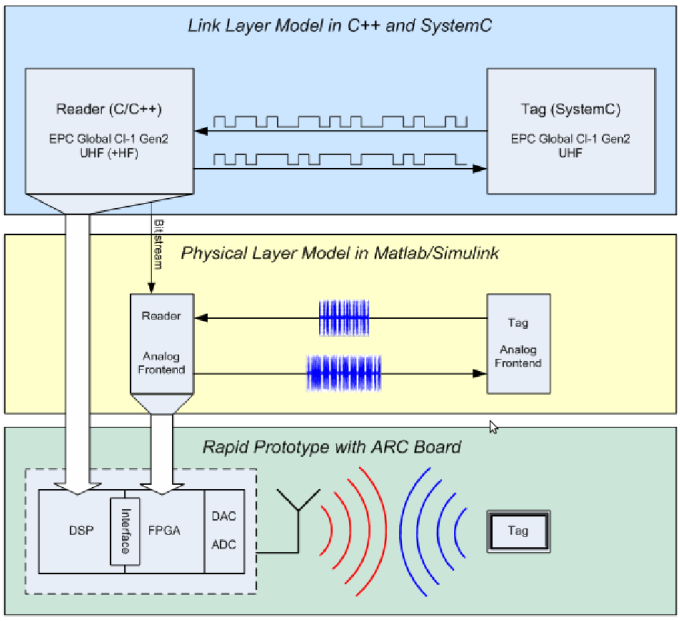


Figure II-24 : Architecture du simulateur [ANG07]

RFIDSim est un autre simulateur de systèmes RFID UHF [FLO08, FLO09], plus complet que le précedent. Il a été développé au sein de l'AutoIDlab, regroupement de laboratoires travaillant sur les différents aspects de la RFID. L'objectif de ce simulateur est l'évaluation des protocoles physiques et logiques pour la RFID dans des systèmes réels en simulant notamment la propagation du signal, le protocole de communication et le mouvement des tags. Il est développé en Java à partir du simulateur d'événements discrets JiST. Il simule uniquement des systèmes UHF et implémente actuellement la norme EPC Classe 1 Génération 2 (ISO1800-6C). Ce simulateur est composé de trois parties : le lecteur, le tag et la transmission du signal.

La partie logique du lecteur implémente la totalité des commandes du protocole EPC Classe 1 Génération 2 décrit au paragraphe 0. Elle implémente aussi différentes méthodes d'inventaire, *i.e.* différentes séquences utilisant les commandes *Query*, *QueryRep*, *QueryAdjust* et *Ack*. La partie analogique de l'émetteur du lecteur est caractérisée par la puissance transmise et la fréquence de la porteuse. Les différents temps de communication et débits spécifiés dans la norme sont implantés. La partie analogique du récepteur du tag est caractérisée par sa sensibilité minimum et par son modèle de capture. L'effet de capture est le phénomène des communications sans-fil qui permet le décodage correct d'un signal bien que d'autres soient arrivés en même temps. Un signal est correctement capturé si sa puissance est bien supérieure à celle des autres signaux. Les erreurs de communication sont modélisées ici : en fonction du rapport signal sur bruit du signal reçu, le lecteur détermine s'il est capable ou non de décoder le message.

La partie logique du tag implémente la totalité des commandes du protocole EPC Classe 1 Génération 2. La partie analogique du récepteur du tag a les mêmes caractéristiques que la partie analogique du récepteur du lecteur. La partie analogique de l'émetteur du tag modélise le phénomène de rétro-modulation propre à la RFID.

Enfin la partie transmission du signal est caractérisée par l'atténuation du signal en fonction de la distance. Plusieurs modèles peuvent être implantés. Actuellement, le modèle d'atténuation utilisé est le plus courant : la puissance est atténuée en fonction d'un facteur dépendant de la distance, de la longueur d'onde de la porteuse et d'un exposant d'atténuation caractéristique du milieu de propagation. De plus, cette partie modélise aussi les phénomènes de chemins multiples du signal par la distribution de *Rician*.

RFIDSim est capable de simuler des scénarii complexes : grand nombre de tags, tags en mouvement, inventaires, écriture de données dans le tag, désactivation complète du tag (commande KILL), *etc.* Le résultat de la simulation est accessible uniquement à la fin de la simulation. Il est composé d'un compte-rendu détaillé sur les statistiques de la communication : débit moyen, distribution des puissances des signaux reçus, taux d'identification global et d'erreurs de lecture, d'écriture et de « KILL ».

Les avantages de ce simulateur vis-à-vis de l'analyse de la sûreté de fonctionnement des systèmes RFID sont :

1. Simulation de plusieurs tags ;
2. Simulation du système complet ;
3. Simulation de scénarii complexes ;
4. Utilisation du protocole EPC Class 1 Gen 2 ;
5. Modélisation haut niveau qui permet une rapidité de simulation.

Les inconvénients de ce simulateur vis-à-vis de l'analyse de la sûreté de fonctionnement des systèmes RFID sont :

1. Pas de communication avec un middleware ;
2. Utilisation d'un langage de haut niveau qui rend impossible la simulation des composants physiques ;
3. Non prévu pour l'analyse de robustesse, ni pour l'injection de fautes.

Le Tableau II-II résume les différentes caractéristiques des simulateurs présentés ici. On peut ainsi voir qu'aucun simulateur présenté ne permet la simulation complète du système et l'observation des composants internes au système.

II.4 Conclusion

Au cours de ce chapitre, les systèmes RFID ont étés présentés. Il est ainsi apparu clairement que ces systèmes sont complexes et hétérogènes du fait des objectifs et de la composition très différents des éléments qui les composent : le tag, élément de petite dimension et de faible coût, destiné à être embarqué avec l'objet ou la personne à identifier ; le lecteur, équipement électronique capable d'identifier plusieurs tags en même temps ; et le logiciel dont le rôle est de gérer la grande quantité de données issues des différents points de lecture. La caractéristique principale de la RFID repose dans le mode de fonctionnement du tag : cet élément, conçu avec l'objectif de réduire le coût de production ainsi que le nombre de manipulations par des opérateurs au maximum, ne possède pas de source d'énergie ni d'émetteur RF à proprement parler : il utilise l'onde émise par le lecteur afin de s'alimenter et de transmettre les données qu'il contient.

La sûreté de fonctionnement a été présentée. Elle consiste en l'étude de différents attributs la caractérisant : la fiabilité, la disponibilité, la sécurité-innocuité, la sécurité-imunité. Ces différents attributs interdépendants permettent d'évaluer la confiance que l'on peut avoir dans un système. Des méthodes permettant d'améliorer ces attributs ont été évoqués. La sûreté de fonctionnement des systèmes RFID a été abordée et différentes méthodes permettant de l'améliorer ont été exposées. Deux méthodes ont particulièrement été abordées. Ces deux méthodes sont des tests en ligne non intrusifs permettant la détection de l'apparition de dysfonctionnement dans le système à partir de l'observation des paramètres de performance, tels que le nombre d'erreurs de lecture ou la quantité de tags passant par unité de temps. Il est en effet possible de détecter une baisse de performance caractéristique de l'apparition d'un dysfonctionnement dans le fonctionnement du lecteur ou du processus contrôlé par le système RFID.

Finalement, différents simulateurs ont été présentés dans la dernière section. Ces simulateurs permettent l'analyse de l'impact de modification du design des tags, des lecteurs ou du middleware sur le système. Néanmoins, chaque simulateur a été développé dans un but précis : soit le prototypage rapide des tags, soit l'analyse et l'amélioration des algorithmes anticollisions ou encore le design des middlewares. Aucun simulateur n'est approprié à l'injection de fautes en vue de l'évaluation des techniques de test en ligne.

Au cours du chapitre suivant, l'analyse de la sûreté de fonctionnement des systèmes RFID est tout d'abord réalisée. Puis, un simulateur est présenté, permettant la simulation d'un système RFID sain dans son ensemble et l'injection de fautes dans celui-ci afin d'observer leurs impacts sur le système complet. Il permet ainsi d'étudier les capacités des méthodes de test en ligne afin de détecter la présence de défaillances dans le système.

Tableau II-II : Synthèse des capacités des différents simulateurs

	Langage ou environnement utilisé	Catégorie de RFID	Normes utilisées	Description			Lien avec un middleware	Avantages	Inconvénients
				tag	protocole physique	protocole logique / lecteur			
[ANG07]	SystemC/C/C++ &Matlab Simulink® DSP & FPGA	UHF	ISO-18000-6C	SystemC	Matlab Simulink®	C/C++		- Description détaillée du tag et du lecteur - Prise en compte du canal de transmission - Possibilité d'injecter des fautes matérielles	- Simulation séparée de l'analogique et du numérique - Pas de lien avec le middleware - Tests des fautes matérielles spécifiques à cette description
[ANG09]	DSP & FPGA	UHF	ISO-18000-6C		Matlab Simulink®	C++		- Description détaillée du tag et du lecteur - Prise en compte du canal de transmission - Possibilité d'injecter des fautes matérielles	- Simulation séparée de l'analogique et du numérique - Pas de lien avec le middleware - Tests des fautes matérielles spécifiques à cette description
[CHE07]	C# .NET	HF & UHF	ISO-18000-6A, 6B et 6C			X		- Description de plusieurs algorithmes anticollision - HF et UHF	- Pas de modélisation du tag, du lecteur - Pas de lien avec le middleware
[DER07]	DSP	UHF	ISO-18000-6C	Matlab Simulink®				- Description implantable du tag - Possibilité d'injection de fautes matérielles	- Pas de description du lecteur et du canal de transmission - Pas de lien avec le middleware - Tests des fautes matérielles spécifiques à cette description
[FLO08] **[FLO09]**	Java – JiST	UHF	ISO-18000-6C	X	X	X		- Simulation complète du système	- Pas de lien avec le middleware - Impossibilité d'injection de fautes matérielles - Utilisation d'un langage haut-niveau
[HAN05]	Matlab Simulink®	UHF	ISO-18000-6C		X			- Représentation détaillée du canal de transmission - Injection possible de fautes physiques	- Pas de description du tag, du lecteur et du protocole anticollision - Pas de lien avec le middleware

	Langage ou environnement utilisé	Catégorie de RFID	Normes utilisées	Description			Lien avec un middleware	Avantages	Inconvénients
[MIR09]	Java	UHF	ISO-18000-6C	Modèle logique		Modèle logique	X	- Une grande partie du système est simulée - Lien avec le middleware	- Pas de description du canal de transmission - Utilisation d'un langage haut-niveau
[JIN06]	Matlab Simulink®	UHF	ISO-18000-6B et 6C		Modulation côté lecteur et propagation			- Représentation détaillée du canal de transmission - Injection possible de fautes physiques	- Pas de description du tag, du lecteur et du protocole anticollision -Pas de lien avec le middleware
[PETRA]	Java	HF	ISO-18000-3			X		-Simulation de la présence de plusieurs tags en face du lecteur	-Pas de modélisation du tag et du canal de transmission -Pas de lien avec le middleware -Utilisation d'un langage haut-niveau
[SOF07]	Verilog-A	HF			Modulation de charge			-Modèle détaillé du canal et du signal transmis entre le tag et le lecteur -Utilisation d'un HDL	-Pas de modélisation du lecteur et du tag -Pas de lien avec le middleware
		UHF			Modulation BPSK				
[RIFIDI]	Java	UHF	EPC LLRP				X	-Simule la présence d'un grand nombre de tags et de lecteurs	-Pas de description du tag, du lecteur, du canal de transmission
[WAN09]	Exécution : DSP & FPGA	UHF	ISO-18000-6B et 6C	Très précise				-Simule plusieurs tags -Injection de fautes matérielles possibles	-Pas de description du canal de transmission ni du lecteur -Pas de lien avec le middleware
[WOE09]	Matlab	UHF	ISO-18000-6C			X		-Description mathématique -Observation de paramètres internes	-Pas de description du tag, du lecteur, ni du canal de transmission -Pas de lien avec le middleware

III Simulation pour l'évaluation de la robustesse des systèmes RFID

Le chapitre précédent présente le fonctionnement des systèmes RFID, différentes méthodes permettant d'assurer la robustesse des systèmes RFID, ainsi que les simulateurs permettant de simuler une partie des systèmes RFID afin d'évaluer cette robustesse localement. Néanmoins, il est nécessaire d'avoir un outil permettant l'évaluation de la qualité de ces méthodes de test et offrant la possibilité de les comparer au niveau du système.

Dans de ce chapitre, la sûreté de fonctionnement des systèmes RFID est étudiée plus précisément au travers d'une analyse des modes de défaillances et de leurs effets (AMDE). Cette analyse est couramment utilisée afin de vérifier qu'un système est suffisamment fiable pour être commercialisé, et afin de proposer des méthodes permettant de détecter, éviter ou corriger les défaillances pouvant apparaître dans le système.

Après cette étude, le simulateur **SERFID** est présenté. **SERFID** permet de simuler un système RFID du tag jusqu'au lecteur et de s'interconnecter avec un middleware. Ainsi, un système RFID peut être simulé en une seule fois dans son ensemble : des signaux numériques internes aux tags jusqu'au middleware gérant la totalité des données issues des différents points de lecture, en passant par les signaux analogiques échangés entre les tags et les lecteurs ou encore le déroulement de l'algorithme anticollision. Ce simulateur permet aussi d'injecter les fautes identifiées au cours de l'AMDE dans le système afin d'observer leurs impact à tous les niveaux du système. Ce simulateur permet aussi d'évaluer la capacité de différentes méthodes à prévenir ou détecter l'apparition des défaillances.

III.1 AMDE d'un système RFID

Pour étudier la sûreté de fonctionnement de la RFID, une première étape est la réalisation d'une analyse des modes de défaillance et de leurs effets – AMDE. Cette analyse est « *une méthode inductive d'analyse de système utilisée pour l'étude systématique des causes et des effets des défaillances qui peuvent affecter les composants de ce système* » [NIE02]. Cette analyse s'inscrit dans le cadre de l'analyse de la sûreté des systèmes RFID. En effet, cette analyse, très répandue dans l'industrie, permet d'avoir une vue d'ensemble des dysfonctionnements d'un système ainsi qu'une idée de leur impact sur le système. Ces dysfonctionnements sont dus à des défaillances. Une défaillance est « un événement survenant lorsque le service délivré dévie de l'accomplissement de la fonction du système » [LAP96]. Le système ne délivre alors plus le service attendu. Toute défaillance a été entraînée par un événement non prévu appelé « *cause possible de défaillance* ». Les dysfonctionnements sont les conséquences observables de la défaillance au niveau du système : ils sont alors appelés « *les effets sur le système* ». Les défaillances peuvent êtres transitoires, périodiques ou permanentes. Les défaillances transitoires peuvent apparaître à n'importe quel instant pour une durée variable puis disparaître. Les défaillances périodiques surviennent à intervalles de temps considérés comme constants pour une durée variable avant de se résorber. Finalement les défaillances permanentes perdurent jusqu'à une intervention extérieure après leur apparition.

La première étape de cette analyse est l'étude du système et sa décomposition en composants. En effet, [LAP96] définit un système comme étant « *un ensemble de composants interconnectés en vue d'interagir* ». [LAP96] précise aussi qu'« *un composant est un autre système* ». Il est alors possible d'effectuer cette analyse sur plusieurs niveaux. Il est aussi

possible d'effectuer cette analyse pour différents découpages du même système, changeant alors le point de vue sur le système. Comme le système reste le même, les différentes analyses auront des points communs ; la plupart des défaillances, de leurs causes et de leurs effets dans une analyse seront présents dans l'autre analyse sous une forme différente. Néanmoins, chaque analyse, focalisée différemment sur le système, aura des spécificités propres et une mise en forme propre.

Le premier découpage que nous proposons du système RFID est un découpage en fonction du modèle ISO/OSI [ISO7498-1] – de l'anglais « Open Systems Interconnection ». Le modèle ISO/OSI permet de décrire de manière homogène les éléments permettant la communication entre différents systèmes communicants. Il est donc adapté à la RFID et à la description des éléments la composant. Ce modèle représente les différentes étapes de la communication en 7 couches ; les données à envoyer passent alors d'une couche à l'autre afin d'être transmises au destinataire. Une couche ne peut communiquer qu'avec les couches directement adjacentes, inférieures ou supérieures, à condition qu'elles existent. Une fois arrivées au destinataire, ces données repasseront dans les différentes couches, en sens inverse, afin d'être délivrées à l'application.

Les communications existant entre les tags et les lecteurs sont assez simples, au regard du modèle ISO/OSI et des fonctionnalités de chacune des couches. Ainsi, toutes les couches ne sont pas utilisées car leurs fonctionnalités n'ont aucun sens pour les communications entre les tags et les lecteurs [FIN03]. Par conséquent, il est possible de simplifier le modèle à 3 couches – Figure III-1 :

1. la couche *physique (1)* : cette couche est chargée de la transmission effective des signaux entre les interlocuteurs à travers le *médium*. Son service est limité à l'émission et la réception d'un bit ou d'un train de bits continu. C'est dans cette couche que sont définis le codage bit, le support de transmission, les niveaux des signaux électriques ou radiofréquences, la modulation, la synchronisation au niveau bit, celle permettant la détection des bits dans le signal analogique ; ici, ce sont les composants permettant d'envoyer les signaux radiofréquences dans l'air.
2. la couche *liaison (2)* : cette couche est chargée de la gestion des communications entre 2 machines adjacentes, directement reliées entre elles par un support physique. Ses tâches sont multiples allant de la gestion de l'accès à la couche physique au filtrage des messages en passant par la détection d'erreurs, les procédures de recouvrement d'erreurs, l'arbitrage des conflits au niveau bit, l'acquittement de réception de messages ou encore la signalisation d'erreurs dans une trame ; ici, cette couche s'occupe principalement de la vérification de l'absence d'erreurs dans la communication (CRC), de l'identification des trames et du tri de celles-ci en fonction du destinataire théorique du message.
3. la couche *application (7)* : cette couche est le point d'accès aux services réseaux générant et manipulant les informations utiles. Elle est donc l'interface entre les données utiles et les moyens de communication : elle analyse ici les différents messages provenant du lecteur, modifie son état si besoin et répond en conséquence.

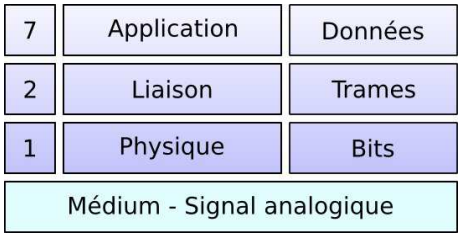


Figure III-1 : Modèle ISO/OSI appliqué à la RFID

Le second modèle est un découpage par fonctionnalité. Au cours du chapitre II, nous avons présenté les composants principaux de la RFID : le tag, le lecteur et le middleware. Néanmoins, ces composants regroupent, en leur sein, différentes fonctionnalités. De plus, utiliser une représentation aussi peu détaillée du système limiterait l'analyse. Comme indiqué plus haut, chaque composant peut aussi être considéré comme un système. Nous découpons donc chaque composant en sous-composants. Ces sous-composants correspondent aux fonctionnalités principales du tag [FIN03, PAR01, PAR08, PRE08].

La Figure III-2 illustre les composants du tag. Nous y retrouvons les composants classiques pour la communication :

- l'*antenne* pour recevoir les signaux et, dans le cas de la RFID UHF, réfléchir les signaux ;
- le *démodulateur* pour décoder les signaux reçus des lecteurs ;
- le *modulateur* pour venir modifier le signal réfléchi par l'antenne dans le cas de la RFID UHF, ou pour modifier la consommation du tag vu par le lecteur dans le cas de la RFID HF ;
- le *contrôleur CRC* permettant de vérifier les messages reçus et de signer les messages à envoyer.

Le tag est aussi composé d'éléments classiques tels que :

- la *mémoire* permettant de stocker les différentes informations associées à l'objet suivi ;
- la *logique de contrôle* permettant d'analyser les messages reçus des lecteurs et d'agir en fonction.

Nous retrouvons aussi un composant spécifique aux tags RFID passifs : le composant chargé d'extraire et mettre en forme la puissance issue de l'onde radiofréquence nécessaire au fonctionnement correct des autres composants.

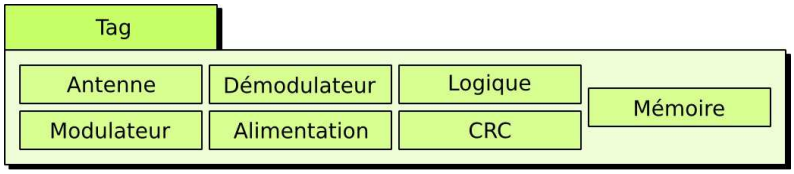


Figure III-2 : Découpage fonctionnel du tag

La Figure III-3 illustre les composants du lecteur. On y retrouve comme pour le tag les composants classiques pour la communication :

- l'*antenne* pour émettre les signaux permettant d'alimenter les tags et de leur envoyer des données et recevoir, dans le cas de la RFID UHF, les ondes réfléchies par les tags ;
- le *modulateur* pour moduler le signal émis par le lecteur afin que celui-ci puisse transporter les données à destination des tags ;
- le *démodulateur* pour décoder les informations transmises par les tags ;
- le *contrôleur CRC* permettant de vérifier les messages reçus et de signer les messages à envoyer.

On y retrouve enfin les composants nécessaires au fonctionnement du lecteur :

- la *logique* de contrôle, permettant de gérer les actions telles que l'identification des tags présents dans son champ, l'envoi de rapport d'identification à l'hôte, … ;

- le *lien avec l'hôte* permettant au lecteur de communiquer avec la station hôte, ou le middleware, afin de leur délivrer les informations qu'il contient.

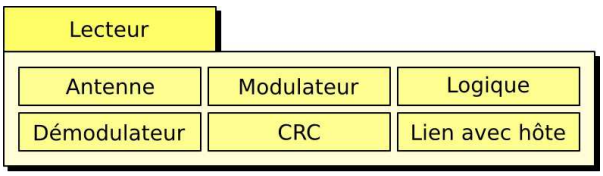


Figure III-3 : Découpage fonctionnel du lecteur

Comme signalé précédemment, il existe naturellement des liens entre ces deux modèles. La Figure III-4 montre ces liens. Ces deux modèles se recoupent uniquement sur les éléments portant sur la communication, ceci étant dû à la nature du modèle basé sur la description ISO/OSI. Ainsi, les composants liés à la gestion des signaux analogiques font partie de la couche *physique n°1* : ce sont l'antenne, le modulateur et le démodulateur. Les composants liés à la liaison entre deux éléments font parties de la couche *liaison n°2* : ce sont le contrôleur CRC et la partie de la logique de contrôle permettant la liaison entre deux équipements. Finalement les composants spécifiques à l'application qu'est la RFID font partie de la couche *application n°7* : ce sont la logique de contrôle et la mémoire. Ainsi, les modes de défaillances présents dans l'AMDE basée sur la description ISO/OSI du système sont présents, sous une forme différente, dans l'AMDE basée sur la description fonctionnelle du système. Il est donc possible d'ajouter à l'AMDE basée sur la description ISO/OSI l'emplacement possible de la défaillance dans la description basée sur la description fonctionnelle. Il est important néanmoins d'avoir ces deux modèles. Le premier modèle, basé sur la description ISO/OSI, décrit parfaitement les communications et les défaillances associées. Néanmoins, les autres fonctionnalités du système, celles servant à autre chose que la communication, ne sont pas prises en compte par cette description. Le second modèle, basé sur la description fonctionnelle, comble ce manque.

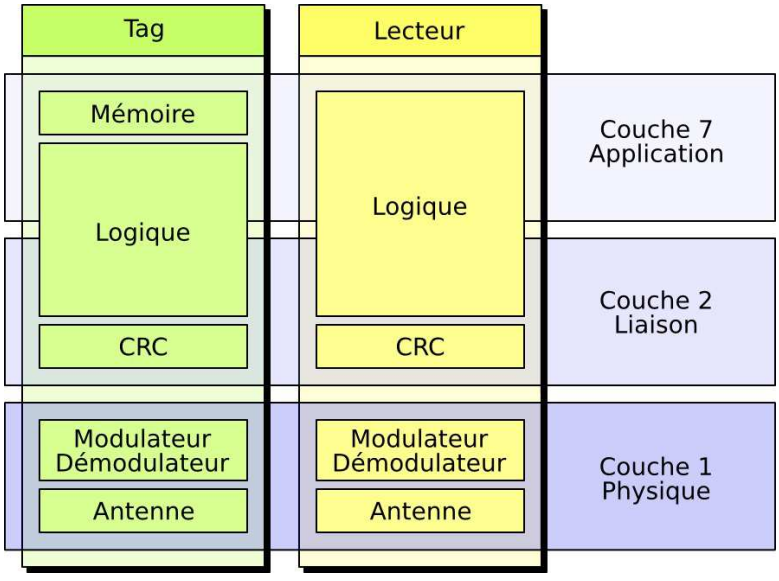


Figure III-4 : Relation entre le modèle OSI simplifié et les modèles comportementaux du tag et du lecteur

Le Tableau III-I est un extrait de l'AMDE basée sur la description ISO/OSI. La totalité des deux AMDE se trouve en annexe E.

La première colonne des AMDE correspond à l'élément pour lequel les défaillances vont être étudiées : pour l'AMDE basée sur la description ISO/OSI, cette colonne recense les

différentes couches ; pour l'AMDE basée sur la description fonctionnelle, cette colonne recense les différents composants. Dans l'extrait ci-dessous, nous nous intéressons donc à la couche OSI/ISO n°1 : la couche *physique*.

La seconde colonne liste les différents modes de défaillance. La première défaillance de la couche physique est la « non réception des signaux par un ou plusieurs tag(s) ou lecteur(s) ». Cela signifie qu'aucun signal n'est détecté au niveau des tags ou des lecteurs. Il y a plusieurs causes à cela.

Ces causes sont répertoriées dans la troisième colonne. Ainsi, une des causes possibles à la défaillance décrite précédemment est une « agression extérieure » des composants. Les agressions extérieures peuvent être dues à des causes naturelles telles que l'humidité ou la chaleur engendrant un vieillissement prématuré des composants. Mais ces agressions extérieures peuvent aussi être dues à des causes accidentelles : choc, coupure, déformation, …

La quatrième colonne, uniquement présente dans l'AMDE basée sur la description ISO/OSI, permet de faire le lien entre les deux AMDE : cette colonne liste les composants fonctionnels pouvant être à l'origine de la défaillance. Ainsi, lorsqu'un ou plusieurs tags ou lecteurs ne reçoivent pas de signaux à cause d'une agression extérieure, le composant de la couche physique ayant produit cette défaillance est probablement l'antenne ou la connexion entre l'antenne et la puce.

La cinquième colonne de l'AMDE tente de lister les conséquences possibles à cette défaillance. La conséquence principale de cette défaillance est ici la perte d'information. En effet, dans le cas où un ou plusieurs tags ne recevraient pas de signal, ils leur seraient impossibles d'envoyer leur identifiant. Dans le cas où ce serait un ou plusieurs lecteurs qui ne recevraient pas de signal, alors il leurs serait impossible de réaliser des inventaires et donc de lister les tags présents dans leur champ respectif.

Ces effets peuvent être plus ou moins critiques en fonction de la manière dont apparait la défaillance : transitoire, périodique ou permanente. Dans le cas de la perte d'information, si l'erreur est transitoire ou périodique, il est tout à fait possible aux lecteurs de récupérer les données présentes sur les tags, à condition d'effectuer plusieurs tentatives et qu'au moins une des tentatives se produise lorsque la défaillance n'est pas présente. Ceci permet alors de masquer l'effet de la défaillance. Dans le cas des défaillances permanentes, il ne sera donc pas possible de récupérer les informations présentes dans les tags, rendant alors l'effet permanent aussi. L'autre effet sur le système est la baisse de ses performances : les tags sont détectés plus difficilement, à des distances plus faibles que dans un fonctionnement sans défaillances.

Les différentes méthodes de détection de ces défaillances se trouvent dans la sixième colonne. Cette colonne recense les différentes méthodes possibles ou utilisées permettant de détecter la présence de la défaillance correspondante. Dans notre exemple, le moyen de détecter l'absence de « *non réception de signal* » peut-être l'acquittement utilisé par la couche 2 du modèle ISO/OSI, la couche *liaison*. En effet, cette couche peut avoir comme fonction la gestion d'acquittement. Ainsi, en l'absence d'acquittement, il est possible de détecter l'absence de réception de signal. Ceci est prévu dans les deux normes RFID que nous avons étudiées pour la plupart des commandes pour les liaisons lecteur vers tags. Après l'envoi d'une commande, le lecteur attend alors une réponse spécifique du tag permettant de prouver que ce dernier a reçu correctement le message. Si le lecteur ne reçoit pas cet acquittement, il considère que le tag n'a pas reçu son message et qu'il n'a donc rien fait. Il arrive, dans certains cas rares, qu'il y ait aussi un mécanisme d'acquittement dans la communication tag vers lecteur, principalement lorsque le lecteur et le tag s'échangent une suite de messages. Ceci permet de vérifier que le message précédent est arrivé correctement et que l'échange de données peut continuer. Il est à noter que c'est un moyen de détection et non de diagnostic. Ainsi, un même moyen de détection peut être utilisé pour détecter plusieurs défaillances possibles : ce moyen de détection permet aussi de détecter la non émission de signaux.

Les trois dernières colonnes correspondent aux différentes techniques permettant de corriger, par recouvrement ou par compensation, ou d'éviter l'apparition de défaillances.

Les actions correctrices sont dans les colonnes sept et huit. La septième colonne liste les actions correctrices par recouvrement, c'est-à-dire les actions tentant de remettre dans un état sain le système. Ces actions sont visibles par l'utilisateur. Par exemple, pour la défaillance « non réception de signaux par un ou plusieurs tags ou lecteurs », une action correctrice par recouvrement est la répétition des trames par l'émetteur jusqu'au succès de la communication. Dans la huitième colonne, nous pouvons voir l'action correctrice par compensation, action qui n'est pas visible par l'utilisateur. Un simple exemple d'action de recouvrement par compensation est l'utilisation de code correcteur d'erreur dans la couche 2 du modèle ISO/OSI permettant de retrouver le message envoyé sans avoir à effectuer d'action particulière pour l'utilisateur. Ces actions correctrices sont principalement du ressort des concepteurs. En effet, bien que les normes précisent beaucoup d'éléments comme les différents messages possibles, la succession de messages pour des actions spécifiques ou encore les différents mécanismes de protection des données lors de transmissions, elles ne précisent pas l'enchaînement des différentes actions, ni le nombre d'équipements et l'architecture des composants. Elles laissent en plus des degrés de liberté aux concepteurs en leur offrant différentes possibilités de paramétrage : taille des frames, différents codages bit, organisation d'une partie de la mémoire, commandes spécifiques au constructeur …

Enfin, la dernière colonne donne les méthodes d'évitement pour les défaillances. Comme leur nom l'indique, ces méthodes permettent d'éviter l'apparition de la défaillance, ou du moins de limiter son apparition. Pour la défaillance étudiée dans ce paragraphe, pour la cause « agressions extérieures », il existe plusieurs méthodes d'évitement :

- la première méthode est la *redondance passive*. La redondance passive consiste à utiliser par exemple plusieurs antennes simultanément. Si une antenne est abimée, les autres antennes continueront d'émettre sans avoir à commuter d'un équipement à un autre ;
- la seconde méthode consiste à améliorer, lors de la conception et de la fabrication, la *qualité physique des composants* afin que ceux-ci soient plus robustes aux agressions extérieures ;
- enfin, la dernière méthode est l'utilisation de *précautions de manipulation*. Il s'agit ici de spécifier les conditions de fonctionnement optimales de l'application et de les respecter.

Cette dernière colonne est souvent dépendante des normes utilisées. Ainsi, en RFID, bien que les normes proposent plusieurs options les concernant, les différents protocoles physiques sont figés afin de garantir l'interopérabilité des équipements. Il n'est possible au concepteur que de choisir le paramètre le plus adapté à son application. En particulier, les différents niveaux de puissance sont sujets à des réglementations. Il n'est alors pas possible d'augmenter au-delà d'un certain niveau la puissance, alors que cela pourrait permettre de fiabiliser la communication.

Il est à noter que chaque élément d'une colonne peut être un cas particulier d'une des colonnes qui la précèdent ou alors commun à plusieurs éléments des colonnes précédentes. Ainsi, on voit par exemple dans la colonne 6 nommée « Moyen de détection » que le premier élément est spécifique à la première cause de la défaillance alors que le second élément est commun aux 3 dernières causes de cette même défaillance.

Tableau III-I : AMDE basée sur le modèle ISO/OSI focalisée sur la couche physique

Couche ISO/OSI	Mode de défaillance	Cause possible	Composant(s) pouvant être à l'origine de la défaillance	Effet sur le système	Moyen de détection	Action correctrice		Evitement
						recouvrement	compensation	
Couche Physique	Non réception des signaux par un ou plusieurs tag(s) ou lecteur(s)	Tag(s) en dehors du champ du lecteur alors qu'il devrait y être	Antenne du lecteur, Canal	Perte d'information, Dégradation des performances	Non détection de modification du champ HF, Acquittement au niveau 2, Analyse statistique	Répétition de la transmission jusqu'au succès	Redondance du tag	Utilisation de puissance plus forte, de protocole physique plus robuste
		Perturbations EM	Canal, Antennes		Utilisation d'acquittement au niveau 2, Analyse statistique			Utilisation de puissance plus forte, de protocole physique plus robuste
		Agression extérieure (déformation, coupure, chocs, âge)	Antennes					Prendre des précautions, Qualité physique, Redondance passive
		Défaillance interne	Antenne, Démodulateur, Canal, Alimentation					Redondance passive de la couche physique
	Non émission des signaux par un ou plusieurs tag(s) ou lecteur(s)	Défaillance interne	Antenne, Modulateur, Canal, Alimentation		Utilisation d'acquittement au niveau 2, Analyse statistique			

Couche ISO/OSI	Mode de défaillance	Cause possible	Composant(s) pouvant être à l'origine de la défaillance	Effet sur le système	Moyen de détection	Action correctrice		Evitement
		Tag(s) en dehors du champ du lecteur alors qu'il devrait y être	Antenne du lecteur, Canal		Utilisation d'acquittement au niveau 2, Analyse statistique			
		Agression extérieure (déformation, coupure, chocs, âge)	Antenne		Utilisation d'acquittement au niveau 2, Analyse statistique			
	Emission continue (dialogue excessif)	Défaillance interne du tag	Modulateur	Canal surchargé, perte d'information, Dégradation des performances	Analyse statistique	Désactivation automatique du tag		
						Retrait physique du tag du champ par une tierce partie		
		défaillance interne du lecteur	Démodulateur, Antenne		Analyse statistique	Désactivation automatique du lecteur		
						Retrait physique du lecteur par une tierce partie		
		Répétition continue de tentatives d'émission	Logique interne du tag ou du lecteur		Comptage des erreurs d'acquittement, Analyse statistique			Limitation personnalisée du nombre de tentatives d'émission

Cette analyse qualitative est subjective, longue et difficile. En effet, au vu de la complexité du système, il est difficile d'être totalement exhaustif. De plus, la prédiction des effets sur le système d'une défaillance reste limitée. Afin de pouvoir injecter des fautes dans le système et simuler des défaillances pour valider cette analyse, et valider les méthodes de test existantes ou futures, nous avons besoin d'un simulateur de systèmes RFID spécifique. L'état de l'art sur les simulateurs, présenté lors du chapitre II, a montré l'absence de simulateur qui permettrait de réaliser cette étude en lien avec la sûreté de fonctionnement. Aussi, la section suivante présente le modèle sur lequel est basé le simulateur que nous avons développé.

III.2 Modèle du système RFID

L'analyse préliminaire des modes de défaillance et de leurs effets a été présentée. Après cette étude, le besoin d'un simulateur est devenu clair. Ainsi, cette section va présenter le simulateur que nous avons développé durant cette thèse.

La première étape dans la conception et le développement de ce simulateur est de lister les fonctionnalités et les caractéristiques attendues de ce dernier. Puis, il est nécessaire de déterminer quel langage de programmation, de description ou environnement de simulation est le plus adapté à notre besoin. Enfin, le simulateur peut être conçu et développé. Cette section explique ces différentes étapes.

Deux versions du simulateur ont été développées : une première version, décrite ici, permettant de simuler un système RFID UHF et une seconde version permettant de simuler un système HF. Cette version de ce simulateur implémente la norme ISO-15693 [ISO15693-1, ISO15693-2, ISO15693-3] encadrant le fonctionnement de système RFID HF. Les phénomènes physiques, les protocoles et les différentes fonctionnalités modélisés dans le simulateur sont alors spécifiques à la RFID HF et présentés en annexe C et D.

III.2.1 Fonctionnalités et caractéristiques

L'objectif principal de ce simulateur est la simulation complète du système RFID UHF basée sur la norme EPC Classe 1 Génération 2 [EPCC1G2], allant du tag jusqu'au middleware afin d'étudier l'impact des défaillances matérielles ou logicielles sur l'ensemble du système.

La Figure III-5, déjà présentée dans la section II.1, représente la vue d'ensemble d'un système RFID. On peut ainsi voir que le middleware – composant logiciel à l'inverse des autres composants matériels – gère plusieurs lecteurs à travers un réseau. De plus, nous avons vu lors de la section II.1.4 que les fonctionnalités d'un middleware sont complexes et qu'il existe déjà beaucoup de middlewares, dont des middlewares Open Source. Il est donc tout à fait envisageable d'utiliser les fonctionnalités réseaux déjà présentes sur un ordinateur ainsi qu'un middleware existant. Ceci permettra d'avoir un fonctionnement proche de la réalité avec de très bons temps d'exécution. De plus, il sera tout de même possible d'injecter des fautes dans ces deux parties logicielles du système :

- dans la partie middleware, souvent développée en Java, des injections de fautes fonctionnelles identifiées durant l'AMDE ou des injections de fautes par mutation peuvent être envisagées ;
- dans la partie réseau, des injections de fautes peuvent être réalisées simplement en ajoutant aux différentes interconnexions des éléments de capture/filtrage et modification des messages [CAV08].

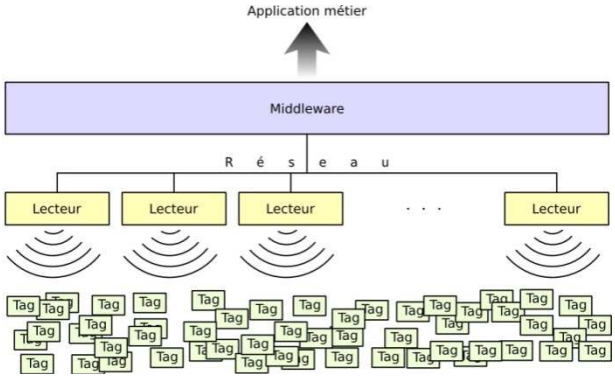


Figure III-5 : Vue globale d'un système RFID

Le simulateur dont nous avons besoin a en charge la simulation des lecteurs, des tags et des communications lecteurs/tags. Ces équipements ont la particularité de s'appuyer sur des composants logiciels et matériels, avec des signaux numériques et analogiques. Il est donc nécessaire d'utiliser un langage de description permettant de modéliser aussi bien les composants analogiques, numériques et logiciels. Comme le montre la Figure III-5, le simulateur doit aussi pouvoir simuler plusieurs lecteurs et de nombreux tags.

Nous souhaitons, dans un premier temps, observer le comportement du système en présence de défaillances fonctionnelles. Ainsi, le simulateur est conçu pour simuler le système avec une description haut niveau (fonctionnelle). Néanmoins, l'objectif du projet dans son ensemble est de proposer des modifications au sein du système afin d'améliorer la sûreté de fonctionnement. Ces modifications pourront être logicielles telles que des méthodes de test en ligne ou de nouveaux algorithmes anticollisions ; ou matérielles : modification de l'architecture du tag, utilisation d'autotest matériel, … Le simulateur devra donc dans le futur autoriser la simulation des descriptions synthétisables du tag ou du lecteur.

L'objectif du simulateur est la simulation complète du système. Il faut donc que les temps de simulations soient relativement proches des temps d'exécution du système physique. Cela permettra de pouvoir utiliser un middleware standard et de le dimensionner afin de l'utiliser dans un cas réel. Il est donc important que le langage utilisé n'engendre pas de temps de simulation prohibitif.

Ainsi, il faut choisir un langage qui permettra :

- de modéliser des aspects logiciels et matériels ;
- de modéliser des aspects numériques – simulations discrètes ;
- de modéliser des aspects analogiques – simulations continues ;
- de modéliser des effets aléatoires courants en RFID ;
- d'avoir des informations quant au comportement temporel du système ;
- de remplacer un composant par un autre ;
- de s'interfacer avec un middleware à travers un réseau ;
- de simuler le système avec des temps raisonnables ;
- de simuler des descriptions synthétisables en même temps que des descriptions fonctionnelles.

Il existe de nombreux environnements de simulation et de nombreux langages de programmation. Il est donc nécessaire d'étudier ces environnements et langages afin de choisir le langage qui nous permette de développer un simulateur répondant aux besoins que nous venons de lister.

III.2.2 Choix du langage de description

Il existe de nombreuses possibilités pour simuler les éléments d'un système : environnement de simulation, langage de description, de programmation, … Au vu de nos besoins, il est important d'étudier les différentes possibilités qui nous sont offertes afin de réaliser notre simulateur.

La première catégorie d'outils pour réaliser un simulateur est composée par les environnements de simulation. On retrouve par exemple, dans cette catégorie, des logiciels comme **CST** ou **MATLAB SIMULINK®**. **CST** est un logiciel permettant la simulation fine de phénomènes électromagnétiques dans un volume. La simulation des phénomènes physiques régissant les communications entre les tags et lecteurs sont modélisés avec beaucoup de précision. Ceci nécessite alors des temps de simulation importants. De plus, il est impossible de simuler les protocoles anticollisions, plusieurs lecteurs et tags, les composants internes des éléments, … Il n'est donc pas possible d'utiliser cet outil. **MATLAB SIMULINK®** permet quant à lui de simuler correctement tous les signaux analogiques et discrets. Néanmoins, il est difficile de simuler les algorithmes anticollisions, et des comportements logiciels complexes. La modélisation du parallélisme des tâches, nécessaire à la simulation des nombreux tags et lecteurs co-existants, n'est pas gérée. De plus, les temps de simulations peuvent être relativement longs, en fonction du niveau de détail souhaité.

La deuxième catégorie correspond aux langages de description matérielle tel que **VHDL**. Ce langage permet de décrire des architectures matérielles. Ainsi, il a été conçu pour réaliser la description, la simulation et la conception de systèmes numériques, permettant ainsi la description des signaux concurrents. De plus, en utilisant son extension nommée **VHDL-AMS**, il est possible de simuler aussi des phénomènes physiques, c'est-à-dire des signaux analogiques. Néanmoins son objectif premier, qui est la conception de circuit, lui impose certaines limites comme par exemple l'impossibilité de simuler facilement les parties logicielles ou les connexions à des programmes tiers à travers le réseau.

La troisième catégorie correspond aux langages de programmation classiques. On y retrouve par exemple les langages de programmations tels que le **C**, le **C++** ou encore le **JAVA**. Ces langages de programmation sont parfaits pour décrire les comportements complexes tels que les algorithmes anticollisions, les procédures d'inventaires, … De plus, il est simple de mettre en œuvre une connexion avec un autre programme à travers le réseau. Ils ont en général des temps d'exécution très faibles, en particulier le **C**. Les langages tels que **C++** et **JAVA** sont intrinsèquement performants pour la description de composants : ce sont des langages orientés objet et donc leur caractéristique principale est justement d'offrir cette notion de composants interchangeables facilement. Il n'est cependant pas directement possible de réaliser des simulations de signaux concurrents continus ou discrets. Il est aussi difficile de les interfacer avec des descriptions synthétisables. Ces langages seuls ne sont donc pas capables de décrire des signaux analogiques ni numériques. Il est donc difficile de réaliser des modélisations des processus concurrents ainsi que les échanges des signaux analogiques existants entre les différents composants sans l'utilisation de librairies complémentaires.

La quatrième et dernière catégorie est une extension de la catégorie précédente. Elle contient les bibliothèques spécifiques à la simulation à événements discrets ou à la simulation de signaux analogiques. On y retrouve par exemple **JiST**, **SystemC** ou encore **SystemC-AMS**. **JiST** est un environnement de simulation à événements discrets développé avec Java. Il permet ainsi de décrire des systèmes synchrones comme le ferait le langage de description **VHDL** en offrant en plus toutes les fonctionnalités du Java : l'implantation d'algorithmes complexes, la connexion au réseau… Une autre bibliothèque, cette fois développé en **C++**, a été conçue pour la simulation et la conception conjointe de matériel et logiciel. Cette bibliothèque s'appelle **SystemC**. Elle permet, comme le **VHDL**, la simulation de circuits

numériques, mais prend aussi toutes les caractéristiques du **C++**, permettant ainsi d'implanter des algorithmes complexes ou des connexions à travers le réseau. De plus, une seconde bibliothèque fonctionnant de pair avec **SystemC** et permettant la simulation de signaux analogiques a été proposée : **SystemC-AMS**. En utilisant ces deux bibliothèques, nous pouvons alors simuler les signaux numériques, analogiques et les parties logicielles du système.

Le Tableau III-II met en avant les différentes caractéristiques de langages et environnements de simulation que nous venons de voir.

Tableau III-II : Caractéristiques des langages et environnements permettant la simulation des systèmes RFID

Langage/ Environnement	Simu. continue	Simu. discrète	Simu. logiciel	Synthétisable	Conn. réseau	Notion de composants
C			X		X	
C++			X		X	X
CST	X					
Java			X		X	X
JiST		X	X		X	X
Matlab Simulink	X	X		X		X
SystemC*		X	X	X	X	X
SystemC-AMS*	X				X	X
VHDL**		X	Difficile	X		X
VHDL-AMS**	X					X

*,** : peuvent être utilisée conjointement

Ainsi, il apparaît clairement que l'utilisation de la bibliothèque **SystemC** permet la modélisation des parties analogiques, numériques et logicielles du système. En effet, **SystemC** ajoute, entre autres, au **C++**, les notions de signaux tels qu'on les connaît en **VHDL**. De plus, la notion de composant est intrinsèque au **C++**, langage utilisée pour développer et exploiter cette bibliothèque. Il est alors possible de multiplier les instances des descriptions réalisées pour permettre la simulation de plusieurs composants identiques : plusieurs tags ou lecteurs. Avec l'utilisation conjointe de la bibliothèque **SystemC-AMS**, il est aussi possible de décrire, avec précision des signaux analogiques.

À partir des spécificités propres à ce langage, des caractéristiques des systèmes RFID identifiées au cours du chapitre II et de l'AMDE présentée dans la section III.1, nous pouvons proposer une modélisation des systèmes RFID. La modélisation proposée est fonctionnelle, principalement centrée sur les transactions temporisées entre les différents composants. Une modélisation à ce niveau permet d'avoir des temps de simulations courts, en gardant néanmoins suffisamment de détails quant aux différentes fonctionnalités du système et aux temps nécessaires à chaque action. Ceci correspond bien à notre objectif : l'analyse des défaillances fonctionnelles du système, et à leurs effets globaux. Grâce aux caractéristiques intrinsèques du langage utilisé, il est possible de venir proposer dans le futur, une description plus détaillée des différents composants. Les sections suivantes décrivent la modélisation des différentes parties du système :

- le *tag*, contenant les données représentatives des objets associés ;
- le *lecteur*, élément actif venant récupérer les différentes données stockées dans les tags et transmettant le résultat au middleware ;
- les *communications* entre les tags et les lecteurs à travers le *canal* de communication, permettant l'échange des données entre les tags et les lecteurs.

Dans la suite du document, les sous-composants et leurs interactions sont définis pour chaque composant, ainsi que les connexions entre chaque composant principal.

III.2.3 Niveau de modélisation

L'objectif de ce simulateur est de rendre compte des phénomènes se produisant au sein des différents éléments du système et d'y injecter des fautes. La définition des fautes est réalisée à partir de l'AMDE présentée précédemment. De plus, on souhaite que les temps de simulation restent raisonnables par rapport aux temps nécessaires aux expérimentations équivalentes. Pour ces raisons, tous les composants et sous composants ont été décrits de manière fonctionnelle. Les interactions entre chaque sous composant et composant doivent pouvoir être observées.

Le modèle présenté par la suite est donc un modèle fonctionnel dit **TLM**, pour « *Transaction Level Model* » ou modèle au niveau transaction. Comme nous souhaitons aussi pouvoir observer les délais de chaque action, le modèle intègre une gestion du temps. Afin de faciliter son implémentation, chaque composant est responsable de son temps de traitement : ainsi lorsqu'il reçoit un message, il attend un certain temps avant de délivrer son message de réponse. La gestion du temps est donc distribuée au niveau de chaque composant : le modèle est alors **TLM-DT** pour « *Transaction Level Model with Distributed Time* » ou modèle au niveau transaction avec temps distribué.

III.2.4 Modélisation du tag

La Figure III-6 illustre les différents sous-composants du tag et leurs interactions. Dans certains cas particuliers, il est possible que des sous-composants soient remplacés par d'autres – cas du code correcteur d'erreur à la place du code détecteur d'erreur CRC – ou ajouté – cas des communications cryptées. Nous ne décrirons ici qu'une architecture classique de la RFID UHF.

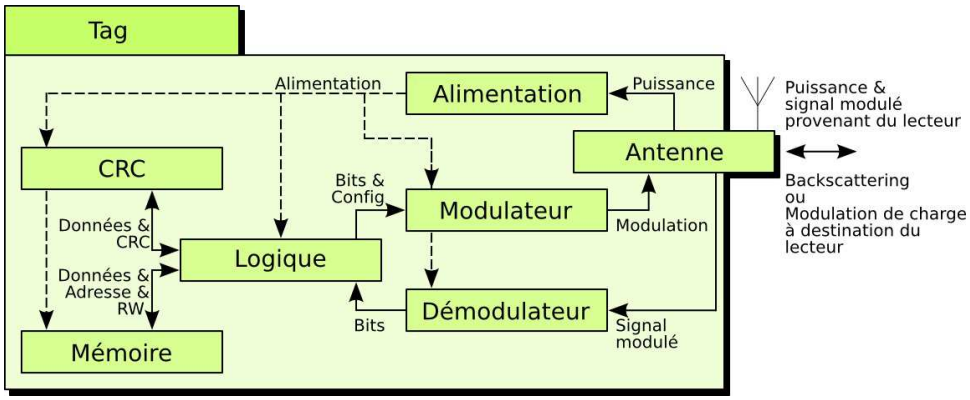


Figure III-6 : Modélisation du tag

On retrouve ainsi un sous-composant central : la *logique* de contrôle du tag. Ce composant est celui qui permet l'exécution des différents algorithmes nécessaires au fonctionnement du tag. Sa fonction principale est de décoder les messages provenant du lecteur. A partir de ce message, il effectue plusieurs actions :

1. réception du message provenant du sous-composant *démodulateur* ;
2. validation du message grâce au code CRC s'il est présent. Il réalise cette action en envoyant au sous-composant *CRC* les données et le code CRC du message reçu ;
3. récupération des données à des adresses précises dans le sous-composant *mémoire* si nécessaire ;
4. traitement et mise en forme des données récupérées ;
5. création d'une réponse si nécessaire et protection contre les erreurs de transmission par code CRC si besoin, en utilisant le sous-composant *CRC* ;

6. configuration du sous-composant *modulateur* en fonction des paramètres fournis par le lecteur – type de codage bit, débit binaire, modulation PSK/ASK, … ;
7. envoi du message grâce au sous-composant *modulateur* ;
8. modification de son état interne – endormi, en attente d'inventaire, inventorié, …

Le deuxième sous-composant central des tags RFID est le sous-composant *mémoire*. En effet, la raison d'être principale d'un tag est de stocker les données relatives à l'objet auquel il est attaché. Le principe de la mémoire est commun à toutes les normes : les données présentes dans la mémoire sont découpées en blocs de taille fixe et chaque bloc est stocké à une adresse dans une banque de données spécifique. Ainsi, pour créer, modifier ou récupérer un bloc, le sous-composant *logique* envoie à la mémoire la banque de données à utiliser et l'adresse du bloc ainsi que l'action à effectuer : lire ou écrire. Le nombre de banques de données, la taille des blocs et le nombre de blocs présents dans les banques de données sont variables en fonction des normes. Les normes définissent aussi les différentes données que peut contenir chaque banque de données. De plus, les normes définissent aussi les modes d'accès au banque de mémoire : lecture seule, écriture unique/lecture multiple, ou lecture et écriture. La norme UHF utilisée [EPCC1G2] spécifie 4 banques de données contenant chacune des données spécifiques dans lesquelles chaque bit est adressable.

Le dernier sous-composant spécifique à la RFID est le sous-composant *alimentation*. Ce composant est en charge d'extraire la puissance du signal provenant du lecteur, de la mettre en forme et de la stabiliser un maximum pour les autres composants. Ainsi, les autres composants peuvent fonctionner normalement et effectuer leurs différentes actions.

On retrouve ensuite les différents composants classiques pour les communications : le contrôleur CRC, le modulateur, le démodulateur et l'antenne.

Le sous-composant *CRC* a pour but de valider la bonne transmission des messages entre tags et lecteurs. Il a pour objectif de calculer le code CRC associé à un message et de vérifier la cohérence entre un message et le CRC associé[1].

Viennent finalement les sous-composants *démodulateur*, *modulateur* et *antenne*. Le premier est en charge d'extraire le message binaire codé dans le signal analogique reçu du lecteur via le sous-composant *antenne*. Il envoie ensuite le message décodé à la *logique* de contrôle. Le sous-composant *modulateur* est en charge d'encoder le message binaire généré par la logique de contrôle dans un signal qui sera à destination du lecteur. Pour cela, il utilisera les techniques décrites dans la section II.1.1 : en UHF, modulation de charge pour modifier le coefficient de réflexion lors de la rétro-modulation. Ces sous-composants sont spécifiques aux normes. En effet, les normes définissent les fréquences exactes, les modulations, les codages bits et les débits binaires que peuvent utiliser les lecteurs et tags pour pouvoir communiquer. Elles définissent aussi les signaux permettant aux différents acteurs de la communication de se synchroniser : signal indiquant le début d'une transmission ou la fin, la durée des bits envoyés, et donc le débit binaire. Par exemple, la norme EPC Classe 1 Génération 2 [EPCC1G2] précise que la modulation utilisée pour la liaison montante, *i.e.* la liaison du lecteur vers le tag, est une modulation ASK ou PR-ASK. L'encodage bit utilisée dans ce sens de la communication est le codage bit PIE. Il existe deux signaux de début de transmission dont un des deux permet une synchronisation et une calibration du tag pour la communication.

[1] Des exemples de procédures de calcul de CRC sont donnés en annexe I.1.

III.2.5 Modélisation du lecteur

SERFID permet d'utiliser plusieurs logiques différentes, en fonction des normes et permet aussi, pour une même norme, de tester différentes implantations. Il est alors possible d'étudier les points forts et les limites des différentes implantations, permettant de proposer des modifications à ces algorithmes et à leur utilisation afin de garantir à l'utilisateur un fonctionnement optimal et robuste. Dans la suite du document, la logique décrite est celle implanté : elle respecte la norme EPC Classe 1 Génération 2.

La Figure III-7 montre les différents sous-composants du lecteur et leurs interactions.

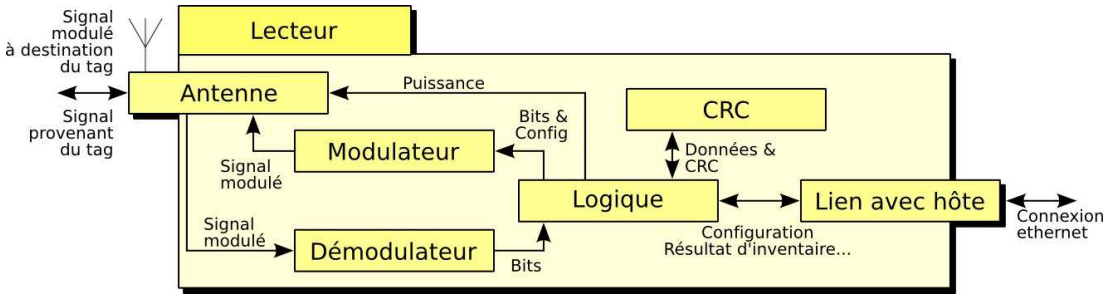


Figure III-7 : Modélisation du lecteur

On retrouve ici un sous-composant central : la *logique* de contrôle. Cette logique de contrôle, beaucoup plus complexe que celle du tag, a de nombreux rôles. Classiquement, cette logique de contrôle est un système complexe à base de microcontrôleur, voire de mini-PC. Les lecteurs commerciaux peuvent donc intégrer d'un simple micro-logiciel jusqu'à un système d'exploitation embarqué. En effet, cette logique de contrôle a beaucoup de fonctions. Sa fonction principale est d'*inventorier les tags* présents dans le champ du lecteur. Cette fonctionnalité est la raison d'être des lecteurs. Pour cela, le lecteur va exécuter l'algorithme anticollision prévu par les différentes normes[1]. Comme il a été vu, les algorithmes anticollision offrent beaucoup de possibilités. Il faut donc que le lecteur, aidé par la configuration de l'utilisateur, soit capable d'exploiter au mieux ces algorithmes. Après cela, il tient à jour une liste des tags qu'il a détectés.

Une autre fonction importante est la surveillance d'événements déclenchant un inventaire. En effet, il existe plusieurs stratégies pour déterminer le moment de départ d'un inventaire. Les lecteurs les plus simples effectuent des inventaires en boucle, sans jamais s'arrêter, renvoyant ainsi à l'hôte une grosse quantité d'informations. Néanmoins, peu d'applications ont réellement besoin d'effectuer autant d'inventaires, car leur fonctionnement est généralement synchrone : inventaire de tags passant à une vitesse définies devant le lecteur, inventaire à effectuer lors de l'arrivé d'une palette … Ainsi, couplé à des minuteurs ou des capteurs de positions, il est possible de déclencher les inventaires à des moments stratégiques : arrivée ou départ d'une palette, vérification de la présence de tags toutes les minutes.

Finalement, la dernière fonctionnalité principale des lecteurs est la génération de rapports d'inventaire et leur transmission à l'hôte ou au middleware.

On retrouve le sous-composant *CRC*, déjà présent dans le tag. Ce composant est exactement le même. Le calcul et le contrôle du code CRC est un élément important de la communication. C'est pour cela que cette fonctionnalité a été séparée de la logique de

[1] Les algorithmes implantés par le simulateur sont décrits dans la section 0, page 30

contrôle, bien qu'habituellement elle fasse partie du logiciel interne au lecteur et donc de la logique de contrôle. Cela permet d'injecter des fautes spécifiquement dans ce module.

Le sous-composant *modulateur* a pour rôle de coder dans un signal analogique les messages binaires générés par le lecteur à destination des tags. Pour cela, il utilise le codage bit et la modulation définis par les normes. Comme les normes définissent plusieurs codages bit, débits binaires et modulations, la logique de contrôle informe le modulateur des paramètres qu'il doit utiliser.

Le sous-composant *antenne* a pour objectif d'envoyer de la puissance aux tags, ainsi que les informations codées dans le signal analogique généré par le modulateur. La puissance envoyée aux tags est spécifiée par la logique de contrôle, en fonction des normes et de la configuration de l'utilisateur. Elle a aussi pour fonction de recevoir, dans le cas des systèmes RFID UHF, l'onde réfléchie par les tags.

Comme tout système de communication, le lecteur possède un sous-composant *démodulateur*. Ce composant est spécifique au domaine de fréquence utilisée. En effet, comme les phénomènes physiques permettant le fonctionnement de la RFID sont très différents en HF et UHF, le fonctionnement même du démodulateur est foncièrement différent. Dans les systèmes HF, le démodulateur est en charge d'observer la consommation d'énergie faite par l'antenne ; en effet, c'est en modifiant la charge vue par le lecteur, et donc la consommation énergétique, que le tag est capable de communiquer avec. Quant aux systèmes UHF, le fonctionnement est plus classique : l'antenne reçoit l'onde réfléchie par les tags, la transmettant alors au démodulateur, comme si le signal avait été émis directement par le tag.

Finalement, le dernier sous-composant est celui permettant le lien avec le système hôte ou le middleware. Celui-ci est décrit dans la section suivante.

III.2.5.1 Connexion avec le middleware

La connexion avec le système hôte est la passerelle de communication entre le simulateur et le middleware, permettant ainsi la co-simulation complète du système : des tags jusqu'au système d'informations.

La première étape est la définition du mode de connexion avec le middleware : connexion série, réseau… Nous avons choisi de réaliser la connexion avec le système hôte par une liaison réseau car le secteur visé, c'est-à-dire les chaînes de logistiques, utilisent un grand nombre de lecteurs mis en réseau. Le simulateur rendra donc mieux compte du fonctionnement réel du système. De plus, le choix de **SystemC** comme langage de description rend possible la mise en place d'un serveur, et donc d'une connexion réseau, grâce à son ascendance avec le langage de programmation **C++**.

La seconde étape est la définition du protocole de communication avec le lecteur. Il existe des standards, notamment celui développé par **EPCGlobal** nommé *LowLevel Reader Protocol (LLRP)* [EPCLLRP]. Ce standard définit les formats et le contenu des messages échangés entre le lecteur et le middleware. Ce standard est complet et capable de s'adapter à la plupart des cas d'application. Ceci le rend complexe et impose de nombreuses contraintes au niveau du lecteur. De plus, les fonctionnalités offertes par ce standard dépassent grandement les objectifs de notre travail : en effet, il offre des fonctionnalités de configuration avancée des lecteurs, comme des règles de déclenchement d'inventaire et d'envoi automatique de rapport alors que nous avons simplement besoin de pouvoir configurer les options de l'inventaire décrit précédemment, de déclencher manuellement le démarrage d'un inventaire applicatif et de recevoir immédiatement le rapport de cet inventaire.

Il faut donc un protocole de communication entre le lecteur et le middleware relativement simple, qui permette de configurer simplement et sommairement le fonctionnement du

lecteur : quand et quels paramètres utiliser pour effectuer l'inventaire et comment rendre compte du résultat de cet inventaire. Il faut aussi, dans la mesure du possible, définir un protocole permettant l'intégration facile du simulateur avec les middlewares existants.

Au cours d'études préliminaires, nous avons étudié le lecteur **ALR-8800** de la société *AlienTechnology*. Ce lecteur UHF, respectant la norme EPC Classe 1 Génération 2 [EPCC1G2], est un des plus utilisés dans l'industrie. Bien que le protocole [ALIEN] qu'il utilise pour communiquer avec un système hôte ne respecte aucun standard, la plupart des middlewares ont implanté ce protocole et sont donc capables de communiquer avec lui. Bien que complet et comportant de nombreuses commandes permettant de configurer et d'utiliser les différentes fonctions de ce lecteur, il possède un ensemble restreint et simple de commandes permettant de gérer les inventaires. Nous avons donc pris le parti de réaliser l'interface du modèle du lecteur et du système hôte en utilisant ce protocole. Ce lecteur est utilisé par la suite pour des expérimentations. Il sera ainsi possible d'utiliser indifféremment notre simulateur ou le lecteur physique dans les différentes applications de test futur.

Ce protocole permet de configurer sommairement l'inventaire applicatif réalisé par le lecteur, de le déclencher et de récupérer la liste des tags lus. L'inventaire applicatif est l'action haut niveau permettant de récupérer la liste complète des tags présents dans le champ du lecteur ; il est réalisé par l'exécution successive d'inventaires simples, tels que défini dans la norme EPC Classe 1 Génération 2. En effet, lors d'un inventaire simple, tous les tags ne sont pas détectés ; il est alors nécessaire de réaliser plusieurs inventaires simples successifs afin de détecter la totalité des tags. D'un point de vue fonctionnel, l'inventaire applicatif permet de déclencher en une seule fois le nombre nécessaire d'inventaires simples permettant la détection de tous les tags présents dans le champ du lecteur.

Pour cela, il organise le protocole défini par EPC Classe 1 Génération 2 à l'intérieur de boucles, permettant à l'utilisateur de préciser le nombre de répétitions de chaque étape du protocole anticollision. Pour cela, il définit trois nouveaux paramètres : **AcqG2Cycles**, **AcqG2Select** et **AcqG2Count**. Le Code III-1 montre l'utilisation qu'en fait le lecteur pour réaliser l'inventaire.

Code III-1 : Algorithme de l'inventaire applicatif réalisé par l'ALR-8800

```
1 Pour chaque cycle dans AcqG2Cycles :
2    Pour chaque select dans AcqG2Select :
3       Envoi de commandes SELECT avec le masque prédéfini ;
4    select suivant ;
5    Pour chaque count dans AcqG2Count :
6       Parcours des slots et des frames ;
7    count suivant ;
8 cycle suivant ;
```

On voit ainsi que le lecteur effectue **AcqG2Count** fois le parcours des slots et des frames, ce qui revient à effectuer **AcqG2Count** fois la résolution des collisions. Ceci permet de s'assurer que tous les tags sont bien inventoriés, même en présence de tags masqueurs[1]. De plus, afin de s'assurer que les bons tags soient sélectionnés pour l'inventaire, le lecteur envoie **AcqG2Select** commandes de sélection. Le lecteur effectue plusieurs inventaires successifs, permettant ainsi d'être sûr d'avoir inventorié tous les tags. En effet, ceci est nécessaire car les systèmes RFID n'ont pas des taux de lecture égaux à 100% [DER07a].

[1] Le principe des tags masqueurs est présenté à la section 0, page 19.

Le protocole du modèle du lecteur encadrant les communications avec le système hôte définit donc les commandes nécessaires à la configuration de l'inventaire et à la définition des paramètres : **ACQG2CYCLES**, **ACQG2COUNT** et **Q**, le nombre de slots. Il définit aussi la manière dont le système hôte déclenche les inventaires et récupère leur résultat. Ainsi, le protocole définit la commande déclenchant à la demande un inventaire, qui renverra à sa fin, son résultat. Le résultat comprend les identifiants des tags et le nombre de fois qu'ils ont été vus lors de l'inventaire.

III.2.6 Modélisation de la communication lecteur/tag

La communication entre les tags et les lecteurs est le point clef du modèle. Comme vu précédemment, cette communication est rarement simulée entièrement : soit elle est limitée au signal analogique se propageant dans le canal de transmission, soit les données binaires et le protocole de communication entre les tags et les lecteurs sont simulés. L'objectif ici est donc de proposer une modélisation permettant de simuler à la fois les données binaires échangées et les signaux analogiques se propageant dans le canal de transmission. La suite du document explique donc la modélisation des données binaires, la modélisation des signaux analogiques et la modélisation du canal de transmission.

III.2.6.1 Modélisation des échanges numériques

Figure III-8 : Diagramme UML du type de données « DATA »

Le premier élément intervenant dans la communication entre le lecteur et le tag est le message binaire. Celui-ci est composé de différents champs, fixés par les normes. Ces champs peuvent être communs à tous les messages échangés, ou bien spécifiques à l'action réalisée. Dans la norme EPC Classe 1 Génération 2 [EPCC1G2], les champs définis sont variables et dépendent de chaque action. Les données ainsi transmises représentent en général quelques dizaines, voire centaines de bits, mais peuvent dépasser les 4 000 bits dans certains cas. Il faut donc un type de données **SYSTEMC** pouvant stocker autant de bits et permettant d'accéder rapidement à la partie du message désirée. Ce type de donnée est appelé **DATA**. Il permet de représenter les différents champs quelle que soit la norme utilisée. Ainsi, les interfaces entre les différents composants sont standards : les échanges entre la logique de contrôle et le modulateur ou entre le modulateur et le démodulateur se réalisent en utilisant ce type de données ; il devient alors simple de remplacer un composant par un autre. De plus, ce composant permet l'accès et la modification directe d'un champ, d'un bit d'un champ ou d'un

bit du message. Sa taille est variable et permet donc de s'adapter à toutes les situations. La Figure III-8 donne le diagramme de la donnée « **DATA** », simplifié aux fonctions propres au simulateur présentées précédemment.

Le message ainsi généré n'a plus qu'à être modulé pour être transmis au destinataire. Le message devient donc un signal analogique. La modélisation de ce signal analogique est présentée dans la section suivante.

III.2.6.2 Modélisation des échanges analogiques

Le signal analogique que s'échangent le lecteur et le tag doit à la fois transmettre de la puissance du lecteur vers le tag et transmettre les informations du lecteur vers le tag et du tag vers le lecteur.

Plusieurs solutions existent pour simuler ce signal. La première solution est la simulation du signal via la librairie **SYSTEMC-AMS**. Cette solution utilise les équations des phénomènes physiques mis en œuvre. La simulation est alors proche de la réalité : le signal analogique est simulé de manière continue. Néanmoins, cette solution est très chronophage à cause de la quantité d'équations à résoudre. Comme nous souhaitons pouvoir simuler de nombreux tags et interfacer le simulateur avec un middleware, cette solution n'est pas envisageable.

La solution que nous avons choisie est la représentation discrète du signal analogique grâce à un type de données appelé « **INFORMATION** ». Cette représentation discrète liste les différents paramètres analogiques et leurs valeurs permettant de reconstruire le signal. Ainsi, il n'est pas nécessaire de simuler le signal de manière continue. Les calculs n'interviendront que lorsqu'un des paramètres change. Evidemment, les caractéristiques des signaux analogiques étant limitées à celles choisies, cela engendre une perte de précision dans la modélisation des signaux analogiques. Ces paramètres, illustrés sur la Figure III-9, sont :

- le type de modulation : ASK, PSK ;
- le codage bit : PIE, « 1 parmi 4 », Manchester, *etc.* ;
- le débit binaire ;
- le début de signal ou « Start of Frame » : c'est le signal analogique permettant d'indiquer qu'une communication commence et qui permet aux différents acteurs de la communication de se synchroniser ;
- la fin de signal ou « End of Frame » : c'est le signal analogique permettant d'indiquer que la communication est terminée ;
- la puissance du signal ;
- les données contenues dans le signal, stockées dans le type de données « **DATA** » défini précédemment ;
- et finalement, un paramètre permettant d'évaluer la qualité intrinsèque du signal. Ce paramètre qualité, exprimé par une valeur comprise entre 0 et 100, tient compte notamment du rapport signal sur bruit, de la qualité de la synchronisation ou encore de la distorsion. Lorsque la qualité d'un signal devient trop faible, certains équipements seront incapables de démoduler et de décoder les données qu'ils contiennent.

Lorsque deux ou plusieurs émetteurs souhaitent envoyer une « **INFORMATION** » à un même destinataire, ils émettent chacun une « **INFORMATION** ». Ces « **INFORMATIONS** » sont alors associées et transformées en un seul et unique « **INFORMATION** » qui est récupéré par le destinataire. On retrouve alors le phénomène de collision présenté à la section II.1.3.1, ainsi que le phénomène de tag masqueur. Pour réaliser cela, il a été nécessaire de créer un nouveau signal **SYSTEMC** dans lequel plusieurs émetteurs peuvent écrire. Ce signal va résoudre l'association des « **INFORMATIONS** ». Pour cela, il va, dans un premier temps, comparer les

données envoyées ainsi que les paramètres du signal : en effet, si les différentes données et les différents paramètres sont identiques, il n'y a pas de collision. Puis, dans le cas où au moins deux « **INFORMATIONS** » contiennent des données différentes, il va comparer les différentes puissances des « **INFORMATIONS** » pour déterminer s'il y a une collision ou si une des « **INFORMATIONS** » est « plus forte » que les autres et les masque. Pour cela, il utilise l'équation (II.1) définie dans la section 0 :

$$P_0 > c \sum_{j=1}^{N} P_j \qquad \text{(III.1)}$$

où P_0 représente la puissance de l'« **INFORMATION** » dont on souhaite vérifier s'il est capturable ; P_j la puissance de l'« **INFORMATION** » ***j***, un des ***N-1*** autres « **INFORMATIONS** »; et ***c*** le ratio de capture. Le ratio de capture est ici fixé à 10.

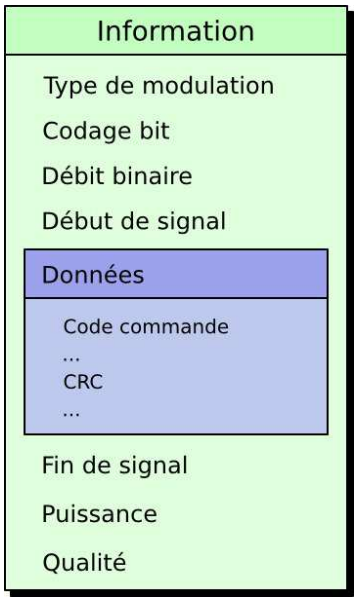


Figure III-9: Paramètres permettant de reconstruire le signal analogique – type de données « INFORMATION »

Tableau III-III : Tableau décisionnel utilisé par le signal de résolution transportant des « INFORMATIONS »

Cas	Toutes les puissances de même ordre de grandeur	Une « INFORMATION » avec une puissance fortement supérieure	
« **INFORMATIONS** » identiques	X	X	Une copie des **INFORMATIONS** avec les puissances sommées
« **INFORMATIONS** » différentes	X		**COLLISION** avec les puissances sommées
		X	Une copie de l'**INFORMATION** la plus puissante avec les puissances sommées

De plus, ce signal est aussi capable d'associer les puissances des « **INFORMATIONS** ». En effet, lorsque plusieurs lecteurs envoient de la puissance aux tags, ceux-ci sont capables de récupérer les puissances issues des différents lecteurs. Ce signal, faisant l'association des « **INFORMATIONS** », effectue la somme des puissances.

Ainsi, lorsque plusieurs « **INFORMATIONS** » entrent dans ce signal de résolution pour être transportés vers le destinataire, une « **INFORMATION** » en sort. Cette « **INFORMATION** » est

l'association des différents « **INFORMATIONS** » entrant dans le signal en utilisant le Tableau III-III.

III.2.6.3 Modélisation du transfert d'« INFORMATIONS » entre les tags et les lecteurs

La dernière partie modélisée du système concerne l'émission des données entre les tags et les lecteurs. Pour les mêmes raisons que pour la modélisation discrète des signaux analogiques, le canal n'a pas été modélisé de manière continue. De plus, afin de séparer les effets locaux et globaux de la communication – bruit global, obstacle local – ce lien a été divisé en deux :

1. le canal de transmission, spécifique à un couple lecteur/tag
2. l'environnement global

Le canal de transmission est en charge de tous les paramètres du transfert de l'« **INFORMATION** » entre un tag et un lecteur. Ces paramètres sont :

1. la distance entre le tag et le lecteur, influençant le transfert de puissance ;
2. le taux d'erreur binaire ;
3. l'impact sur la qualité du signal, prenant en compte notamment le bruit et les différents obstacles présents entre ce tag et le lecteur. Le canal de transmission modifie donc l'**INFORMATION** entrante en fonction de ces paramètres avant de l'envoyer vers l'équipement destinataire.

Ainsi, le signal **INFORMATION** sortant voit ses paramètres modifiés. Sa puissance est diminuée en fonction de la distance. Le facteur d'atténuation est fonction de la distance entre l'émetteur et le récepteur, mais aussi de la longueur d'onde du signal utilisé pour communiquer. Pour les communications UHF, la diminution de l'intensité du signal est proportionnelle à $\frac{1}{d}$. [SEO05] donne la relation en champ lointain existant entre la puissance émise par le lecteur et la puissance reçue par le tag :

$$\frac{P_r}{P_t} = g_t g_r \left(\frac{\lambda}{4\pi d}\right)^n \quad \text{(III.2)}$$

Où $\boldsymbol{P_r}$ est la puissance reçue par le tag, $\boldsymbol{P_t}$ est la puissance transmise par le lecteur, $\boldsymbol{g_t}$ est le gain de l'antenne de l'émetteur, $\boldsymbol{g_r}$ est le gain de l'antenne du receveur, λ est la longueur d'onde du signal, $\boldsymbol{d}$ est la distance entre l'émetteur et le receveur et $\boldsymbol{n}$ est l'exposant d'affaiblissement. Cet exposant est caractéristique de l'environnement : en champ libre, sans obstacle ni élément perturbateur, il est égal à 2. Plus cet exposant est grand, plus il y a d'obstacles, plus l'environnement atténue la puissance du signal. Comme le facteur d'atténuation est implanté dans le canal de transmission, il faut séparer les effets des antennes de l'effet du canal. Ainsi, le facteur d'atténuation du canal seul est :

$$D = \left(\frac{\lambda}{4\pi d}\right)^n \quad \text{(III.3)}$$

Les données que contient une **INFORMATION** transitant dans le canal de communication sont aussi modifiées, en fonction du taux d'erreur binaire. Ainsi, le canal de communication inverse les bits de la donnée en fonction de modèle probabiliste. [MCM70, TSA69] proposent une modélisation de cet effet à base de chaînes de Markov. Cette modélisation permet de représenter les effets classiques engendrant des erreurs binaires : en effet, les erreurs binaires apparaissent groupées dans les données binaires. Dans notre cas, nous avons adopté une simple loi uniforme permettant de décider si un bit doit être inversé ou non en fonction du Taux d'Erreur Binaire.

La Figure III-10 illustre ce canal de transmission ainsi que les échanges d'informations et de puissance qui sont réalisées.

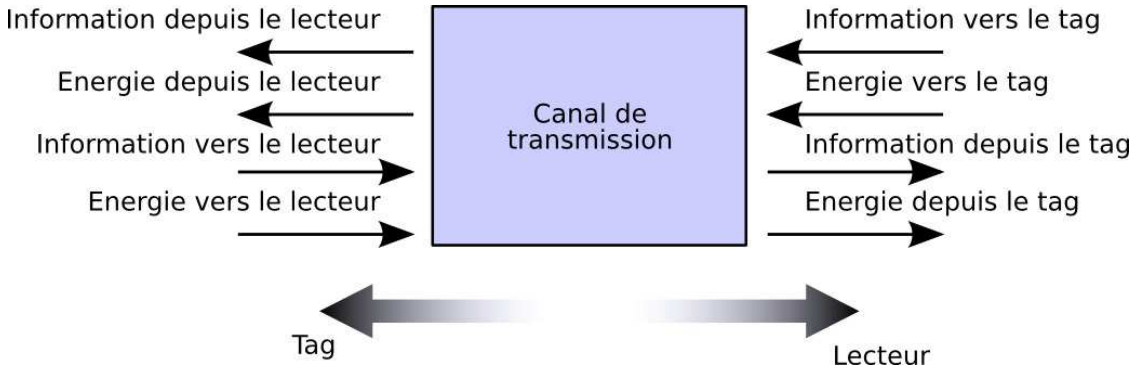


Figure III-10 : Modélisation du canal de transmission

L'environnement global est en charge des paramètres globaux, impactant la qualité du signal, prenant en compte notamment le bruit commun à tous les équipements et les différents effets comme les réflexions et les chemins multiples.

Tous les paramètres de l'environnement du tag, donc du canal de transmission et de l'environnement global tels que définis dans **SERFID**, peuvent être modifiés en cours de simulation. Par exemple, au cours de la simulation, la distance entre un tag et un lecteur peut être modifiée, simulant alors le déplacement du tag dans son environnement. Il est alors possible de réaliser des scénarios complexes tels que le passage de tags devant plusieurs lecteurs grâce à un convoyeur ou encore la rotation d'une palette vis-à-vis d'un point de lecture.

Le code de ce composant est donné en exemple en annexe F.

III.2.7 SERFID : Simulation et Evaluation des systèmes RFID

L'implantation de ces modèles au sein d'un même programme donne le simulateur nommé **SERFID**, pour Simulation et Evaluation des systèmes RFID[1]. La Figure III-11 illustre les différentes interactions qu'il existe entre tous ces modèles au sein de **SERFID**. Ainsi, on peut voir que **SERFID** est capable de simuler plusieurs lecteurs et plusieurs tags simultanément. On retrouve aussi la représentation des liens locaux entre les lecteurs et les tags – les canaux – et la représentation des paramètres environnementaux communs à tous les équipements – l'environnement global : chaque couple lecteur/tag utilise son propre canal de communication, qui interrogera l'environnement global pour connaître les paramètres globaux. Tags, lecteurs et canaux sont connectés les uns aux autres grâce aux signaux de résolution décrits dans la section III.2.6.2. Ainsi, il y a un unique signal de résolution pour relier le lecteur #1 à tous ses canaux ; un unique signal pour relier le tag #1 à tous ses canaux ; *etc.*

Lorsque le lecteur #1 souhaite envoyer un message aux tags, il envoie une **INFORMATION** sur son signal d'envoi – le trait continu rouge. Cette **INFORMATION** arrive alors à tous ses canaux : L1R1, L1R2, …, L1R*n*. Chaque canal va alors modifier cette **INFORMATION** en fonction des paramètres locaux et globaux, puis la transmet au tag associé en écrivant l'**INFORMATION** modifiée sur le signal du tag servant à recevoir les **INFORMATIONS** – trait continu rouge. Le tag #1 souhaitant répondre va alors mettre sur son signal d'envoi – signal tiret bleu – l'**INFORMATION** de réponse, la transmettant par la même occasion à tous ces canaux : L1R1,

[1]Ou encore "Simulation and Evaluation of RFID systems".

L2R1, …, L*k*R1. Chaque canal va alors modifier l'**INFORMATION** avant de la transmettre vers le lecteur associé en l'écrivant sur le signal de réception de ce dernier – trait tiret bleu.

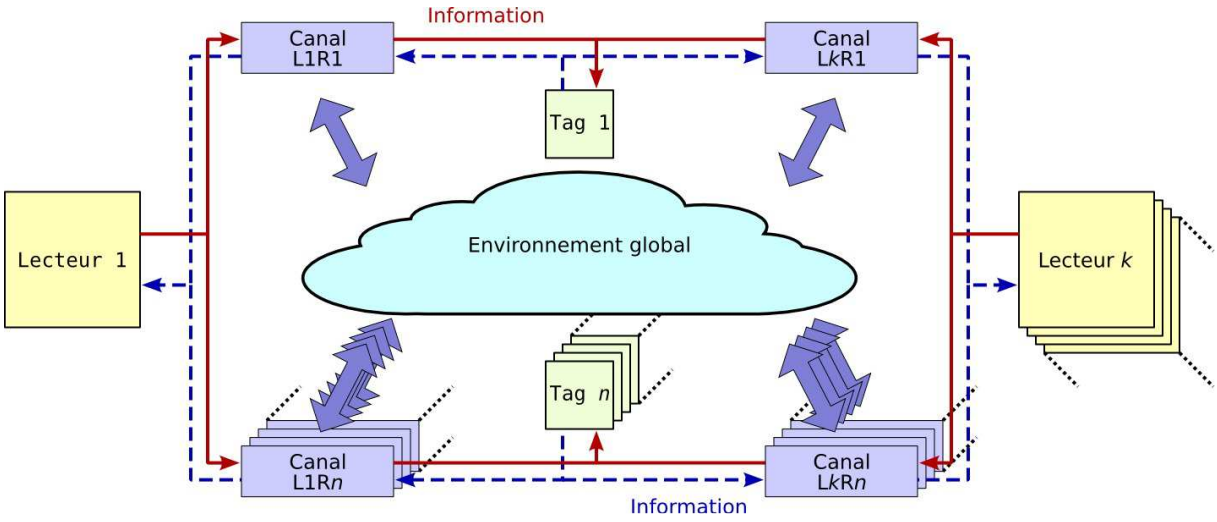


Figure III-11 : Interconnexion des différents modèles au sein de SERFID

Un modèle de système RFID vient d'être proposé. Ce modèle est basé sur une description fonctionnelle du système permettant à la fois d'avoir des temps de simulation relativement courts et prenant en compte les principaux effets physiques du système.

III.3 Validation du modèle

Afin de s'assurer du bon fonctionnement du simulateur, c'est-à-dire que son comportement représente correctement les systèmes réels, il est important de valider chacun des éléments modélisés du système.

III.3.1 Validation unitaire de chaque composant

Le fonctionnement de chaque composant du système a été individuellement validé par des tests unitaires. Dans ce paragraphe nous nous intéresserons particulièrement à la validation du canal de transmission et à deux de ces fonctionnalités : le transfert d'énergie et l'injection d'erreur binaire.

La Figure III-12 illustre la simulation de la puissance reçue par un tag en fonction de la distance qui le sépare du lecteur et lorsque celui-ci lui envoie 2 000 mW. La puissance reçue par le tag diminue bien proportionnellement à $\frac{1}{d^2}$.

A partir de cette puissance en fonction de la distance, il est possible de calculer la distance de fonctionnement maximale du tag. Par exemple, si un tag a besoin d'un signal réceptionné d'au moins 0,04 milliwatt, il sera capable de fonctionner correctement jusqu'à 6 mètres.

L'implantation du taux d'erreur binaire[1] est aussi vérifiée : la concordance entre le taux d'erreur binaire simulé correspond au taux d'erreur théorique souhaité. La Figure III-13 illustre la cohérence entre ces deux taux.

[1] Le taux d'erreur binaire se calcule avec la formule suivante :

$$Taux\ d'erreur binaire = \frac{Nombre\ de\ bits\ erronés}{Nombre\ de\ bits\ transmis}$$

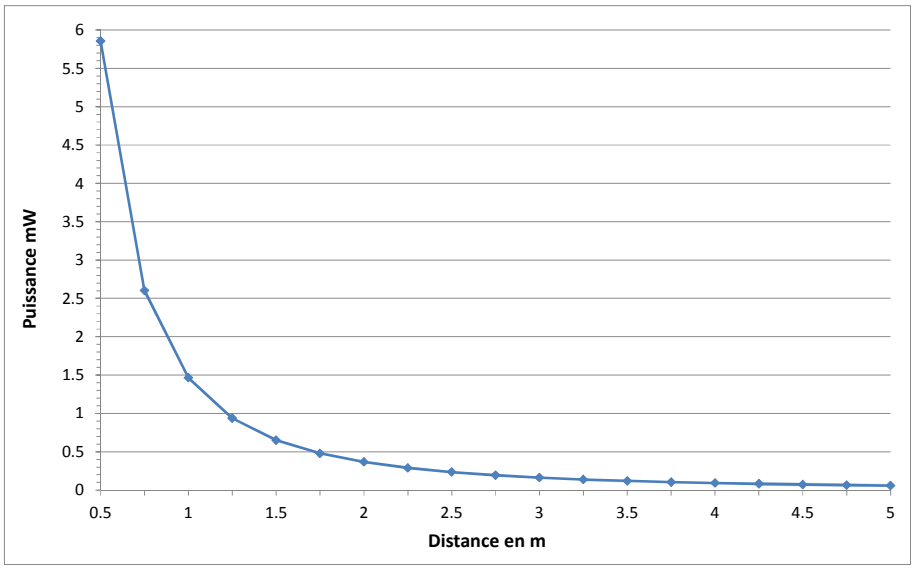


Figure III-12 : Puissance reçue par les tags en fonction de leur éloignement vis-à-vis du lecteur

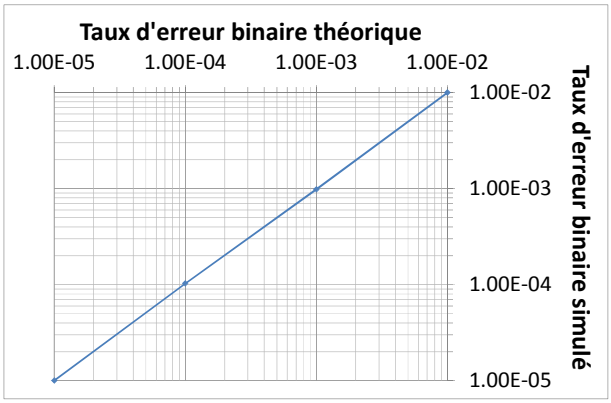


Figure III-13 : Taux d'erreur binaire simulé par rapport au taux d'erreur binaire théorique

Dans la suite du document, nous validons le comportement dynamique du simulateur pour des systèmes complets. Deux points principaux sont étudiés : le déroulement de l'algorithme anticollision et l'estimation du temps nécessaire à cet algorithme pour fonctionner.

III.3.2 Validation fonctionnelle de l'algorithme anticollision

Le premier point à valider est l'implantation du protocole anticollision. Après avoir validé le comportement de chaque équipement indépendamment, nous validons le fonctionnement dynamique du simulateur, c'est-à-dire le comportement de l'algorithme anticollision lorsqu'il y a plusieurs tags à inventorier. Afin de valider ce protocole, nous réutilisons les équations (II.5), (II.6) et (II.7), permettant de prédire le nombre de slots *singletons*, *vides* ou *collisions* au cours d'un frame. Plusieurs cas ont été simulés. La première simulation réalisée concerne un système possédant 5 tags à inventorier. Pour cela, le lecteur a été configuré pour que son paramètre **Q** soit égal à 4, soit d'après l'équation (II.2), pour utiliser des frames composés de 16 slots. Le Tableau III-IV donne les résultats attendus ainsi que les résultats moyens obtenus lors des 250 simulations. Ainsi, dans ce cas-là, on a en moyenne théorique $S(5{,}16) = 3{,}9$ slots ayant été choisis par un seul et unique tag ; $V(5{,}16) = 11{,}6$ slots n'ayant été choisis par aucun tag ; et $C(5{,}16) = 0{,}5$ slots ayant été choisis par au moins deux tags. Après avoir réalisé plusieurs inventaires, le nombre moyen de slots *singletons* est égal à 3,84, soit une erreur de ± 0,06 slots ; le nombre moyen de slots *vides* est égal à 11,59, soit une erreur de ±

0,01 slots ; et le nombre moyen de slots *collisions* est égal à 0,572, soit une erreur de ± 0,072 slots.

Tableau III-IV : Nombre de slots *singletons*, *vides* ou *collisions* pour 5 tags et 16 slots – théorie et simulation

Théorie	Simulation			
	Moyenne	Ecart type	Min	Max
$S(5,16) = 3,9$	3,84	1,20	1	5
$V(5,16) = 11,6$	11,59	0,62	11	13
$C(5,16) = 0,5$	0,572	0,59	0	2

La Figure III-14 donne les résultats des autres simulations. Durant ces simulations, le lecteur a été configuré avec une valeur de **Q** égale à 8 : il a donc utilisé des frames de 256 slots. Plusieurs simulations différentes ont été faites afin d'avoir une bonne couverture de l'utilisation de l'algorithme anticollision :

- une simulation avec 0 tags. Le résultat est prévisible : uniquement des slots *vides* ;
- une simulation avec 10 tags. On se retrouve avec l'utilisation de frame surdimensionné : le nombre de slots *vides* est proche de 246 ; celui de slots *singletons* est légèrement inférieur à 10, dû à la probabilité non nulle que, malgré le nombre de slots disponibles, 2 tags choisissent quand même un slot identique ; et celui de slots *collisions* proche de 0 ;
- une simulation avec une centaine de tags. Dans cette simulation, bien que le nombre de slots *vides* soit toujours majoritaire, le nombre de slots *singletons* devient important et le nombre de slots *collisions* n'est plus nul ;
- une simulation avec 256 tags. Dans cette simulation, il y a autant de slots que de tags. Dans ce genre de cas, il y a autant de slots *vides* et *singletons* et légèrement moins de slots *collisions* ;
- une simulation avec un millier de tags. Dans cette simulation, le nombre de slots *collisions* devient fortement majoritaire. Quelques tags réussissent néanmoins à choisir un slot dans lequel ils sont seuls. Et le nombre de slots *vides* est très faible.

Le détail des résultats de ces simulations est donné en annexe H.

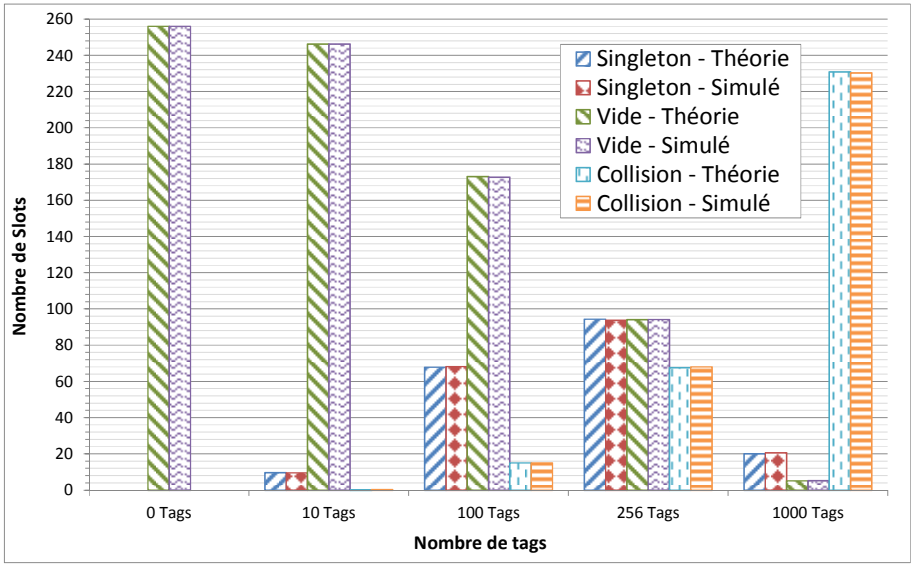


Figure III-14 : Nombre de slots *singletons*, *vides* ou *collisions* pour 0, 10, 100, 256 et 1000 tags dans un frame de 256 slots – théorie et simulation

Grâce à ces observations, il apparaît que la description fonctionnelle basée sur les transactions entre les différents composants intervenant dans l'algorithme anticollision fonctionnent correctement. Ainsi, le comportement fonctionnel des différentes logiques de contrôle, mémoires, modulateur, démodulateur ou contrôleur CRC est celui attendu. En effet, si un de ces composants ne fonctionnait pas correctement, il serait alors impossible d'exécuter cet algorithme.

III.3.3 Validation temporelle du simulateur

Un autre point reste à valider : le simulateur doit permettre de donner un comportement temporel réaliste du système. Ainsi, chaque composant introduit des délais dans les différentes transactions qu'il réalise. Afin de valider ce comportement, il faut dans un premier temps étudier l'enchaînement des commandes lors de l'algorithme anticollision. Ainsi, il sera possible de prédire la durée d'un frame de l'algorithme anticollision.

Comme vu à la section 0, l'algorithme anticollision commence avec une phase de sélection des tags grâce à la commande « **SELECT** ». Après la phase de sélection, le lecteur commence le parcours des slots du premier frame grâce à la commande « **QUERY** ». Le lecteur passe ensuite d'un slot à un autre grâce à la commande « **QUERYREP** ». Finalement, lorsqu'il détecte un tag unique durant un slot, il utilise la commande « **ACK** » pour récupérer l'identifiant EPC contenu dans le tag. Le tag a deux réponses possibles : l'envoi de son identifiant temporaire et l'envoi de son identifiant EPC complet. Les Figure III-15, Figure III-16 et Figure III-17 illustrent l'utilisation de ces commandes durant l'exécution d'un algorithme d'inventaire respectivement dans le cas de slots vides, de singletons, ou de collisions.

Ainsi, la Figure III-15 montre pour des slots vides :

- en haut, après un signal de synchronisation **FS**, une commande **SELECT**, suivis, après un temps **T4**, d'un signal de préambule **P** et d'une commande **QUERY** permettant de démarrer l'inventaire (et par la même occasion le premier slot) puis d'une attente minimum définie par la norme EPC Classe 1 Génération 2 [EPCC1G2] ;
- en bas, un signal de synchronisation **FS** et une commande **QUERYREP**, abrégée par **QR**, permettant de démarrer un slot et une attente minimum définie par la norme.

L'attente minimum comprend le temps **T1** d'attente d'une réponse de la part du tag et d'un temps **T3** permettant au lecteur de renvoyer une nouvelle commande.

La Figure III-16 montre le déroulement d'un slot *singleton* :

- dans la partie haute du graphique :
  - en haut, le démarrage du slot lorsqu'il est le premier de l'inventaire ;
  - au milieu, le démarrage du slot dans les autres cas ;
  - et en bas la réponse d'un tag envoyant son identifiant temporaire après un temps **T1** ;
- dans la partie basse du graphique :
  - en haut : que le slot soit le premier ou non, l'envoi de la commande **ACK** après un temps **T2** ;
  - en bas : après un temps **T1** l'identifiant EPC ; le lecteur attend enfin un temps **T2** avant de passer au slot suivant.

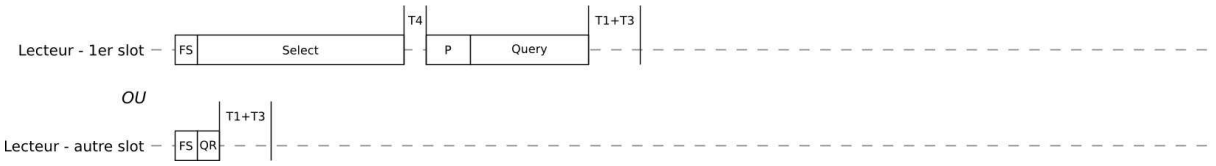


Figure III-15 : Composition d'un slot *vide* – figure du haut : cas du 1er slot avec les commandes SELECT et QUERY ; figure du bas : cas des autres slots avec la commande QueryREP (QR)

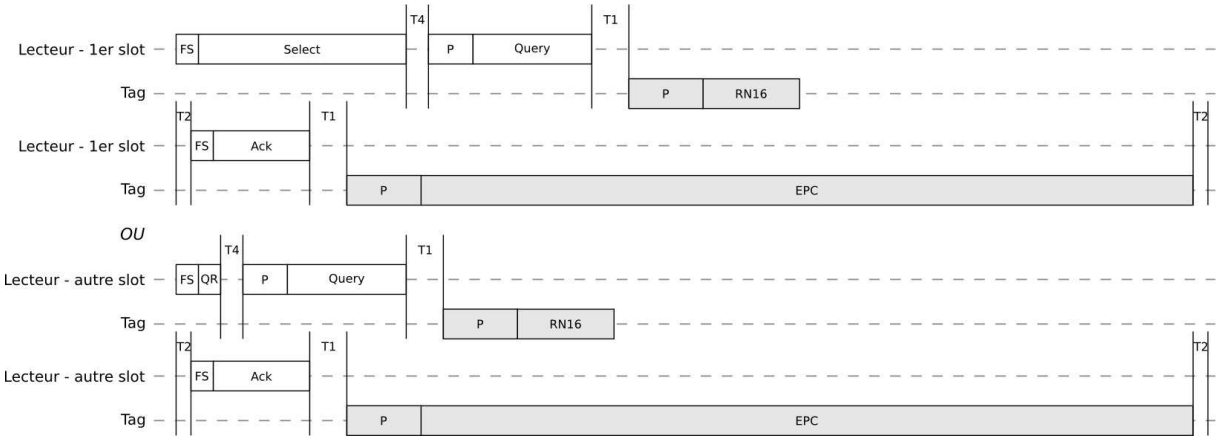


Figure III-16 : Composition d'un slot *singleton* – figure du haut : cas du 1er slot avec les commandes SELECT et QUERY ; figure du bas : cas des autres slots avec la commande QueryREP (QR)

La Figure III-17 montre enfin le déroulement d'un slot collision :

- en haut, le démarrage du slot lorsqu'il est le premier de l'inventaire ;
- en dessous, le démarrage du slot dans les autres cas ;
- en bas, deux tags souhaitent envoyer en même temps leur identifiant temporaire après un temps **T1**. Il y a donc collision et le slot s'arrête directement après. Le lecteur peut passer au slot suivant après un temps **T2**.

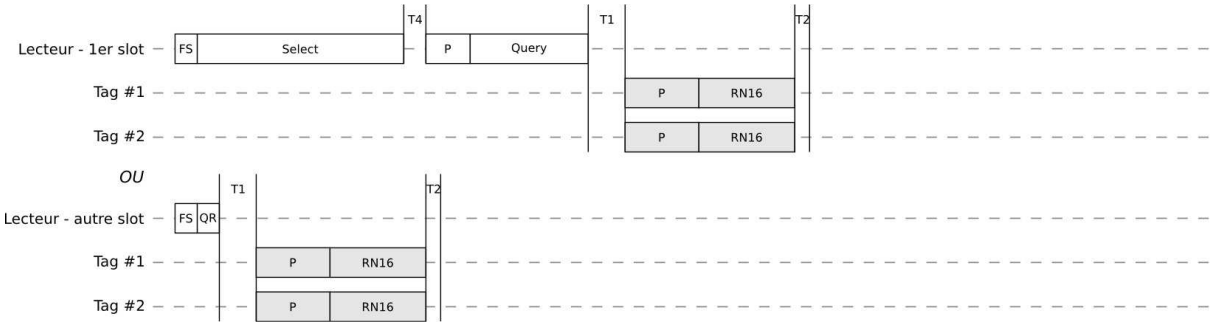


Figure III-17 : Composition d'un slot *collision* – figure du haut : cas du 1er slot avec les commandes SELECT et QUERY ; figure du bas : cas des autres slots avec la commande QueryREP (QR)

Par ces figures, on peut voir que les slots *singletons* sont significativement plus longs que les slots *vide* ou *collisions*. En effet, pour les slots autres que le premier, les slots *singletons* ont une durée moyenne de ***2 180µs*** alors que les slots *vides* ou *collisions* ont respectivement une durée moyenne de ***150µs*** et ***442µs***. Dans le cas du premier slot, ces valeurs sont majorées de ***630µs***.

On peut aussi voir sur ces figures les contraintes temporelles liées à la norme :

- le lecteur doit attendre un temps **T4** entre deux envois de commande, pour permettre au tag de récupérer suffisamment d'énergie ;
- le lecteur doit laisser au tag un temps **T1** pour répondre ;
- après une réponse d'un tag, le lecteur attend un temps **T2** avant d'envoyer une commande ;
- après une attente **T1** sans réponse, le lecteur attend un temps **T3** avant d'envoyer une autre commande.

En calculant le temps que prend chaque commande, il est alors possible de déterminer le temps nécessaire au parcours de chaque type de slots. En utilisant les équations (II.5), (II.6) et (II.7) donnant le nombre théorique de slots *vides*, *singletons* et *collisions* dans un frame, il est alors possible de déterminer le temps que prend le parcours du premier frame. Afin de calculer les différents temps intervenant dans la communication, les valeurs des paramètres définis dans la norme ont été fixées dans **SERFID** dans l'intervalle proposé par cette dernière. Ces paramètres et ces valeurs sont disponibles dans le Tableau G-I de l'annexe G. On y retrouve par exemple le temps d'un symbole **0** et d'un symbole **1** pour la liaison montante, temps différents de par l'utilisation du codage bit PIE[1] ; les temps T1, T2, T3 et T4 définis précédemment ; le débit binaire de la liaison montante ; et la durée des différents signaux permettant de marquer le début et la fin du signal analogique, nommé **FRAMESYNC** – FS sur les figures – et **PREAMBULE** – P sur les figures.

Le Tableau III-V donne le temps de chacune des commandes, à partir du nombre de bits contenus dans la commande et des paramètres définis précédemment. Il est à noter que la commande **ACK** n'a pas une durée fixe : en effet, sa durée est dépendante du nombre de **0** et de **1**. Or, il n'est pas possible pour cette commande de connaître cette information puisqu'elle utilise l'identifiant temporaire aléatoire des tags, cet identifiant étant généré aléatoirement lors de l'exécution de l'algorithme.

Tableau III-V : Temps des différentes commandes entrant dans l'exécution de l'algorithme anticollision

Commande	Taille	Nombre de 0	Nombre de 1	Début de signal	Temps
SELECT	45	33	12	FS	393,75µs
QUERY	22	16	6	P	250µs
QUERYREP	4	2	2	FS	75µs
ACK	18	Entre 1 et 17	Entre 1 et 17	FS	Entre 150µs et 262,5 µs
RN16	16				291,7µs
EPC	128				1458,3µs

Ainsi, d'après toutes les informations précédentes, le temps d'un frame est en moyenne, avec l'équiprobabilité de la distribution binaire pour la commande **ACK** :

$$\begin{aligned} t_{moyen}(N,L) = \quad & (t_{Select} + T4) + \left(t_{Query} + T1\right) \\ & + (L(N,L) - 1)\left(t_{QueryRep+T1}\right) \\ & + \left(S(N,L) + C(N,L)\right)(T_{RN16} + T2) \\ & + S(N,L)\left(T_{ACK_{moyen}} + T1\right) \\ & + S(N,L)(T_{EPC} + T2) + E.T3 \end{aligned} \qquad \text{(III.4)}$$

[1] Le codage bit PIE est présenté dans la section II.1.2.1, page 13.

Dans le meilleur cas, lorsqu'il y a un maximum de **0** dans la commande **ACK**, et dans le pire il y a un maximum de 1.

Le Tableau III-VI donne les temps du premier frame de l'inventaire lorsque le lecteur est configuré pour utiliser 256 slots et qu'il doit inventorier 0, 10, 100, 256 et 1000 tags. Le cas de simulation dans lequel il n'y a aucun tag à inventorier ne fait pas apparaître de différences entre les temps minium, maximum et moyen. En effet, dans ce cas-là, le lecteur n'envoie jamais de commande **ACK**, et il n'y a donc pas d'inconnu pour le calcul du temps du frame. On peut ainsi voir que l'erreur entre le temps attendu et la simulation est de 0,01 ms, soit proche de 0%. Il est de plus à noter, pour les autres cas, qu'il est tout à fait normal d'avoir des temps simulés minimums vraiment inférieurs au temps théorique minimum.

On remarque aussi que le temps moyen d'un tour d'inventaire – *i.e* le parcours des slots du premier frame – n'est pas proportionnel au nombre de tags. Ceci est dû au fait que, dans un premier temps, lorsqu'il y a peu de tags, il n'y a pas de slot collision. Ensuite, jusqu'à une certaine quantité de tags, dépendant du nombre de slots disponibles, le nombre de slots *collisions* et le nombre de slots *singletons* augmentent et le nombre de slots *vides* diminue : le temps augmente donc aussi. Passé cette quantité de tags, le nombre de slots *vides* et *singletons* diminuent au profit des slots *collisions* : le temps du tour d'inventaire diminue alors. En effet, la durée d'un slot *vide* est en moyenne 3 fois plus courte que la durée d'un slot *collision* qui est en moyenne 5 fois plus courte que la durée d'un slot *singleton*. On observe ce phénomène dans le Tableau III-VI : le temps d'un tour d'inventaire de 256 slots pour 256 tags est plus long que le temps d'un tour d'inventaire de 256 slots pour 1000 tags. En effet, d'après les équations (II.5), (II.6) et (II.7), lorsque l'on inventorie 256 tags, il y a théoriquement 94 slots vides, 94 slots singletons et 68 slots collisions lors du premier tour d'inventaire, soit l'équivalent de 1844 slots *vides* alors que lorsque l'on inventorie 1000 tags, il y a théoriquement 4 slots *vides*, 20 slots *singletons* et 232 slots *collisions* soit l'équivalent de 1764 slots *vides*.

Tableau III-VI : Temps théoriques et simulés en millisecondes du premier frame d'inventaire pour 0, 10, 100, 256, 1000 tags

	Théorie (S, C, E constants)			Simulation (S, C, E dépendants de la simulation)			
	Moyenne min	Moyen	Moyenne max	Min	Moyen	Max	Ecart type
0 Tags	38,99	38,99	38,99	38,97	38,97	38,97	0
10 Tags	58,10	58,64	59,19	51,76	58,44	59,38	1,63
100 Tags	177,42	181,24	185,06	156,50	181,95	204,14	11,5
256 Tags	245,07	250,38	255,69	211,88	249,29	293,30	15,76
1000 Tags	145,89	147,02	148,15	127,51	148,05	168,17	7,45

La Figure III-18 reprend graphiquement le tableau précédent. On peut ainsi voir que les temps simulés moyens sont tous compris entre le minimum et le maximum des temps théoriques, en s'approchant tout de même assez de la moyenne.

Au cours de ce paragraphe, nous avons validé le fonctionnement du simulateur. Il est donc capable de simuler des systèmes RFID sains. Dans la section suivante, nous exposons les modèles de fautes qui seront injectés dans les systèmes sains afin d'observer les effets des défaillances identifiés lors de l'AMDE de la section III.1.

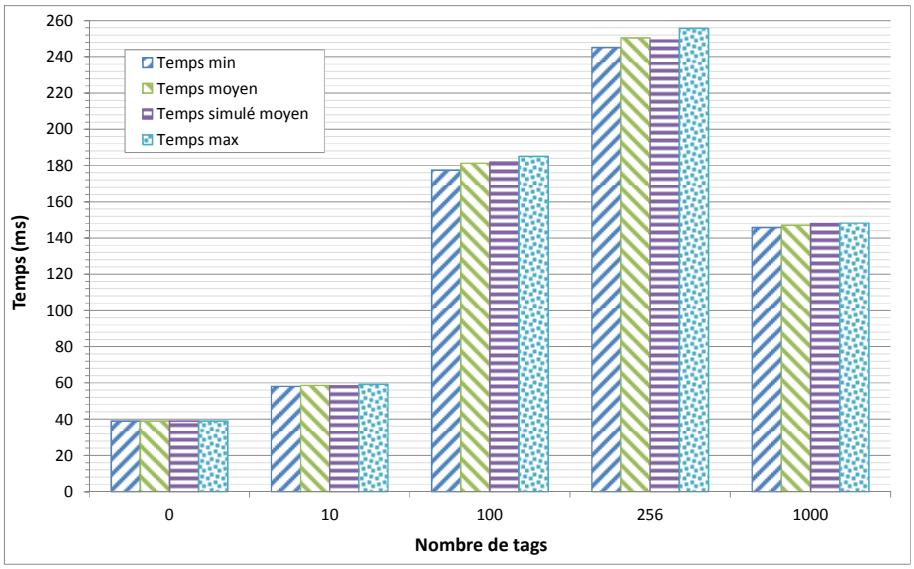


Figure III-18 : Temps simulé moyen comparé aux temps théoriques minimum, maximum et moyen

III.4 Modèle de fautes

Plusieurs modèles de fautes ont été proposés. Ces modèles de fautes s'intéressent principalement à la communication entre le tag et le lecteur. En effet, de par la nature du système, la communication est la partie la plus sensible.

Il existe trois méthodes d'injection de fautes [JEN94,MIS07] :

1. la méthode des *saboteurs* : un module d'injection de fautes supplémentaire est ajouté afin de modifier le comportement du système ;
2. la méthode des *mutants* : un composant est remplacé par un autre dont le comportement a été modifié ;
3. l'utilisation de *commandes spécifiques au simulateur* : ce sont des fonctionnalités du simulateur qui permettent alors l'injection de fautes.

Nous utilisons trois modèles de fautes différents :

1. désactivation puis réactivation des liens entre les tags et les lecteurs ;
2. aucune communication entre les tags et le lecteur alors qu'ils sont dans son champ ;
3. diminution de la qualité du lien existant entre les tags et le lecteur.

III.4.1 Premier modèle de fautes : désactivation et réactivation

Le premier modèle de fautes que nous utilisons s'appuie sur un *saboteur*. Ce saboteur a pour mission de désactiver puis réactiver des canaux de communication, donc des liens, entre les tags et les lecteurs à intervalle de temps régulier. La désactivation du canal est comparable à la sortie du tag du champ du lecteur : le tag ne reçoit plus alors ni énergie ni donnée. Ce modèle de fautes correspond directement aux modes de défaillances périodiques de l'AMDE suivants :

- « Couche physique » ou « Composants canaux et antennes » du tag ou du lecteur :
- Non réception des signaux par un ou plusieurs tag(s) ou lecteur(s) de manière périodique ;
- Non émission des signaux par un ou plusieurs tag(s) ou lecteur(s) de manière périodique.

Bien que d'autres modèles de fautes soient plus pertinents, ce premier modèle de fautes peut aussi, dans une moindre mesure, représenter les modes de défaillances suivants :

- Couche physique ou composant modulateur du tag ou du lecteur : absence de modulation de manière périodique – la puissance est transmise mais pas l'information ;
- Couche physique ou composant démodulateur du tag ou du lecteur : absence de démodulation de manière périodique – la puissance est reçue mais le composant n'est pas capable de détecter l'information contenue dedans;
- Composant alimentation du tag : non alimentation de manière périodique – les données sont reçues mais le tag n'est pas capable de récupérer la puissance nécessaire à son fonctionnement ;
- Couche application ou composant logique du tag : n'exécute aucune commande de manière périodique – la puissance et les informations contenues dans le signal sont récupérées par le tag, mais ignorées.

Ce modèle de fautes est caractérisé par trois paramètres : le nombre de tags/lecteurs concernés par la désactivation/réactivation de leur canal, la durée de désactivation du canal, et la période de désactivation/réactivation. Plusieurs variantes ont été implantées dans **SERFID** :

- 5 ou 10 % des tags affectés ;
- Période de 100ms avec une durée de désactivation de 50ms : cette période et cette durée de désactivation sont plus courtes que la durée d'un inventaire ;
- Période de 600ms avec une durée de désactivation de 300ms : la durée de désactivation est de l'ordre de grandeur de la durée d'un inventaire ;
- Période de 1200ms avec une durée de désactivation de 600ms : la durée de désactivation est alors plus longue qu'un inventaire.

III.4.2 Second modèle de fautes : communication impossible

Le second modèle de fautes s'appuie sur la norme intitulée *compatibilité électromagnétique (CEM)* [IEC61000-4-4]. Cette norme définit les essais de compatibilité électromagnétique permettant de valider le fonctionnement des appareils dans des environnements critiques. Elle définit, entre autres, la notion de *burst* : ces bursts, présentés sur la Figure III-19 doivent être appliqués sur les sources d'alimentation, les signaux ou les ports de contrôle. Ce sont des transitions répétitives et rapides d'une durée de 15ms, qui se répètent toutes les 300ms. Le système doit assurer son service, ou un service dégradé sécurisé, et recouvrer un service normal après que ces bursts aient été appliqués.

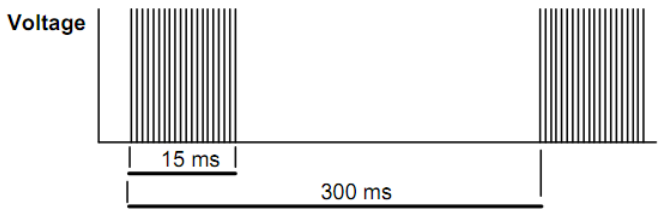


Figure III-19 : Transitions répétitives et rapides définies par la norme IEC-61000-4-4

Dans le cas de la RFID UHF, ces bursts bloquent complètement la communication. Il n'est donc plus possible d'envoyer de données pendant les 15ms que durent les bursts. Malgré cela, la télé-alimentation du tag est toujours opérationnelle. Ainsi, ce modèle de faute correspond aux modes de défaillances suivants :

- Couche physique ou composant antenne du tag ou du lecteur :
    - Non réception des signaux par tous les tag(s) et lecteur(s) de manière périodique ;
    - Non émission des signaux par un ou plusieurs tag(s) ou lecteur(s) de manière périodique ;
    - Emission continue.
- Couche physique ou composant modulateur du tag ou du lecteur : absence de modulation de manière périodique ;
- Couche physique ou composant démodulateur du tag ou du lecteur : absence de démodulation de manière périodique ;

De par la nature des bursts, ces derniers affectent la totalité des tags présents dans le champ du lecteur. Avec ce modèle de faute, il est possible d'étudier la robustesse du système face aux problèmes de CEM.

III.4.3 Troisième modèle de faute : diminution de la qualité du lien tag/lecteur

Le dernier modèle de fautes concerne la dégradation du lien entre les tags et les lecteurs. Elle apparaît sous la forme d'une augmentation du nombre d'erreurs binaires. Ce modèle de fautes correspond aux modes de défaillances suivants :

- Couche physique ou composant antenne du tag ou du lecteur :
    - Mauvaise réception des signaux par un ou plusieurs tag(s) ou lecteur(s) de manière permanente ;
    - Mauvaise émission des signaux par un ou plusieurs tag(s) ou lecteur(s) de manière permanente;
- Couche physique ou composant modulateur du tag ou du lecteur : mauvaise modulation de manière transitoire ;
- Couche physique ou composant démodulateur du tag ou du lecteur : mauvaise démodulation de manière transitoire ;
- Couche application ou composant logique du tag : n'exécute aucune commande de manière transitoire.

Ce modèle de fautes est caractérisé par le nombre de canaux affectés et par l'augmentation du taux d'erreur binaire (TEB). Lorsque l'augmentation du TEB est donnée par un intervalle de valeurs, l'augmentation réalisée est alors une valeur aléatoire comprise dans cet intervalle. Plusieurs variantes ont ainsi été implantées :

- 10% des tags sont touchés, avec une forte augmentation du taux d'erreur binaire :

$$TEB_{fautif} = TEB_{sain} + [5.10^{-3}; 7.10^{-3}] \quad \text{(III.5)}$$

- 10% des tags sont touchés, avec une augmentation du taux d'erreur binaire :

$$TEB_{fautif} = TEB_{sain} + [2.10^{-3}; 5.10^{-3}] \quad \text{(III.6)}$$

- 10% des tags sont touchés, avec une faible augmentation du taux d'erreur binaire :

$$TEB_{fautif} = TEB_{sain} + 2.10^{-3} \quad \text{(III.7)}$$

- 5% des tags sont touchés avec une augmentation moyenne du taux d'erreur binaire :

$$TEB_{fautif} = TEB_{sain} + [4.10^{-3}; 6.10^{-3}] \quad \text{(III.8)}$$

- 5% des tags sont touchés avec une faible augmentation du taux d'erreur binaire :

$$TEB_{fautif} = TEB_{sain} + 3.10^{-3} \quad \text{(III.9)}$$

- 2% des tags sont touchés avec une augmentation moyenne du taux d'erreur binaire :

$$TEB_{fautif} = TEB_{sain} + 5.10^{-3} \quad \text{(III.10)}$$

Les augmentations ont pour but d'affecter suffisamment la communication, sans pour autant l'empêcher. Ainsi, au vu du nombre de bits échangés, affecter un taux d'erreur binaire de plus de 0,7% engendrerait trop d'inversions de bits dans les messages, qui les rendraient alors tous erronés. Les modes de défaillances ainsi modélisés sont transitoires : le moment de leur apparition est aléatoire et non prévisible.

III.4.4 Injection automatique des fautes

Afin de faciliter l'injection de fautes dynamiquement dans le système, différentes commandes ont été ajoutées au serveur contenu dans le modèle du lecteur. Ces commandes s'ajoutent aux commandes permettant la gestion du lecteur – démarrer un inventaire, configurer l'inventaire applicatif … Ainsi, le serveur contenu dans le lecteur est aussi un point d'accès permettant la gestion du simulateur. Les différentes fautes décrites précédemment peuvent toutes être configurées et déclenchées en cours d'exécution du simulateur. Il est alors possible de développer des scénarii complexes mettant en œuvre des tags se déplaçant devant un ou plusieurs lecteurs et dont les fautes sont injectées dynamiquement.

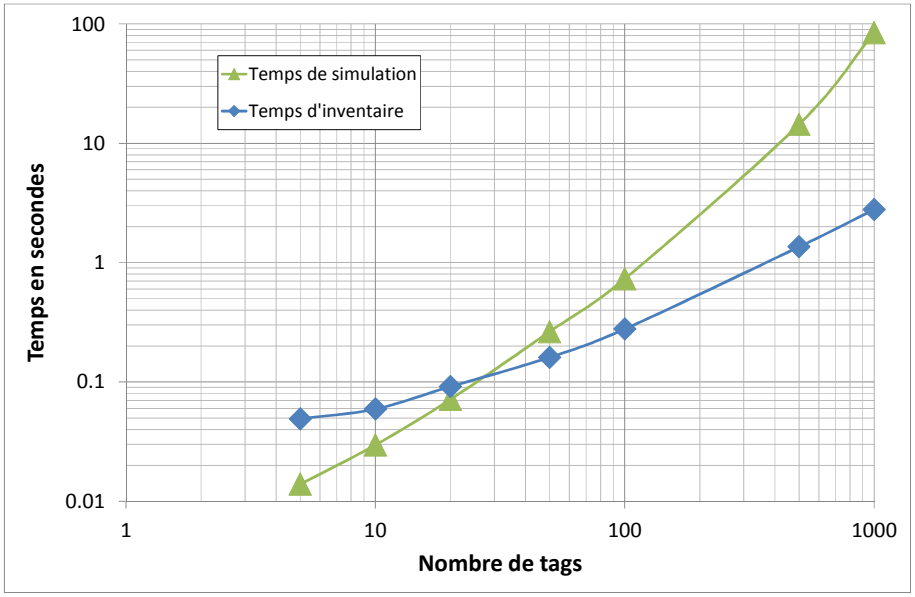


Figure III-20 : Temps de simulation et temps d'inventaire en fonction du nombre de tags

III.5 Temps de simulation

L'objectif du simulateur est la simulation de fautes pour l'évaluation de la robustesse des systèmes RFID et l'évaluation des méthodes de test en ligne. Les temps de simulations doivent donc être les plus courts possibles. La Figure III-20 donne les temps de simulation et

les temps d'inventaires théoriques en fonction du nombre de tags pour un inventaire applicatif composé d'un seul inventaire simple. Ces temps de simulations sont obtenus avec 30 tirages réalisés sur un serveur de calcul doté de 8 Go de mémoire vive et de 16 cœurs cadencés à 2,4GHz. On voit ainsi que, pour des simulations avec 100 tags et moins, les temps de simulations d'un inventaire sont proches de ceux d'un inventaire réel. Il est alors envisageable d'utiliser le simulateur afin de réaliser des campagnes de caractérisation.

III.6 Conclusion

Au cours de ce chapitre, une étude des défaillances des systèmes RFID a été réalisée. Cette étude, nommée *analyse des modes de défaillance et de leurs effets* ou *AMDE* [FRI10, FRI11c], montre qu'il est difficile de prédire les effets sur le système complet de fautes locales. Ainsi, le besoin d'un simulateur afin d'étudier ces effets, et afin de pouvoir proposer et évaluer une méthode de test en ligne est apparu clairement.

Ce simulateur s'appelle **SERFID** pour *Simulation et Evaluation des systèmes RFID* [FRI11a]. Ce simulateur – représentant au total un peu plus de 20 000 lignes de code – a été conçu pour modéliser le système dans son ensemble : plusieurs tags et plusieurs lecteurs communiquant avec un système hôte – middleware ou autre. Les différents aspects du système ont été modélisés :

- la communication analogique : les antennes, modulateurs et démodulateurs ; le lien spécifique entre le lecteur et le tag permettant l'échange de signaux analogiques transportant de l'énergie et de l'information ; l'environnement global dans lequel évolue le système, affectant ses performances ; les phénomènes physiques tels que la télé-alimentation ou encore les collisions ;
- la communication numérique : le protocole anticollision permettant l'inventaire des tags présents dans le champ du lecteur ;
- les composants internes du tag et du lecteur : mémoire, contrôle, contrôleur CRC, …
- la réception de requêtes en provenance d'un système hôte et l'envoi de réponses associées : démarrage d'un inventaire, configuration de l'inventaire.

Associé à ce simulateur, plusieurs modèles de fautes ont été proposées. Ces modèles de fautes représentent les défaillances identifiées lors de l'AMDE. Grâce à ce simulateur et à ces modèles de fautes, il est alors possible d'étudier les effets globaux sur le système des défaillances locales et ainsi de proposer des méthodes de test en ligne afin de les détecter.

IV Méthodes de test en ligne

Les méthodes de test en ligne sont des techniques permettant de détecter des comportements fautifs du système à partir de l'observation de celui-ci. Ces comportements erronés sont caractéristiques de la présence de faute et d'erreurs dans le système. Comme indiqué dans la section II.2, il existe deux types de méthodes de test en ligne : les méthodes non intrusives et les méthodes intrusives.

Nous nous intéresserons particulièrement aux méthodes non-intrusives. En effet, elles permettent l'observation du système sans en modifier le comportement. La détection des défaillances au cours de la vie du système est importante en RFID. Son importance apparaît triviale pour les systèmes RFID critiques utilisés notamment dans le nucléaire ou l'aéronautique, mais elle n'est pas moins importante pour les applications logistiques. En effet, l'apparition de défaillance entraîne l'arrêt de la chaîne logistique. Cet arrêt doit alors être contrôlé et réparé par un opérateur. Or, dans le contexte de la RFID dont le but est d'automatiser un maximum les opérations, le nombre d'opérateurs est réduit au minimum. Il faudra alors faire intervenir un opérateur devant se déplacer. Les temps d'arrêt deviennent anormalement longs, entraînant une perte de temps et d'argent importante.

Au cours du chapitre II, deux méthodes de test en ligne ont été présentées :

1. une méthode s'appuyant sur l'observation du volume de tags passant devant un point de lecture par unité de temps. Cette méthode s'appelle *volume moyen de trafic de tags*, ou *ATTV* pour « *Average Tag Traffic Volume* »;
2. une méthode s'appuyant sur l'observation du nombre d'erreurs de lecture à un point de lecture par unité de temps. Cette méthode s'appelle *nombre d'erreurs de lecture par rapport au nombre de lectures totales*, ou *RETR* pour « *Read Errors to Total Reads* ».

Ces deux méthodes sont utilisées afin de détecter une baisse de performance caractéristique de la présence de défaillances dans le système. Néanmoins, ces défaillances doivent avoir un impact important sur le système pour dégrader significativement ces deux mesures puisqu'elles doivent modifier les performances moyennes du système sur un temps donné, ce temps pouvant être assez long.

Nous nous concentrerons sur un cas d'étude particulier : un système RFID UHF composé d'un lecteur et de palettes homogènes composées de 110 cartons de 6 bouteilles métalliques contenant du liquide, sur lesquels sont fixés 110 tags. La palette est déposée sur un plateau tournant entre deux antennes afin de réaliser l'inventaire. Une antenne est utilisée pour l'émission du signal et l'autre antenne est utilisée pour la réception du signal. La vitesse de rotation de la plateforme est de 4,5 tours par minute. La Figure IV-1 est une photo de ce système. On peut y voir la palette de cartons sur la plateforme tournante ainsi que les deux antennes du lecteur. Un agrandissement des cartons montre l'emplacement et l'orientation des tags sur les cartons extérieurs. La position de chaque carton sur la palette et la position et l'orientation de chaque tag collé sur les cartons ont été optimisées afin d'obtenir les meilleurs taux de lecture possibles[1].

[1] Ce travail a été réalisé par Grenoble-INP RFTLab, laboratoire d'expertises et d'essais en CEM, RFID et radiofréquence, partenaire du projet SafeRFID dans lequel s'inscrit cette thèse.

Figure IV-1 : Représentation du système étudié

Dans de ce chapitre, nous étudions donc, sur ce système, les méthodes de test en ligne existantes et proposons une nouvelle approche. Pour cela, nous étudions dans un premier temps le comportement statistique du système vis-à-vis de la lecture des tags qui le composent. Puis, nous proposons une méthode de test en ligne basée sur ces observations. Enfin, nous comparons cette méthode aux méthodes de test en ligne décrites précédemment grâce notamment au simulateur décrit dans le chapitre précédent.

IV.1 Comportement statistique du système

La première étape de l'étude consiste à configurer le système afin d'obtenir les résultats les plus satisfaisants possibles : c'est-à-dire être capable au cours d'un inventaire applicatif[1], composé ici de 100 inventaires simples, de détecter au moins une fois tous les tags en un minimum de temps.

Ces paramètres ont été fixés de manière empirique : les influences sur les temps et les taux de lectures dans notre système ont été évaluées indépendamment pour chacun de ces paramètres ; puis, à partir de ces observations, les différents paramètres ont été fixés. Les valeurs utilisées dans notre application sont données dans le Tableau IV-I.

Tableau IV-I : Valeurs des paramètres de l'inventaire applicatif

Paramètre	Valeur
AcqG2Cycles	100
AcqG2Select	1
Masque	Sans masque : tous les tags doivent être inventoriés
AcqG2Count	3
Q	5

Une fois ces paramètres fixés, nous mesurons les taux de lectures de chacun des 110 tags pour plusieurs inventaires applicatifs. La Figure IV-2 montre les taux de lectures de chaque tag pour 3 inventaires applicatifs différents. On observe que ces taux de lectures sont très

[1] L'inventaire applicatif est défini dans le paragraphe III.2.5.1. Son algorithme est donné dans le Code III-1. Un inventaire applicatif consiste à exécuter un certain nombre de tentatives de lectures ou inventaires simples.

hétérogènes d'un tag à l'autre : en effet, bien que les conditions de fonctionnement soient correctes, qu'il n'y ait pas de perturbations et que le système ait été correctement paramétré, certains tags sont lus seulement 10 fois sur les 100 inventaires de l'inventaire applicatif alors que d'autres sont lus systématiquement lors de chaque inventaire simple. On observe aussi que les taux de lectures de chaque tag restent à chaque inventaire applicatif très proches : un tag lu dans 100% des tentatives de lecture – ou inventaires simples – lors d'un inventaire applicatif est lu aux alentours de 100% des inventaires simples de l'inventaire applicatif suivant ; tout comme un tag détecté dans 20% des inventaires simples lors d'un inventaire applicatif sera lui aussi inventorié environ 20% des tentatives lors de l'inventaire applicatif suivant.

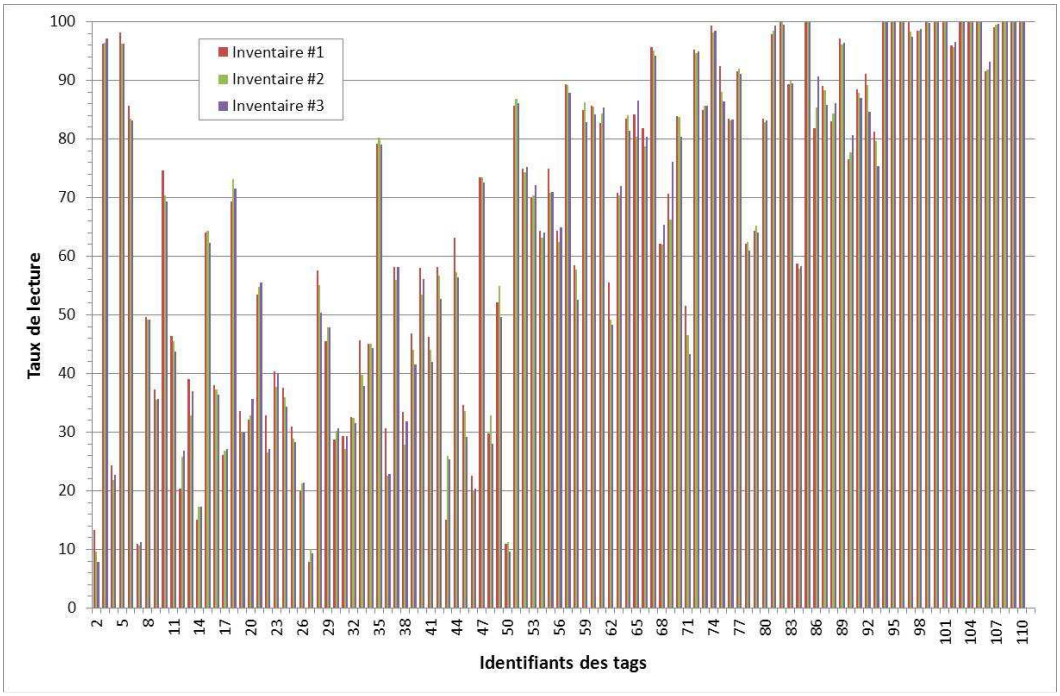


Figure IV-2 : Taux de lecture des tags de la palette

Les mêmes mesures ont été effectuées avec deux autres groupes de tags. Ces groupes de tags sont placés sur une palette identique à la première : les contraintes liées à l'environnement sont ainsi identiques pour tous 3 groupes de tags. Afin de comparer les différents inventaires, les taux de lectures de chaque inventaire applicatif ont été ordonnés par ordre décroissant. Les identifiants des tags, différents d'un groupe de tags à un autre, n'entrent pas en compte lors des comparaisons

La Figure IV-3 illustre les taux de lectures ordonnés des différents inventaires applicatifs réalisés sur les 3 groupes de tags – chaque couleur représentant un inventaire applicatif. On peut ainsi voir que ces taux lors des différents inventaires applicatifs ont la même forme : une quinzaine de tags sont détectés lors de tous les inventaires, puis les taux de lectures diminuent jusqu'à atteindre moins de 10 détections pour les tags les plus faiblement lus : seul 5 ou 10% des inventaires simples ont permis de récupérer les identifiants stockés dans ces tags. De plus, à un même index, les différents taux de lecture ordonnés sont de la même grandeur. Par exemple, quel que soit le groupe de tags, le taux de lectures classé 43ème sera de l'ordre de 82 détections sur 100 tentatives, avec un minimum de 80 détections et un maximum de 84 détections. L'écart type pour ce 43ème taux de lectures est de 1,2 : les différentes valeurs sont donc relativement proches les unes des autres.

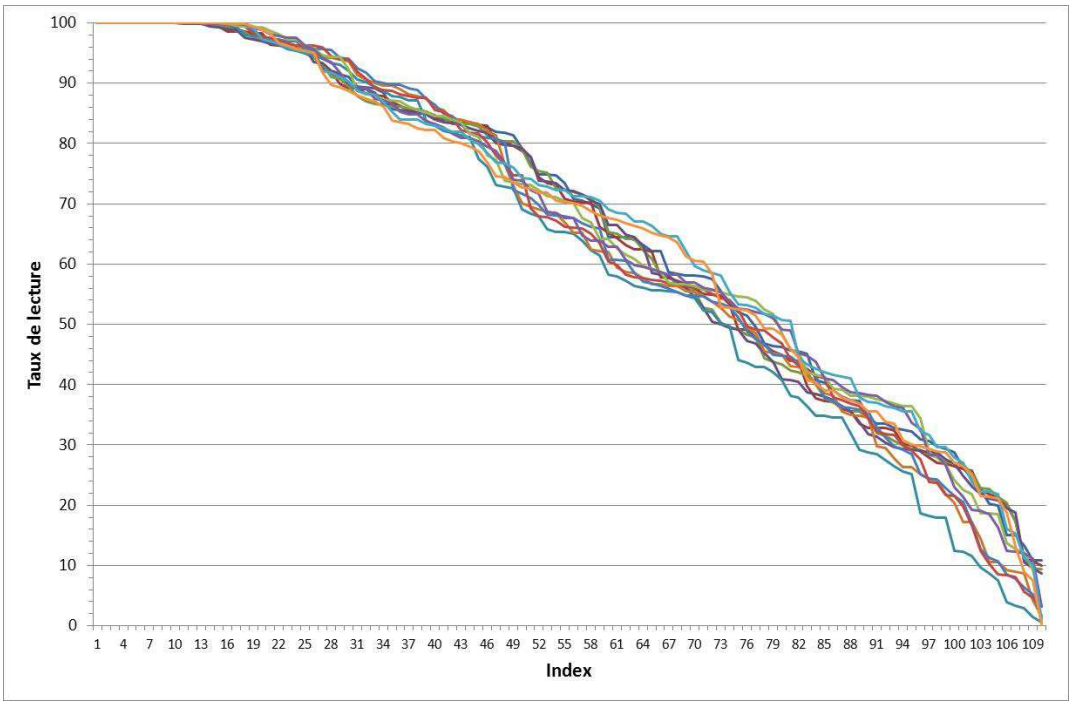


Figure IV-3 : Taux de lecture ordonnés des tags de la palette

Cette observation nous amène à introduire la notion de profil d'un système RFID. Cette notion est détaillée dans la suite du document.

IV.2 Méthode Profil

Le profil d'un système RFID est caractéristique de l'ensemble composé :

- d'un lecteur configuré ;
- d'un groupe de tags ;
- d'un environnement comprenant notamment les produits auxquels sont associés les tags et la disposition de ces derniers.

Le profil est calculé à partir des taux de lectures donnés par le lecteur. Le profil est l'ensemble ordonné décroissant des taux de lectures de l'ensemble des tags. Ainsi, la Figure IV-4 illustre le profil d'un inventaire applicatif. Le lien existant entre l'identifiant du tag et son taux de lecture est alors totalement absent en effet, le résultat d'inventaire tel qu'on peut le voir sur la Figure IV-2 montre des taux de lectures non ordonnés, mais classés par identifiant de tag, alors que sur la Figure IV-3, les taux de lectures sont ordonnés, sans information sur l'identifiant des tags associés. Néanmoins, cette courbe permet de comparer l'évolution du taux de lectures de chaque tag au cours de la vie du système. En outre, l''observation du profil permet d'évaluer précisément les performances du système. En effet, si l'unité de temps choisie pour les méthodes RETR et ATTV est le temps d'un inventaire applicatif, il est alors possible, à partir du profil de retrouver à la fois la valeur associée du RETR et de l'ATTV :

$$RETR = \frac{\sum_{i=1}^{N}(100-e_i)}{100.N} \quad \text{(IV.1)}$$

$$ATTV = N \quad \text{(IV.2)}$$

Où $\boldsymbol{e_i}$ représente le $\boldsymbol{i}$-ème taux de lecture au cours d'un inventaire applicatif et N le nombre total de tags présentant au moins une détection au cours d'un inventaire applicatif.

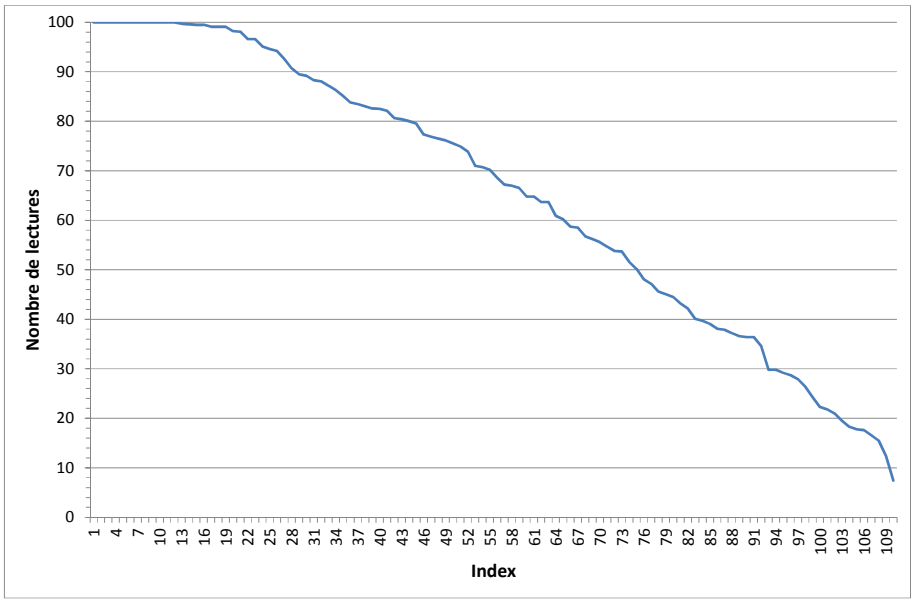


Figure IV-4 : Profil d'un inventaire applicatif

IV.2.1 Test en ligne à partir de l'observation du profil

La méthode proposée a pour objectif la détection de défaillances dans le système à partir de l'observation de ses performances. Elle s'appuie, pour cela, sur l'observation du profil d'un ensemble tags-lecteur-environnement.

La première étape est l'étude du système afin de déterminer le comportement moyen de celui-ci. Un profil moyen est alors calculé à partir des M profils obtenus lors de la campagne de mesures. Il est calculé grâce à l'équation suivante :

$$profil_{moyen} = \left\{ \bar{n}_i \;/\; \bar{n}_i = \frac{1}{M} \sum_{j=1}^{M} e_{i_j} \right\} \qquad \text{(IV.3)}$$

Où e_{i_j} correspond au ***i***-ième taux de lectures du ***j***-ième profil et n_i à la moyenne des ***i***-ième taux de lecture des M profils. Pour chaque index, l'écart type est aussi calculé :

$$\sigma_{profils} = \left\{ \sigma_i \;/\; \sigma_i = \sqrt{\frac{1}{n} \sum_{j=1}^{M} (e_{i_j} - \bar{n}_i)} \right\} \qquad \text{(IV.4)}$$

Où e_{i_j} correspond au ***i***-ième taux de lectures du ***j***-ième profil, n_i à la moyenne des ***i***-ièmes taux de lectures des M profils et où σ_i correspond à l'écart type des i-ièmes taux de lectures des M profils.

Le Tableau IV-II donne des exemples de résultats de campagnes de mesures. Par souci de clarté, tous les résultats n'ont pas été présentés. Mais à des fins d'illustrations, certains de ces taux de lectures – le plus élevé, le 61^{e}, 62^{e} et 63^{e} ainsi que le plus faible – sont mis dans le Tableau IV-II.

Ces taux de lectures ont été classés par ordre décroissant. Ainsi, on retrouve dans ce tableau le profil des différents inventaires applicatifs. Les deux dernières colonnes correspondent au profil moyen et à l'écart type de chaque index des profils. On peut voir que dans ce cas pratique, l'inventaire applicatif numéro 2 et l'inventaire applicatif numéro *i* ont leur plus faible taux de lectures égal à 0 : il y a donc un tag non détecté dans ces inventaires applicatifs. En effet, malgré la configuration du lecteur et l'absence de défaillance, il arrive que certains tags ne soient pas détectés. Ceci est dû aux caractéristiques intrinsèques du système : la communication est sujette à de nombreuses erreurs et il arrive qu'il faille plus de 100 inventaires pour observer au moins une fois certains tags faibles. Ces erreurs de communication sont amplifiées par l'environnement : en effet, les tags enfouis au milieu de la

palette présentent de nombreux obstacles entre eux et le lecteur. Les deux dernières lignes du tableau montrent le calcul du taux d'erreur de lecture sur le taux de lecture total – RETR en prenant comme unité de temps la durée d'un inventaire applicatif – ainsi que le nombre moyen de tags détectés – ATTV en prenant comme unité de temps la durée d'un inventaire applicatif.

Tableau IV-II : Profils des inventaires, profil moyen et écart type

	Inventaire 1	Inventaire 2	…	Inventaire *i*	Inventaire *j*	…	Moyenne	σ
1er nombre de lectures	100	100	…	100	100	…	100	0
…	…	…	…	…	…	…	…	…
61e nombre de lectures	63	62	…	62	63	…	61,12	3,44
62e nombre de lectures	62	58	…	58	58	…	59,08	3,03
63e nombre de lectures	58	57	…	56	57	…	57,94	2,89
…	…	…	…	…	…	…	…	…
*k*e nombre de lectures	8	0	…	0	7	…	3,11	3,89
RETR	33,49	34,24	…	34,38	34,64	…	34,80	1,27
ATTV	110	109	…	109	110	…	109,23	0,73

A partir de ces mesures, il est alors possible de définir un profil limite. Ce profil limite permettra, par comparaison, de déterminer si un système est défaillant ou non.

Afin de déterminer une méthode de calcul de cette limite, il est intéressant de connaître la loi statistique que suivent les taux de lectures. [RAK11] propose des méthodes afin de vérifier la normalité d'une distribution. Il propose, par exemple, un outil graphique permettant de vérifier si une distribution suit une loi normale. Cet outil s'appelle la droite de Henry, du nom de son concepteur. A partir de la distribution, une courbe est tracée. Lorsque la distribution suit totalement une loi normale, la courbe résultant de l'analyse est alors une droite. La méthode de calcul des droites de Henry est donnée en annexe I.2. La Figure IV-5 donne 4 exemples de droites de Henry pour les inventaires applicatifs expérimentaux. Ces droites de Henry sont calculées à partir des inventaires ordonnés. Compte tenu de la difficulté de réaliser une grande campagne de mesures, ces courbes ont été obtenues avec peu de valeurs. Toutefois, nous pouvons tout de même considérer que les nombres de lectures suivent des lois normales.

En admettant que la distribution des nombres de lectures est normale, le profil limite est défini par l'équation suivante :

$$profil_{limite} = \{l_i \,/\, l_i = \bar{n}_i - 3.\sigma_i\} \qquad \text{(IV.5)}$$

où l_i est la i-ième valeur limite, $\bar{n}_i$ est la i-ème moyenne du profil moyen et σ_i est l'écart type des i-ièmes nombres de lectures de tous les profils.

En effet, en connaissant la moyenne et l'écart type d'une distribution suivant une loi normale, il est possible de déterminer l'intervalle dans lequel se trouve 99,7% de la population – Figure IV-6. Ainsi 99,7% de la population se trouve dans l'intervalle$[\bar{n} - 3.\sigma; \bar{n} + 3.\sigma]$. Le profil limite est alors calculé comme étant la borne inférieure de cet intervalle.

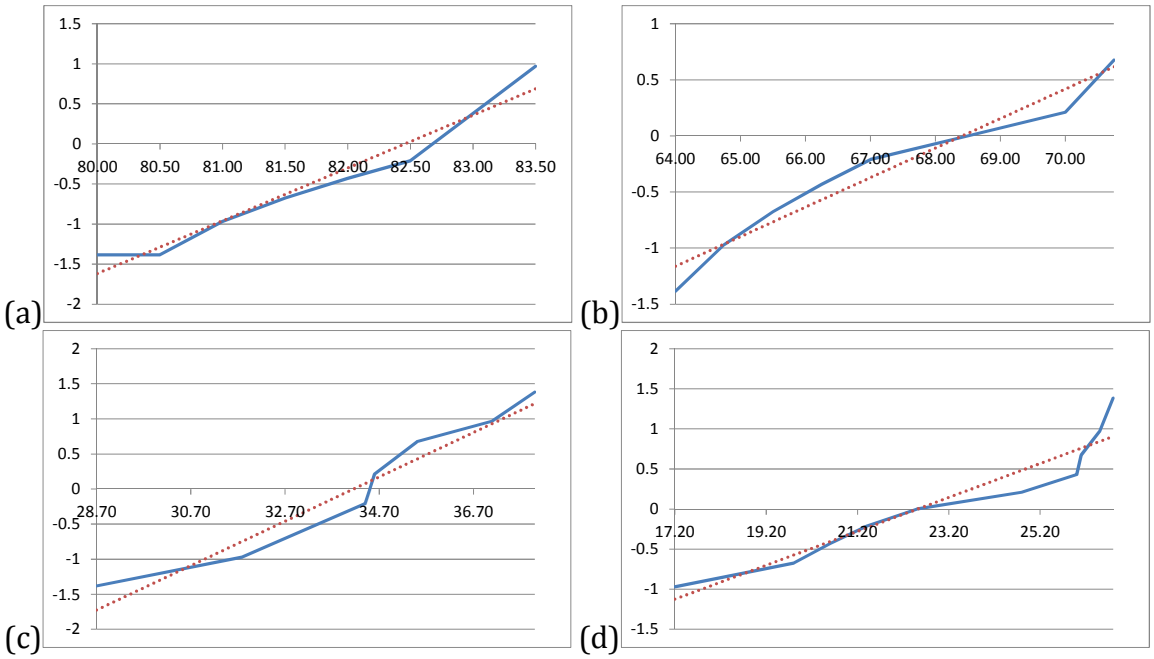


Figure IV-5 : Droites de Henry pour les (a) 43e nombre de lectures, (b) 57e nombre de lectures, (c) 90e nombre de lectures et (d) 101e nombre de lectures

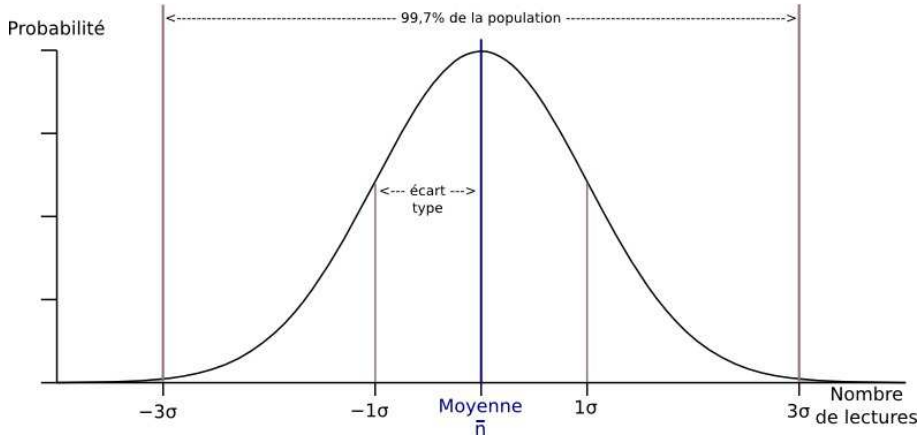


Figure IV-6 : Répartition de la population suivant la loi normale

De par les propriétés gaussiennes du système et de par la construction du profil limite, chaque point de ce dernier est en dessous de 99,7% des points correspondants des différents profils. Un profil est dit « en dessous d'un autre » si au moins un de ces taux de lectures est en dessous du taux de lectures au même index de l'autre profil. Ainsi, si un profil est en dessous, il y a donc une grande probabilité que celui-ci soit le profil d'un système défaillant. En effet, comme présenté au chapitre précédent dans la section III.4, les défaillances que nous étudions ont comme effet sur le système de diminuer les taux de lectures des tags, et donc leur taux de lectures au cours d'un inventaire applicatif. Ainsi, les taux de lectures composant un profil vont diminuer en présence de défaillances. Les profils des systèmes fautifs auront tendance à être en dessous du profil limite. En effet, si un taux de lectures diminue, il est alors, de par la construction du profil, déplacé à un index plus grand, dû au classement décroissant des taux de lectures. Tous les points entre sa position initiale incluse et sa nouvelle position non incluse sont alors décrémentés d'un index. Ainsi, à ce nouvel index i, compris entre la position initiale du taux de lectures impacté par la défaillance et sa nouvelle position, alors que la probabilité que le profil soit en dessous de la limite était très faible – de l'ordre de 0,3%

- elle est maintenant beaucoup plus grande comme le montre la Figure IV-7. En effet, on voit sur cette figure les courbes gaussiennes du taux de lectures i original et du taux de lectures impacté. La probabilité d'être en dessous de la limite est alors égale à la zone rouge. De plus, une diminution entraine plusieurs décalages, ce qui permet d'augmenter la probabilité qu'au moins un des points passe en dessous de la limite. Plus cette diminution est importante et survient dans les premiers nombres de lectures, plus le nombre de décalage est grand et plus il y a de probabilité que le décalage soit détecté.

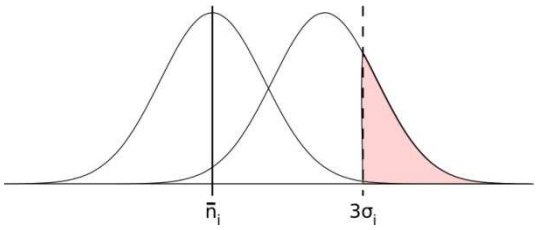


Figure IV-7 : Chevauchement de gaussiennes

La Figure IV-8 permet de comparer le profil limite – en bleu, trait plein – avec un profil d'un système sain – en vert, pointillés – et un profil d'un système défaillant – en rouge, tirets. On voit ainsi que tous les taux de lectures composant le profil du système sain sont au-dessus des taux de lectures du profil limite alors que quelques points du profil du système défaillant sont en dessous. Au niveau de l'index 24, il y a alors dépassement de la limite. On peut aussi observer que la limite n'est pas ordonnée. En effet, les points du profil limite sont calculés indépendamment les uns des autres à partir de l'observation des points au même index des profils du système : ils dépendent donc uniquement de la variabilité des taux de lectures à l'index donné.

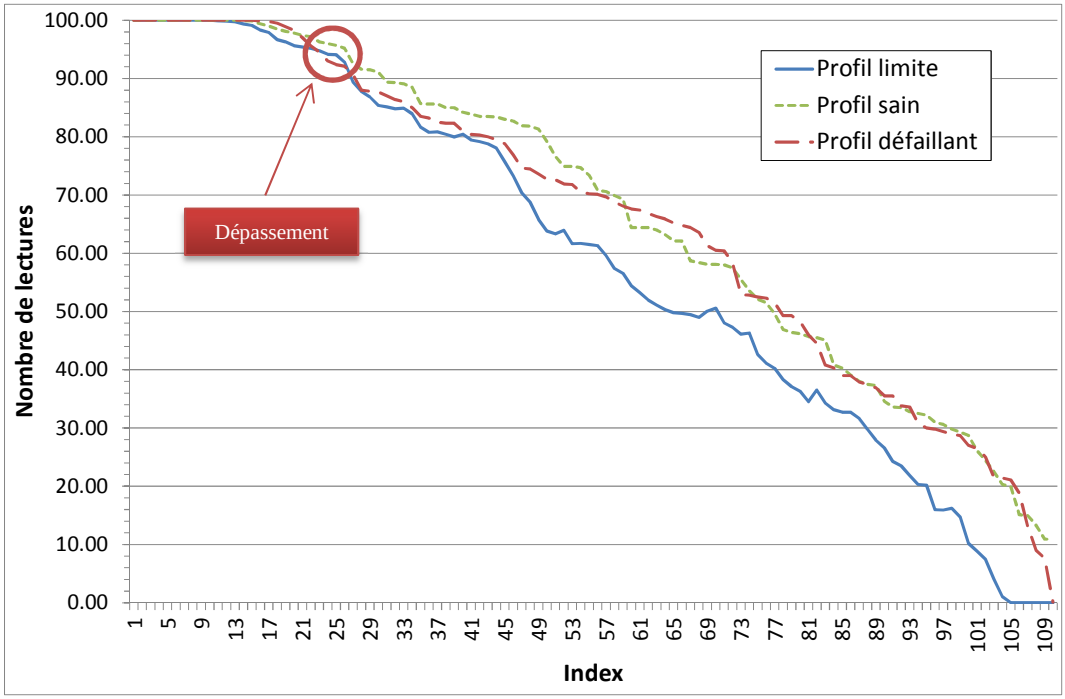


Figure IV-8 : Comparaison entre le profil limite, un profil sain et un profil défaillant

Nous venons de proposer une nouvelle méthode de test en ligne basée sur l'observation des performances du système à travers la définition d'un profil. Les sections suivantes vont

s'intéresser à l'évaluation de cette méthode et sa comparaison avec les méthodes existantes ATTV et RETR.

IV.3 Evaluation de la méthode profil

L'évaluation de la méthode profil a été réalisée par deux démarches : une première démarche expérimentale et une seconde par simulation. La première méthode a permis de valider par la pratique le fonctionnement du test en ligne proposé et de comparer ce test aux tests en ligne classiques. Néanmoins, il est complexe de mettre en place de nombreux scénarii en pratique, à la fois pour des questions de manipulation – poids et encombrement d'une palette de 110 cartons – mais aussi à cause de la difficulté d'injecter des fautes variées. La seconde méthode par simulation a donc permis de valider cette méthode de test en ligne sur plusieurs systèmes différents avec un grand nombre d'injections de fautes.

IV.3.1 Evaluation expérimentale

Nous réalisons l'étude du système présenté au début du chapitre. Cette étude nous permet de récolter un grand nombre de profils sur trois palettes différentes. Le Tableau IV-II présenté précédemment donne quelques résultats issus de cette campagne de mesures. A partir de ces mesures, nous avons calculé le profil limite associé à ce système homogène – le nombre et le type de tags restent identiques d'une palette à une autre, le même lecteur est toujours utilisé avec la même configuration et l'environnement dans lequel évoluent tous les équipements reste inchangé. Le calcul du profil limite est réalisé à l'aide de la formule (IV.5).

L'étude précédente permet aussi de déterminer les limites des méthodes RETR et ATTV. Nous choisissons comme unité de temps d'observation pour ces méthodes la durée d'un inventaire applicatif (l'inventaire des tags sur une palette). Les observations ainsi faites restent cohérentes et comparables car le système étudié est composé de palettes homogènes.

La méthode RETR revient à calculer le taux d'erreurs moyen de lecture qui se sont produites durant l'inventaire applicatif d'une palette. A partir des taux d'erreurs de lecture des différents inventaires du système sans faute, il est possible de calculer une moyenne $\bar{n}_{RETR}$ et un écart type σ_{RETR}. La Figure IV-9 montre la droite de Henry de la distribution des RETR que nous avons mesurés. On peut ainsi en déduire que les taux d'erreurs de lecture d'un inventaire applicatif suivent une loi normale.

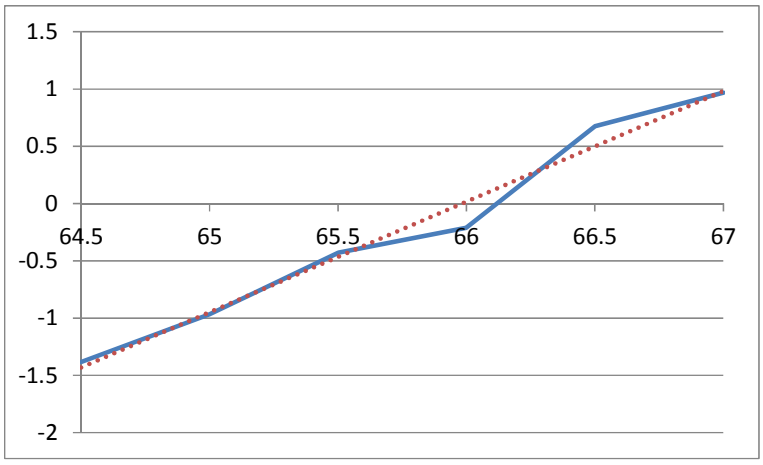


Figure IV-9 : Droite de Henry pour la distribution des erreurs de lecture - méthode RETR

Ainsi, pour calculer la limite de la méthode RETR de manière à ce que 99,7% des RETR mesurés soient au-dessus de cette dernière, nous utilisons la formule suivante :

$$limite_{RETR} = \bar{n}_{RETR} - 3.\sigma_{RETR} \quad \text{(IV.6)}$$

Le rapport limite entre le nombre d'erreurs et le nombre total de lectures est donc fixé à $\frac{\text{nombre d'erreur}}{\text{nombre total de lecture}} = 38{,}24\%$. Si le taux d'erreur moyen dépasse cette valeur alors cela signifie que l'on détecte un problème.

Lorsque l'unité de temps d'observation est la durée d'un inventaire applicatif, la méthode ATTV revient à étudier le nombre de tags détectés au cours de l'inventaire applicatif. L'étude préliminaire du système montre que les 110 tags présents sur la palette sont tous détectés dans 10 cas sur 12 – soit 83% des cas – et qu'un tag seulement n'est pas détecté dans les 2 cas restants. La limite devant être fixée en dessous de tous les cas sains, nous la fixons à 109 : lorsque le système détecte moins de 109 tags, nous considérons qu'il y a une défaillance dans le système.

Le système est maintenant modifié afin d'y introduire des dysfonctionnements. Les fautes injectées dans le système réel impactent l'environnement du système : en effet, il est difficile d'introduire en pratique des fautes matérielles. Cela nécessiterait la conception d'équipements spécifiques capables de reproduire un fonctionnement matériel erroné. Ainsi, les fautes injectées sont les suivantes :

1. Tourner de 90 degrés 5 tags choisis aléatoirement : ainsi ces 5 tags ne seront plus correctement alignés avec les antennes du lecteur. Ceci engendre une communication de mauvaise qualité, et change donc les performances de ces tags ;
2. Modifier l'emplacement de 5 tags choisis aléatoirement : l'environnement de ces tags ainsi que de leurs voisins sont ainsi modifiés, engendrant un fonctionnement différent du système ;
3. Modifier l'emplacement de 15 tags choisis aléatoirement : l'environnement du système est alors davantage modifié que dans le cas précédent ;
4. Modifier l'emplacement de 21 tags choisis aléatoirement : augmentation de la modification de l'environnement du système par rapport à la faute précédente ;
5. Arrêter la rotation de la palette pendant 15 secondes : le trajet des tags est alors modifié et les performances des communications sont alors diminuées ;
6. Arrêter la rotation de la palette pendant 20 secondes.

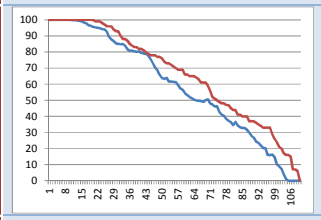
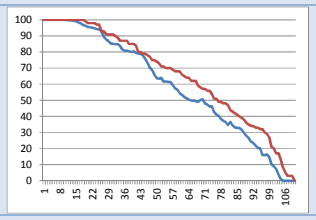
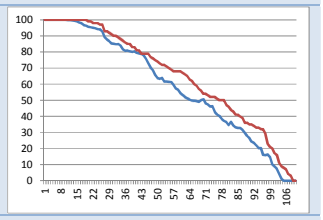
Les Tableau IV-III, Tableau IV-IV et Tableau IV-V donnent les résultats d'observations de ces expérimentations. Le système 1 correspond à la première faute décrite précédemment, le système 2 à la seconde, le système 3 à la troisième, *etc.* Plusieurs essais ont été réalisés pour les systèmes 1 et 2. Pour chaque système et chaque essai, le nombre de tags affectés par une diminution du taux de lectures supérieure ou égale à 10 est indiqué. L'impact sur le taux d'erreur global est aussi indiqué. Ces paramètres permettent d'observer l'impact des différentes modifications sur le système : plus le nombre de tags affectés et l'augmentation du taux d'erreurs de lectures sont élevés, plus les fautes injectées affectent le système. Les tableaux sont présentés dans l'ordre d'importance des fautes : les fautes impactant faiblement le système sont présentées en premier lieu alors que les fautes impactant fortement le système sont présentées à la fin.

Le nombre de tags détectés au cours de l'inventaire applicatif est aussi indiqué. Cette information correspond à la mesure associée à la méthode ATTV. En effet, lorsque le nombre de tags détectés est inférieur à 109, la méthode ATTV détecte la présence d'une défaillance.

Un graphique permet la comparaison du profil de l'inventaire applicatif, en rouge, avec le profil limite, en bleu. Le nombre de points du profil de l'inventaire applicatif en dessous du profil limite est indiqué sous le graphique. D'après la définition de la méthode, il suffit qu'un

seul point soit en dessous de la limite pour détecter la présence d'un dysfonctionnement dans le système.

Tableau IV-III : Comparaison des méthodes de test en ligne - résultats expérimentaux – Injections de fautes 1

	Système 1		
Nombre de tags dont le taux de lecture diminue d'au moins 10%	12	17	19
Augmentation du taux d'erreurs	-0,48	0	0,41
Nombre de tags détectés	109	109	108
Profil vs Profil limite			
Points du profil en dessous de la limite	0	0	0
ATTV			X
RETR			
Profil			

Les différents essais concernant la première injection de fautes sont présentés dans le Tableau IV-III. Durant le premier essai, 5 tags ont été choisis et leur orientation par rapport aux antennes du lecteur ont été modifiées : ils ont été tournés de 90°. Lors des seconds et troisièmes essais (2ème et 3ème colonnes), un second groupe de 5 tags a été tourné après que le premier groupe ait été remis en place. On peut ainsi voir que cette faute affecte peu le système : bien qu'il y ait plus de 10% des tags dont le taux de lectures diminue significativement, les taux d'erreurs de lecture durant les inventaires applicatifs sont restés inchangés. Ceci est dû au fait, qu'en modifiant l'environnement des tags, les paramètres de la communication ont changés : certains tags ont donc vu leurs taux de lecture augmenter, équilibrant alors avec les tags dont le taux de lecture a diminué. Seul le dernier essai est détecté comme défaillant par la méthode ATTV.

Le Tableau IV-IV donne les résultats pour deux essais (1ère et 2ème colonnes) pour la seconde injection de fautes et pour un seul essai pour la troisième injection de fautes : 5 tags puis 15 tags ont respectivement été déplacés. Déplacer 5 tags a un effet un peu plus conséquent que la première faute : le nombre de tags affectés significativement reste du même ordre de grandeur, mais le taux d'erreurs de lecture a diminué, améliorant alors le comportement global du système. En effet, les tags agissent comme obstacle lors de la lecture d'autres tags. Ici, le déplacement des tags a permis de supprimer des éléments perturbateurs pour d'autres tags. Un essai sur deux a été détecté par la méthode ATTV. Le déplacement de 15 tags affecte grandement le système : un quart des tags ont été affectés significativement et le taux d'erreurs de lecture a fortement augmenté. Ces augmentations du taux d'erreurs de lecture et du nombre de tags impactés font que le profil du système a été suffisamment

impacté pour transcrire la diminution des performances du système malgré la détection de tous les tags.

Tableau IV-IV : Comparaison des méthodes de test en ligne – résultats expérimentaux – Injections de fautes 2 et 3

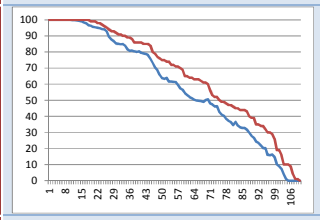
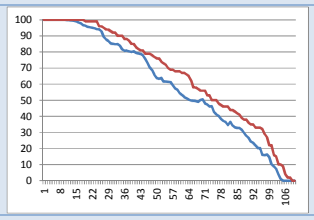
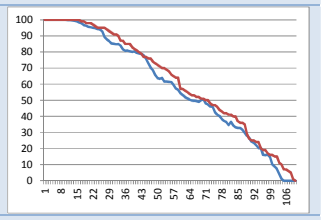
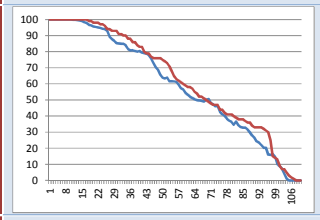
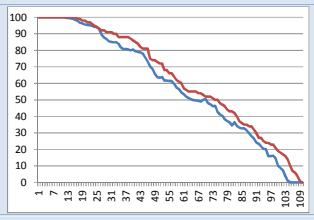
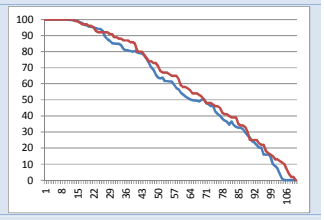
	Système 2		Système 3
Nombre de tags dont le taux de lecture diminue d'au moins 10%	14	15	28
Augmentation du taux d'erreurs	-1,17	-0,57	2,56
Nombre de tags détectés	108	109	109
Profil vs Profil limite			
Points du profil en dessous de la limite	0	0	3
ATTV	X		
RETR			
Profil			X

Tableau IV-V : Comparaison des méthodes de test en ligne – résultats expérimentaux – Injections de fautes 4, 5 et 6

	Système 4	Système 5	Système 6
Nombre de tags dont le taux de lecture diminue d'au moins 10%	29	25	34
Augmentation du taux d'erreurs	2,48	1,66	3,28
Nombre de tags détectés	107	109	109
Profil vs Profil limite			
Points du profil en dessous de la limite	7	1	13
ATTV	X		
RETR			
Profil	X	X	X

Finalement, le Tableau IV-V donne les résultats des expérimentations pour les trois dernières fautes : déplacement de 21 tags, arrêt de la rotation de la palette pendant 15 puis 20 secondes. Dans ces trois derniers cas le système est fortement impacté : 25 à 30% des tags ont leur taux de lectures ayant diminué significativement et le nombre d'erreurs de lecture a fortement augmenté. Lors du déplacement des 21 tags, le lecteur n'est plus capable de détecter 3 tags du système, probablement en raison d'une position ou d'un environnement peu propice à la communication. Ainsi, le profil et le nombre de tags détectés (ATTV) ont été fortement impactés permettant à ces deux méthodes de détecter la faute présente dans le système. Lors de l'arrêt de la rotation de la palette pendant 15 secondes, tous les tags ont pu être détectés au moins une fois, mais de manière générale, tous les tags ont moins biens été détectés, engendrant une diminution du profil global du système. Il en va de même pour l'arrêt de la rotation de la palette pendant 20 secondes.

La Figure IV-10 résume les résultats obtenus : la méthode RETR n'a détecté aucune modification du système ; la méthode ATTV a détecté 3 fautes sur les 9 injectées, dont 2 qui n'ont été détectées par aucune autre méthode ; et la méthode Profil a détecté 4 fautes sur les 9 injectées, dont 3 qui n'ont été détectées par aucune autre méthode. De plus, cette figure illustre la complémentarité de la méthode Profil avec les méthodes ATTV et RETR. Ainsi, en utilisant conjointement les deux méthodes Profil et ATTV, 6 fautes sur 9 ont été détectées.

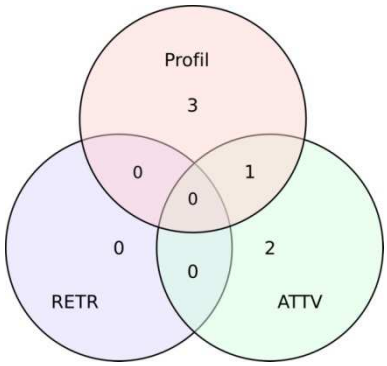


Figure IV-10 : Complémentarité des méthodes de test en ligne

Ces expérimentations ont permis de prouver que la méthode profil est une méthode détectant plus de cas de dysfonctionnements que les autres méthodes. Néanmoins, il apparaît aussi assez clairement qu'il est plus intéressant d'utiliser conjointement toutes ces méthodes.

Ces expérimentations permettent d'avoir une première évaluation comparative des différentes méthodes de test en ligne. Malheureusement, l'injection de fautes dans un système réel aussi complexe est difficilement contrôlable : les différentes défaillances identifiées lors de l'AMDE de la section III.1 ainsi que leurs effets ou leurs causes sont difficilement reproductibles sans matériel conçu spécifiquement. Il est de plus difficile de prédire l'impact d'une modification du système sur les performances de ce dernier : on a ainsi pu voir que le déplacement de quelques tags pouvait faire diminuer le nombre global d'erreurs de communication alors que certains tags étaient moins souvent détectés. De plus, on a aussi vu que le changement opéré sur un petit groupe de tags affectait aussi les tags voisins en modifiant leur environnement. De plus, il est complexe d'injecter des fautes car cela nécessite beaucoup de manipulations d'équipements lourds et encombrants. Pour ces raisons, et afin d'obtenir suffisamment de données sur différents systèmes en présence de différentes fautes pour conclure sur l'intérêt et l'utilisation possible de ces méthodes de test en ligne, il devient nécessaire de réaliser des évaluations par simulation.

IV.3.2 Evaluation par simulation

Afin d'évaluer et de comparer les différentes méthodes de tests sur plusieurs systèmes en présence de différentes fautes, nous avons réalisé différentes simulations. Les premières simulations simplistes, réalisées à l'aide d'Excel, permettent de conclure quant aux cas d'utilisations possibles de la méthode. Ensuite, au vu de la complexité du système et des interactions existantes entres les différents éléments le composant, nous utiliserons **SERFID** afin de réaliser des scénarios de tests plus complexes permettant de conclure quant aux performances de ces méthodes de tests.

IV.3.2.1 Première simulation – Excel

Dans un premier temps, nous avons réalisé des simulations simples ne prenant pas en compte les interactions existantes entre les tags. L'objectif est ici d'avoir un premier aperçu des limites de la méthode profil et d'effectuer une première comparaison de celle-ci avec les méthodes ATTV et RETR.

Pour cela, nous avons repris un profil sain obtenu par expérimentation puis nous y avons injecté des fautes. Les fautes injectées ici ont été déterminées à partir des observations réalisées lors des expérimentations ainsi que des conclusions faites lors de l'analyse des modes de défaillances de la section III.1. En effet, nous avons vu que la majorité des défaillances dans les systèmes RFID se traduisent par la diminution du taux de lectures d'un certain nombre de tags. Nous avons donc réalisé deux injections de fautes différentes :

1. le taux de lecture de 5 tags choisis aléatoirement parmi les 110 tags est diminué de 40% ;
2. le taux de lecture de 20 tags choisis aléatoirement parmi les 110 tags est diminué de 10%.

Chaque faute est injectée 100 fois aléatoirement. Le Tableau IV-VI donne le nombre de détections réalisées pour chaque méthode pour les deux types d'injections de fautes réalisées. On peut ainsi voir que la méthode profil détecte plus de cas fautifs que les autres méthodes. On remarque aussi que la méthode RETR a détecté des défaillances. Le taux de détection étant très faible – 4% –, ceci explique pourquoi, lors des expérimentations, la méthode RETR n'a détecté aucune défaillance. De plus, de par la simplicité de la simulation ne prenant pas en compte les différentes interactions existantes entre les tags, le taux d'erreur de lecture est toujours diminué lors de l'injection de faute. Or, on a vu que par simulation, il arrivait qu'une faute améliorait globalement le taux d'erreur de lecture alors que certains tags étaient grandement affaiblis. Quant à la méthode ATTV, les simulations confirment les résultats obtenus par expérimentations : bien que moins performante que la méthode profil, elle en est complémentaire. En effet, elle détecte des cas que la méthode profil ne détecte pas. Ainsi, en utilisant conjointement les différentes méthodes, il est possible de détecter plus de cas fautifs. Ce résultat peut être confirmé théoriquement. En effet, la limite est fixée de manière à déclarer la présence d'une défaillance lorsqu'au moins deux tags sont absents de l'inventaire applicatif. Il faut donc que l'injection de fautes se fasse sur au moins 2 tags dont le taux de lecture est inférieur à la faute injectée. D'après le profil moyen, il y a 26 tags dont le taux de lecture est en dessous de 40% et il y a 3 tags dont le taux de lecture est en dessous de 10%. En utilisant les formules classiques de dénombrement et de probabilité, il est alors possible de déterminer la probabilité qu'une injection de faute engendre la disparition d'au moins deux tags. Pour la première injection, cela revient à calculer la probabilité que deux tags, trois tags, quatre tags ou cinq tags avec un taux de lecture de moins de 40% soient affectés par l'injection de faute :

$$p_1 = \frac{nombre\ de\ cas\ possible}{nombre\ de\ cas\ total}$$

$$= \frac{\binom{26}{2}.\binom{84}{3}+\binom{26}{3}.\binom{84}{2}+\binom{26}{4}.\binom{84}{1}+\binom{26}{5}.\binom{84}{0}}{\binom{110}{5}}$$
$$= 0{,}34 \quad \text{(IV.7)}$$

Pour la seconde injection de faute, la probabilité revient à calculer la probabilité que deux tags ou trois tags permis les 3 tags ayant un taux de lecture inférieur à 10% soient affectés par l'injection de faute :

$$p_2 = \frac{nombre\ de\ cas\ possible}{nombre\ de\ cas\ total}$$
$$= \frac{\binom{3}{2}.\binom{107}{18}+\binom{3}{3}.\binom{107}{17}}{\binom{110}{20}}$$
$$= 0{,}09 \quad \text{(IV.8)}$$

On retrouve ainsi les taux de détection de la méthode ATTV. Les différences sont dues au fait que les injections de fautes ont été réalisées sur les profils obtenus par expérimentation. Or, 20% de ces profils ont déjà un tag n'apparaissant pas lors de l'inventaire applicatif associé. Ces profils ajoutent donc quelques cas supplémentaires non pris en compte lors du calcul de probabilité précédent.

Tableau IV-VI : Résultats de simulation simples

	5 taux de lecture diminués de 40 points	20 taux de lecture diminués de 10 points
ATTV	35%	11%
RETR	4%	5%
Profil	63%	92%

Comme précédemment, étudions aussi la complémentarité de ces méthodes. On peut ainsi voir à nouveau ici qu'en utilisant conjointement les méthodes de test en ligne, il est possible d'améliorer le taux de détection des défaillances. Ainsi, dans le premier cas, alors qu'en utilisant uniquement la méthode profil il n'est possible de détecter que 63% des défaillances, il est possible d'atteindre 82% de détection. Dans le second cas, on passe de 92% de détection avec la méthode profil, et à 94% de détection en utilisant la méthode profil et la méthode ATTV.

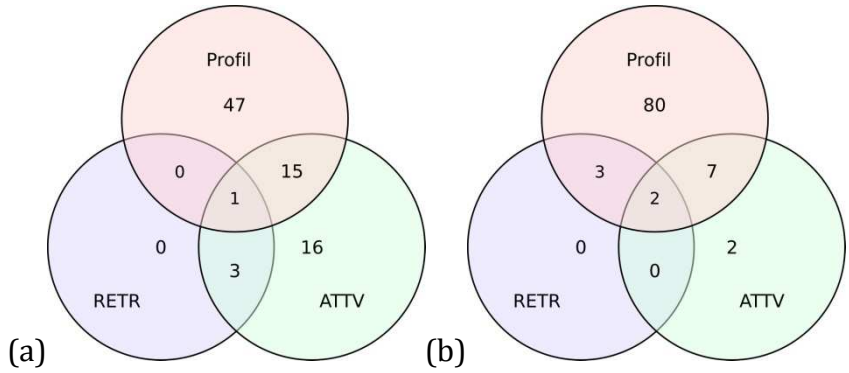


Figure IV-11 : Complémentarité des méthodes de test en ligne lorsque (a) 5 tags voient leur taux de lecture diminuer de 40 points et (b) 20 tags voient leur taux de lecture diminuer de 10 points

Cette simulation nous a aussi permis d'évaluer l'impact des fautes en fonction de leur position dans le profil et de leur importance. Les Figure IV-12 et Figure IV-13 illustrent l'impact des fautes en fonction de leur position dans le profil : la courbe bleue représente le profil limite et la courbe rouge représente le profil fautif. La Figure IV-12 illustre l'impact des

fautes lorsqu'elles se produisent sur les premiers éléments du profil, *i.e.* les tags ayant un taux de lecture élevé. Comme indiqué page 87 présentant le fonctionnement de la méthode profil, ces fautes engendrent un grand nombre de décalages et offrent donc plus de possibilités de dépassement de la limite. Les cercles sur cette figure mettent en avant les zones de dépassement de la limite. Les flèches indiquent le décalage engendré par les fautes : l'extrémité gauche indique l'emplacement du taux de lecture avant l'injection de la faute et l'extrémité droite indique le nouvel emplacement du taux de lecture. La Figure IV-13 illustre l'impact des fautes lorsqu'elles se produisent sur la seconde moitié du profil. On voit ainsi que les décalages des points du profil engendrés par ces fautes sont moins conséquents et donc plus difficiles à détecter.

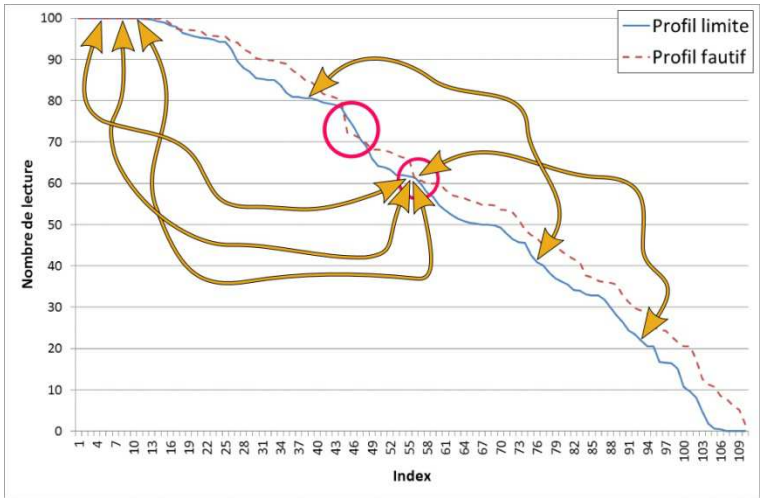


Figure IV-12 : Impacts des fautes lorsqu'elles apparaissent sur les premiers éléments du profil

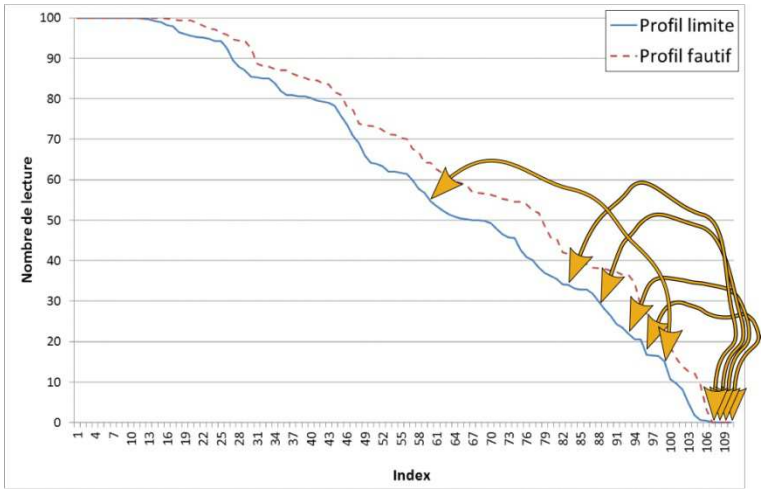


Figure IV-13 : Impacts des fautes lorsqu'elles apparaissent sur les derniers éléments du profil

Cette première simulation nous a permis d'avoir un aperçu des limites de la méthode profil. Néanmoins, les interactions existantes entre les tags ne sont pas prises en compte durant ces simulations. Ces simulations n'ont pas permis non plus d'étudier le comportement des différentes méthodes sur des systèmes RFID différents. De plus, le taux de faux positifs, c'est-à-dire le pourcentage de systèmes déclarés fautifs alors qu'ils sont sains, n'a pas non plus été évalué. Nous réalisons donc des simulations plus complexes à l'aide du simulateur **SERFID** présenté dans le chapitre III.2.

IV.3.2.2 **Simulation complète –** SERFID

Afin de réaliser un grand nombre d'essais sur différents systèmes et en présence de fautes diverses pour évaluer la méthode Profil et la comparer aux méthodes ATTV et RETR, nous utilisons le simulateur **SERFID**, présenté au cours du chapitre III.2. Afin d'automatiser les tests un logiciel spécifique en Java a été développé, nommé « *RFID Simulator Manager* ». La Figure IV-14 montre l'interface permettant de configurer le système, à savoir le nombre de cycles, de slots, de selects et de répétitions des frames pour l'inventaire applicatif, le nombre de tags attendus et le nombre de répétition des inventaires applicatifs. Ainsi, il est possible de répéter facilement une centaine de simulations avec des paramètres identiques. La Figure IV-15 montre l'interface de gestion de l'injection de fautes. On peut ainsi choisir le type de faute injectée : désactivation des canaux ou augmentation du taux d'erreur binaire[1]. Une fois le type de faute choisi, cette interface permet de paramétrer ces injections de fautes :

- le nombre de tags qui seront affectés par la faute ;
- pour l'augmentation du taux d'erreur binaire, un intervalle de valeurs dans lequel sera choisie aléatoirement l'augmentation pratiquée sur un tag ;
- pour la désactivation des canaux de communications, la période et le rapport cyclique caractérisant la désactivation et la réactivation des canaux ;
- le nombre de fois que l'injection de fautes sera répétée avant de passer à l'injection de faute suivante.

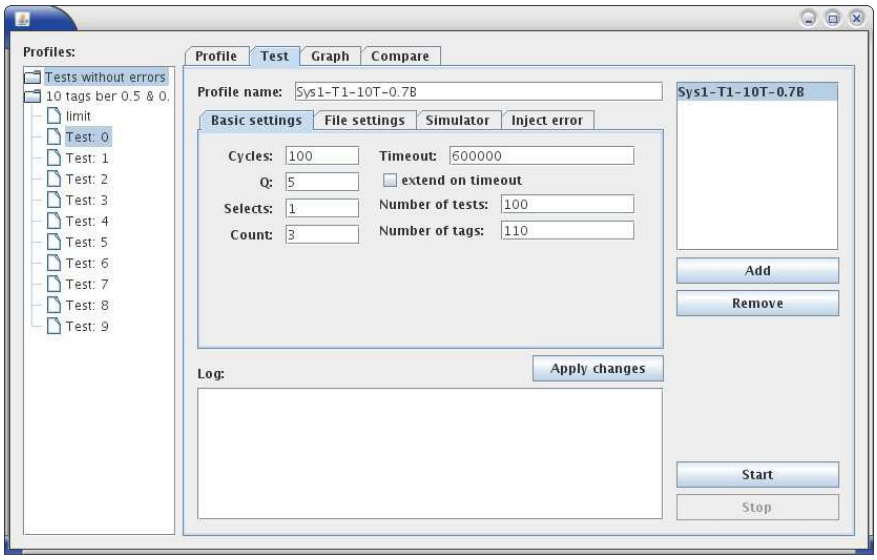


Figure IV-14 : « RFID Simulator Manager » – Interface de gestion des simulations – Onglet « Paramétrage du système »

Ainsi, il est possible d'effectuer un grand nombre de tests automatiquement pour un grand nombre de systèmes. En effet, une fois les tests configurés, les simulations sont lancées : pour chaque test, ce programme réalise autant d'essais que paramétrés. Pour chaque essai, il choisit aléatoirement les tags qui sont affectés par la faute et répète cette injection autant de fois que spécifié, avant de passer à l'essai suivant. On peut ainsi évaluer le résultat de simulation pour le même système et exactement les mêmes fautes – même tags, même impact sur le taux d'erreur binaire – et ainsi conclure sur la reproductibilité des effets des fautes sur le système.

La Figure IV-16 montre l'interface qui permet de réaliser la comparaison entre deux profils, comme par exemple un profil limite et un profil fautif. Ainsi, il est possible d'avoir un

[1] Les modèles de fautes injectées sont décrits dans la section III.4

premier aperçu des taux de détection avant de généraliser la comparaison sous Excel. Un ensemble de plugins a été développé dans cet outil afin d'ajouter ou modifier les différentes méthodes de test utilisées pour comparer deux profils, permettant alors de valider rapidement leur fonctionnement sur quelques cas simples.

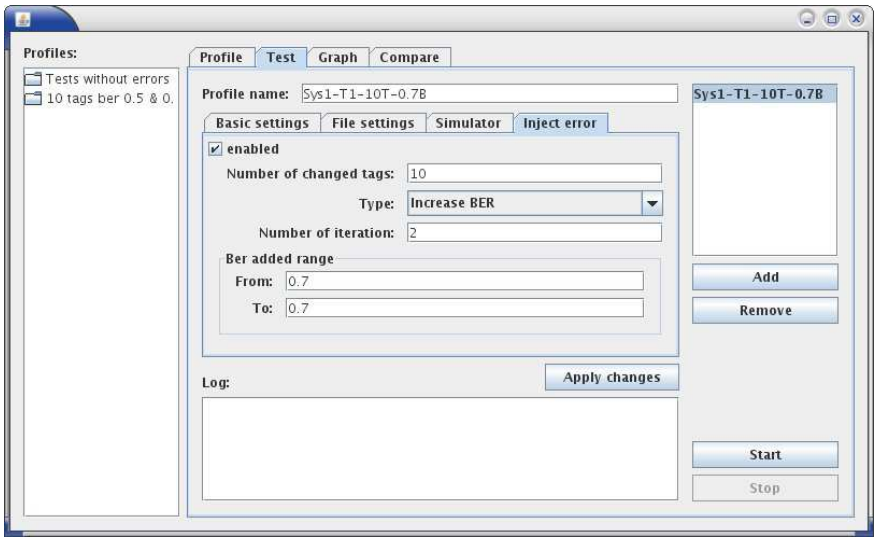


Figure IV-15 : « RFID Simulator Manager » – Interface de gestion des simulations – Onglet « injection de fautes »

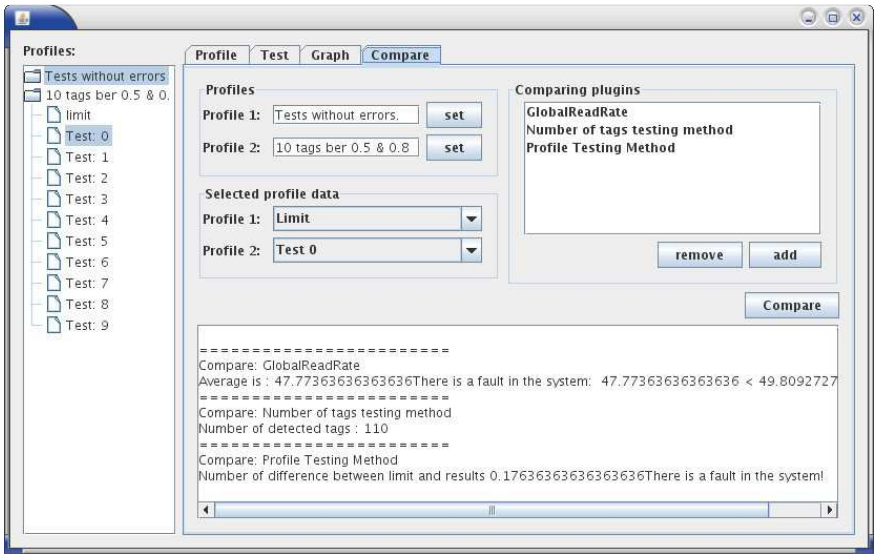


Figure IV-16 : « RFID Simulator Manager » – Interface d'analyse des résultats

IV.3.2.2.1 *Systèmes simulés*

Afin d'évaluer la méthode profil ainsi que ses caractéristiques et ses limites de fonctionnement, nous réalisons différentes simulations. Pour cela, nous définissons différents systèmes RFID à simuler avec différentes caractéristiques. Ces systèmes se basent tous sur le système réel défini précédemment (palette tournante) mais ont des liens tags-lecteurs aux caractéristiques variées. On peut ainsi conforter les premiers résultats obtenus par expérimentations.

En fait, l'efficacité des méthodes de test en ligne repose grandement sur les caractéristiques statistiques du système. Leurs performances sont directement liées en particulier, à la variabilité des taux de lectures du système. Ainsi, en partant du système réel,

nous proposons différentes variantes dans lesquelles les systèmes auront des comportements plus ou moins variables vis-à-vis des taux de lectures.

Les systèmes sont définis par 6 paramètres principaux. Les 4 premiers paramètres concernent la configuration du lecteur. On retrouve ainsi les paramètres de l'inventaire applicatif : le nombre de slots ***N*** caractérisé par le paramètre ***Q***[1], le nombre de ***Cycles*** d'inventaires, le nombre ***Select*** de répétitions de la phase de sélection durant un inventaire simple, le nombre ***Count*** de répétitions de la phase de parcours des frames[2]. Les systèmes simulés sont aussi définis par le nombre de tags ***L*** présents dans le champ du lecteur. Le dernier paramètre est le taux d'erreur binaire ***TEB*** de la communication pour chaque couple lecteur/tag.

Les différents systèmes ont des taux de lectures moyens similaires au système réel mais avec différentes variabilités: faible, moyenne et forte. Le profil de ces systèmes est donc similaire au profil expérimental. Ainsi, on retrouvera entre 10 et 15 tags avec des taux de lectures proches de 100%, 5 à 15 tags avec des taux de lectures aux alentours de 10 à 20%, et les autres taux de lectures des tags seront répartis uniformément. De plus, comme les tags sont sur une palette tournante, le ***TEB*** dépend de la position du tag et de la position de départ de la palette : en effet, les obstacles entre un tag et les antennes du lecteur varient au cours de la rotation. Une composante sinusoïdale est ajoutée au TEB afin de prendre en compte ce phénomène.

Les équations (IV.9) décrivent les ***TEB*** de chaque couple de lecteur-tag d'indice ***i*** compris entre 0 et 109 du premier système. Les constantes présentes dans les équations ont été déterminées à partir du nombre de bits échangés au cours d'une communication et affinées empiriquement à partir des simulations : leurs valeurs ont été définies de manière à respecter les contraintes du système décrites précédemment. Le premier système est défini de sorte que les variations des taux de lecture des tags soient dues à la rotation de la palette, en prenant en compte les spécificités décrites précédemment :

$$\begin{cases} \forall i < 10, TEB_i = \left[1.10^{-7} + 2.\sin\left(\frac{2\pi}{13t} + \frac{2\pi}{110i}\right)\right].10^{-4} \\ \forall i \in [10;105[, TEB_i = \left[-9{,}6 + 2.\sin\left(\frac{2\pi}{13t} + \frac{2\pi}{100i}\right) + 1{,}05i\right].10^{-4} \\ \forall i \geq 105, TEB_i = \left[1 + 2.\sin\left(\frac{2\pi}{13t} + \frac{2\pi}{100i}\right).10^{-2}\right].10^{-2} \end{cases} \quad \text{(IV.9)}$$

Le second système est défini de manière à ce que les tags de ce système aient des taux de lectures fortement variables. La composante sinusoïdale est remplacée par un nombre aléatoire suivant une loi gaussienne. En effet, la composante sinusoïdale précédente fait varier le TEB de 0,04 alors que la composante gaussienne introduite pour ce système fait varier le système de 0,09. L'équation définissant les TEB de chaque tag i de ce système est :

$$\begin{cases} \forall i < 10, TEB_i = [1.10^{-7} + 2.\mathrm{g}(0;0{,}03)].10^{-4} \\ \forall i \in [10;105[, TEB_i = [-9{,}6 + 2.\mathrm{g}(0;0{,}03) + 1{,}05i].10^{-4} \\ \forall i \geq 105, TEB_i = [1 + 0{,}02.\mathrm{g}(0;0{,}03).10^{-2}].10^{-2} \end{cases} \quad \text{(IV.10)}$$

où $\mathrm{g}(\bar{n};\sigma)$ est la fonction donnant un nombre aléatoire suivant une loi gaussienne de moyenne $\bar{n}$ et d'écart type σ.

[1] Pour rappel, le lien entre N et Q est donné par l'équation (II.2) : $N = 2^Q$

[2] L'utilisation de ses paramètres par le lecteur est donnée dans la section 0 à la page 20

Le troisième système possède quant à lui des tags avec des taux de lectures très faiblement variant. Ainsi, en repartant des équations (IV.9) et (IV.10), et en en supprimant les composantes aléatoires, l'équation définissant les TEB de ce système est :

$$\begin{cases} \forall i < 10, TEB_i = 1.10^{-9} \\ \forall i \in [10; 105[, TEB_i = [-9{,}6 + 1{,}05i] . 10^{-4} \\ \forall i \geq 105, TEB_i = 1. 10^{-2} \end{cases} \quad \text{(IV.11)}$$

Enfin, le dernier système va permettre d'étudier l'impact de variabilités hétérogènes des taux de lectures sur les tags du système. En effet, il est apparu au cours des expérimentations que la variabilité des éléments du profil n'est pas la même en fonction de l'élément considéré : les variabilités sont plutôt faibles concernant les premiers et derniers taux de lectures (certains tags sont toujours lus et d'autres ne sont jamais lus), alors que les variabilités des taux de lectures centraux sont moyennes. Afin de prendre en compte ce phénomène, le système #4 a les 15 plus faibles et 15 plus forts ***TEB*** peu variables et les autres ***TEB*** moyennement variables :

$$\begin{cases} \forall i \leq 15, TEB_i = 1.10^{-9} \\ \forall i \in]15; 95], TEB_i = [-1{,}851 + g(0; 0{,}01). 10^{-1} + 1{,}2345i. 10^{-1}] . 10^{-2} \\ \forall i > 95, TEB_i = 1. 10^{-2} \end{cases} \quad \text{(IV.12)}$$

où $g(\bar{n}; \sigma)$ est la fonction donnant un nombre aléatoire suivant une loi gaussienne de moyenne $\bar{n}$ et d'écart type σ.

Le Tableau IV-VII résume les paramètres décrits ci-dessus. Il fait aussi le lien avec les taux de lecture des tags obtenus par simulation de la Figure IV-17 décrite ci-dessous.

Tableau IV-VII : Paramètres des systèmes simulés

Système	Q	Cycles	Select	Count	L	TEB	Taux de lecture
#1 Variabilité moyenne	5	100	1	3	110	éq. (IV.9)	Figure IV-17 (a)
#2 Variabilité forte	5	100	1	3	110	éq. (IV.10)	Figure IV-17 (b)
#3 Variabilité faible	5	100	1	3	110	éq. (IV.11)	Figure IV-17 (c)
#4 Variabilité faible aux extrémités Variabilité moyenne ailleurs	5	100	1	3	110	éq. (IV.12)	Figure IV-17 (d)

Afin d'étudier le comportement statistique de chaque système, et de pouvoir établir les limites des différents tests, au moins 100 inventaires applicatifs ont été simulés pour chaque système. Comme chaque inventaire applicatif est constitué de 100 inventaires simples, ou tentatives de lectures, chaque taux de lectures est calculé à partir de 10 000 tentatives de lectures. En effet, le taux de lectures $\boldsymbol{TL_i}$ du tag $\boldsymbol{i}$ présent dans le champ du lecteur lors d'un inventaire applicatif composé de $\boldsymbol{N_1}$ inventaires simples est défini par :

$$TL_i = \frac{\sum_{j=1}^{N_1} s_{ij}}{N_1} \quad \text{(IV.13)}$$

Où s_{ij} est égal à 1 si une lecture du tag ***i*** s'est produite lors du ***j***-ème inventaire simple, et à 0 sinon. Le taux de lecture moyen TL_{i_moy} du tag ***i*** présent dans le champ du lecteur durant tous les N_2 inventaires applicatifs est défini par :

$$\begin{aligned} TL_{i\_moy} &= \frac{\sum_{k=1}^{N_2} TL_{ik}}{N_2} \\ &= \frac{\sum_{k=1}^{N_2} \frac{\sum_{j=1}^{N_1} s_{ijk}}{N_1}}{N_2} \\ &= \frac{\sum_{k=1}^{N_2} \sum_{j=1}^{N_1} s_{ijk}}{N_1.N_2} \end{aligned} \qquad \text{(IV.14)}$$

où TL_{ik} est le taux de lectures du tag ***i*** lors du ***k***-ième inventaire applicatif et s_{ijk} est égal à 1 si une lecture du tag ***i*** s'est produite lors du ***j***-ème inventaire simple du ***k***-ième inventaire applicatif, et à 0 sinon. Ainsi, les taux de lectures considérés pour étudier le comportement statistique des systèmes et établir les limites ont été déterminés à partir d'un grand nombre de tirages, garantissant donc de se rapprocher du comportement statistique réel des systèmes, d'après la loi des grand nombres[1].

La Figure IV-17 illustre les variations des taux de lectures des tags obtenues par simulation sur les 4 systèmes : les courbes représentent les moyennes des taux de lectures de chaque tag et les traits autour de la courbe les écarts types. On peut ainsi voir sur la partie (a) que le premier système a des variations moyennes. La partie (b) représente les taux de lectures du deuxième système. Ces taux de lectures ont des moyennes plus basses et un écart type plus grand que le système 1. La partie (c) de la Figure IV-17 donne la moyenne et l'écart type des taux de lectures de chaque tag pour le système 3, celui-ci est peu variant. Finalement, la partie (d) de cette figure donne la moyenne et l'écart type des taux de lectures pour le dernier système. On observe sur cette dernière partie deux cassures de la moyenne : une entre l'index 15 et 16 et une autre entre l'index 95 et 96. Ces cassures sont dues aux changements de variabilité tels que décrits précédemment.

Avant de calculer les profils limite pour ces systèmes, nous étudions en premier leur comportement statistique. La Figure IV-18 montre les droites de Henry pour l'index 24 des profils de ces systèmes. Les droites de Henry de tous les points des profils des systèmes sont assimilables à des droites : il est donc possible de conclure que chacun des points du profil a une distribution qui peut s'apparenter à une loi normale. Ainsi, comme les points des profils suivent des lois normales, il est possible d'utiliser l'équation (IV.5) :

$$profil_{limite} = \{l_i \,/\, l_i = \bar{n}_i - 3.\sigma_i\} \qquad \text{(IV.15)}$$

où l_i est la i-ième valeur limite, $\bar{n}_i$ est la i-ème moyenne du porfil moyen et σ_i est l'écart type des i-ièmes nombres de lectures de tous les profils. Avec les équations (IV.1) et (IV.2), il est possible aussi de calculer directement les limites pour les méthodes RETR et ATTV.

Les limites des méthodes profil, ATTV et RETR sont données dans le Tableau IV-VIII. On observe sur le profil limite du système #4 une cassure au niveau de l'index 16. Cette cassure est due au changement de variabilité : en effet, les 15 premiers tags sont très peu variables, leur taux de lectures étant à 100% lors de quasiment tous les inventaires applicatifs, alors qu'à partir du 16e tag, la variabilité devient forte. Les taux de lectures de ce 16e index sont compris entre 84 et 97, avec une moyenne de 90,5 et l'écart type passe donc d'un chiffre proche de 0 à 2,5. La limite est alors pour ce 16e index de $l_{16} = \bar{n}_{16} - 3.\sigma_{16} = 83$ alors

[1] *« la loi des grands nombres exprime le fait que les caractéristiques d'un échantillon aléatoire se rapprochent d'autant plus des caractéristiques statistiques de la population que la taille de l'échantillon augmente. »*[http://fr.wikipedia.org/wiki/Loi_des_grands_nombres]

qu'elle était de 100 pour l'index 15, en raison de la faible variabilité des taux de lectures de cet index.

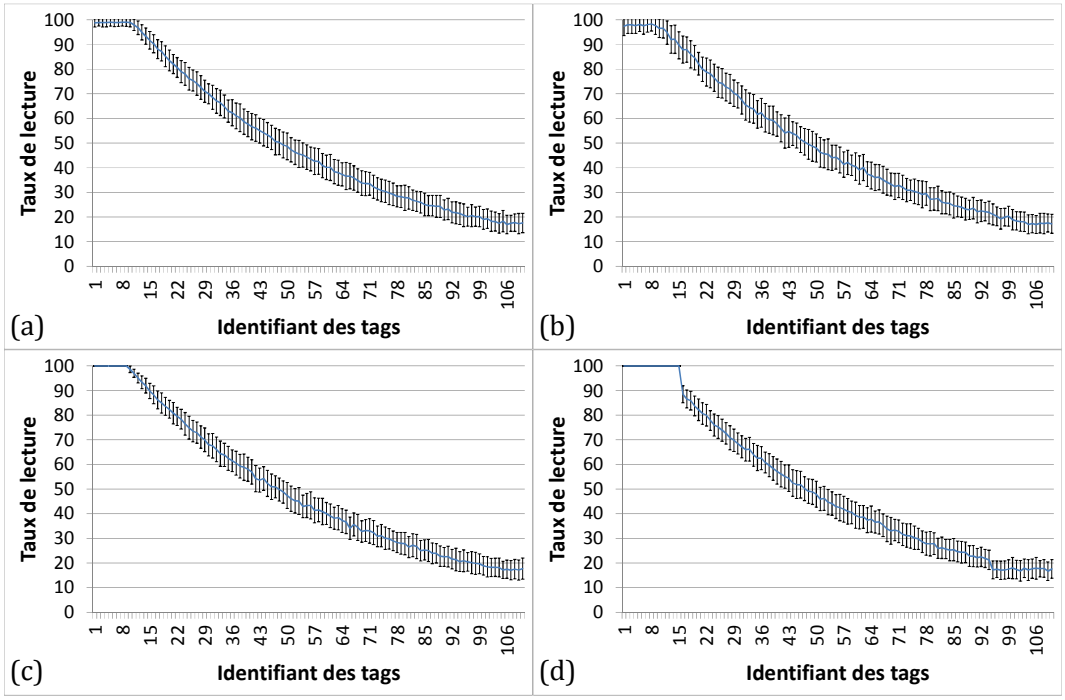


Figure IV-17 : Taux de lecture des tags pour (a) le système #1, (b) le système #2, (c) le système #3 et (d) le système #4

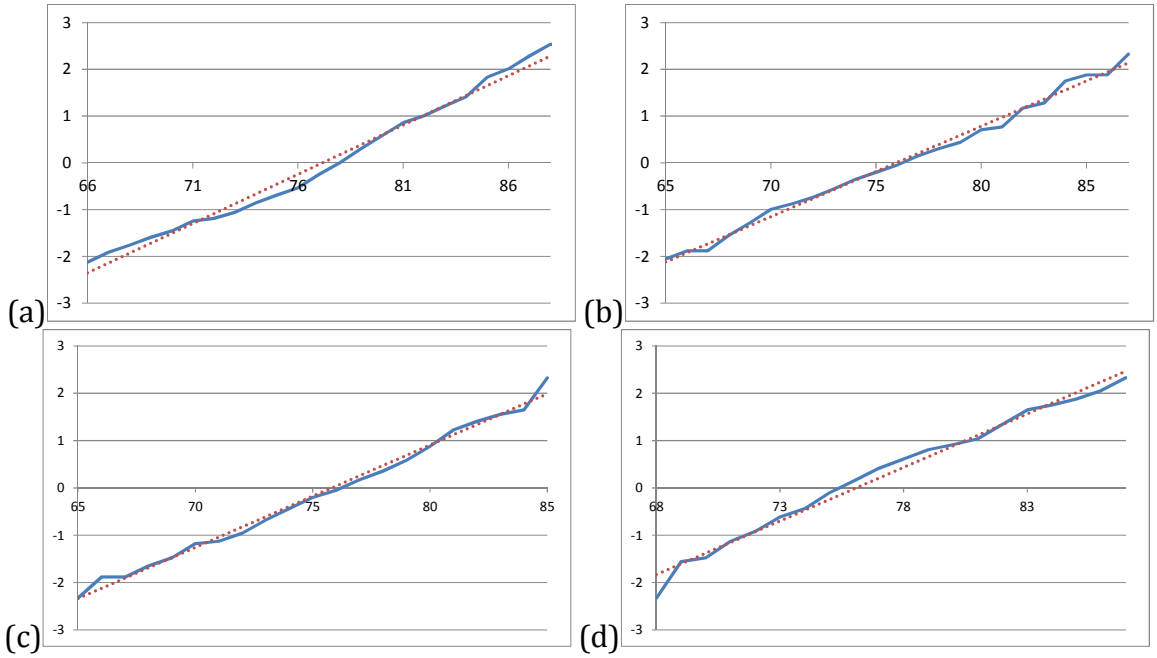
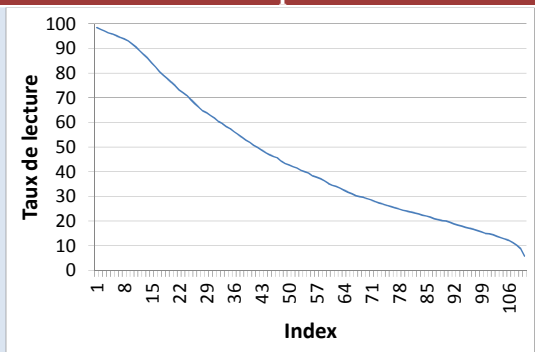
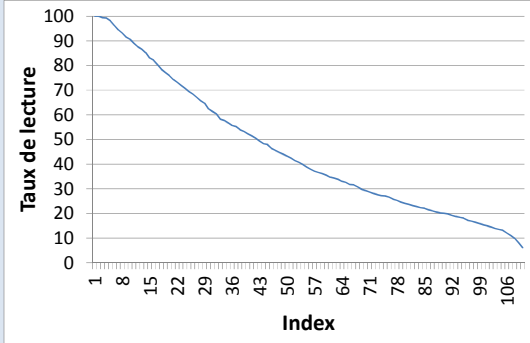
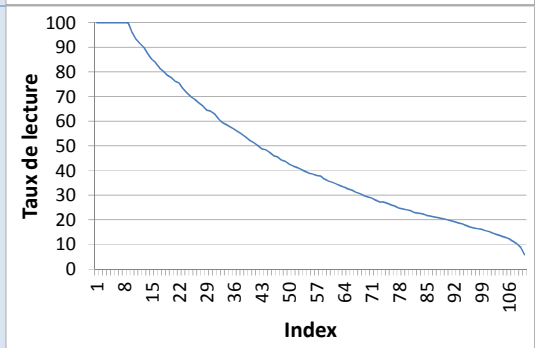
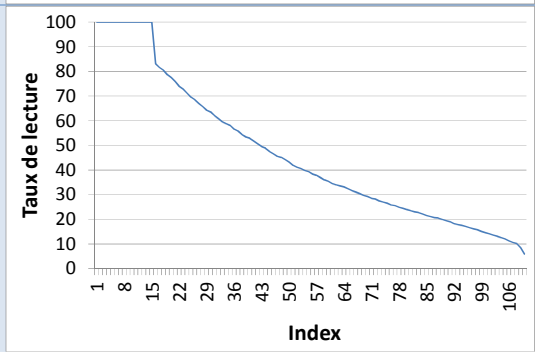


Figure IV-18 : Droite de Henry de l'index 24 des profils (a) du système #1, (b) du système #2, (c) du système #3 et (d) du système #4

On observe aussi dans ce tableau que la limite pour la méthode ATTV est égale à 110. Cela signifie que cette méthode détecte une erreur lorsque le nombre de tags détectés au cours d'un inventaire applicatif est inférieur à 110, c'est-à-dire lorsqu'il manque au moins un tag. En effet, tel que nous avons défini les systèmes simulés, tous les tags sont inventoriés au moins 10 fois au cours d'un inventaire applicatif. Il n'est donc pas possible d'avoir un tag

manquant au cours de ces inventaires : le nombre de tags détectés par la totalité des inventaires applicatifs est donc égal à 110. Le volume moyen de trafic de tag est donc égal à 110, avec un écart type nul, lorsque l'unité de temps utilisé est l'inventaire applicatif.

Tableau IV-VIII : Limites des méthodes profil, ATTV et RETR pour les systèmes simulés

Système	Limite ATTV	Limite RETR	Limite profil
#1	110	50,93	
#2	110	51,61	
#3	110	51,12	
#4	110	50,97	

Le comportement des différents systèmes sains vient d'être étudié. A partir de cette étude, des limites pour trois méthodes de test en ligne ont été fixées. Nous utilisons maintenant le simulateur **SERFID** afin d'étudier les capacités de détection de défaillances de ces méthodes, ainsi que le nombre de faux positifs – c'est-à-dire le nombre de cas où le système est déclaré défaillant par la méthode alors qu'il n'y a pas de défaillances.

Les fautes décrites dans la section III.4.3 ont été injectées dans chacun des systèmes décrits ci-dessus. Ainsi, nous pouvons étudier les performances des méthodes de tests profil, ATTV et RETR en présence de fautes diminuant la qualité du canal de communication. Pour chaque faute et pour chaque système, un minimum de 20 simulations ont été réalisées. Lors de chaque simulation, les tags affectés par la faute ont été choisis aléatoirement dans toute la population des tags présents. Les simulations sont gérées par le logiciel « RFID Simulator Manager » présenté en début de section.

Dans la suite du document, nous décrivons la méthode d'évaluation des tests en détaillant les résultats de simulation seulement pour le quatrième système et en présence seulement de la faute augmentant le taux d'erreur binaire de 3.10^{-3} pour 5% des tags présents dans le champ du lecteur.

Pour évaluer le comportement de ce système et la capacité des méthodes de test à détecter les défaillances, 100 inventaires applicatifs ont été réalisés avec le système fautif. Pour chaque inventaire, 5 tags ont été choisis aléatoirement et les taux d'erreur binaire associés à leurs canaux ont été augmentés de 3.10^{-3}. Le Tableau IV-IX donne quelques exemples de choix de tags et les modifications des taux d'erreur binaire associés.

Tableau IV-IX : Tags affectés lors des injections de fautes

Inventaire applicatif	Tags affectés		
	Identifiant	Ancien TEB	Nouveau TEB
1	11	$1{,}0.10^{-9}$	$3{,}0.10^{-3}$
	18	$9{,}0.10^{-4}$	$3{,}9.10^{-3}$
	84	$7{,}8.10^{-3}$	$1{,}1.10^{-2}$
	102	$1{,}0.10^{-2}$	$1{,}3.10^{-2}$
	103	$1{,}0.10^{-2}$	$1{,}3.10^{-2}$
⋮			
8	12	$1{,}0.10^{-9}$	$3{,}0.10^{-3}$
	17	$8{,}0.10^{-4}$	$3{,}8.10^{-3}$
	22	$1{,}4.10^{-3}$	$4{,}4.10^{-3}$
	31	$2{,}2.10^{-3}$	$5{,}2.10^{-3}$
	83	$7{,}9.10^{-3}$	$1{,}1.10^{-3}$
⋮			
27	21	$1{,}3.10^{-3}$	$4{,}3.10^{-3}$
	45	$3{,}8.10^{-3}$	$6{,}8.10^{-3}$
	69	$6{,}3.10^{-3}$	$9{,}3.10^{-3}$
	70	$6{,}4.10^{-3}$	$9{,}4.10^{-3}$
	109	$1{,}0.10^{-2}$	$1{,}3.10^{-2}$
⋮			
31	47	$3{,}9.10^{-3}$	$6{,}9.10^{-3}$
	63	$5{,}7.10^{-3}$	$8{,}7.10^{-3}$
	95	$1{,}0.10^{-2}$	$1{,}3.10^{-2}$
	100	$1{,}0.10^{-2}$	$1{,}3.10^{-2}$
	107	$1{,}0.10^{-2}$	$1{,}3.10^{-2}$
⋮			

Pour chaque inventaire applicatif du système fautif, les trois méthodes de test en ligne ont été appliquées. Le nombre de tags identifiés, le nombre d'erreurs de lectures sur le nombre total de tentatives de lectures et le profil des inventaires applicatifs ont ainsi été calculés. Le Tableau IV-X donne quelques exemples de résultats de simulations. On y retrouve le numéro de l'inventaire applicatif, que l'on peut alors relier au Tableau IV-IX explicitant les fautes

affectant le système. Pour chaque inventaire applicatif, on retrouve le nombre de tags détectés, nombre comparé à la limite de la méthode ATTV. On y retrouve aussi le rapport entre le nombre d'erreurs de lecture et le nombre de tentatives de lecture, rapport comparé à la limite de la méthode RETR. On y retrouve aussi le nombre de points du profil de l'inventaire applicatif passant en dessous du profil limite. Les cases vertes indiquent que la méthode correspondante a détecté le système comme défaillant : ainsi, le premier inventaire applicatif est considéré fautif par la méthode profil uniquement ; le huitième inventaire applicatif est déclaré défaillant par la méthode RETR et la méthode profil ; le vingt-septième inventaire applicatif est vu comme ayant une défaillance uniquement par la méthode RETR ; et finalement, le trente-et-unième inventaire applicatif est considéré sain par toutes les méthodes.

Tableau IV-X : Résultats des méthodes de tests face aux inventaires applicatifs fautifs

Inventaire applicatif	Méthode ATTV	Méthode RETR	Méthode profil
1	110	50,72	2
…			
8	110	51,56	6
…			
27	110	51,36	0
…			
31	110	49,87	0

Ces simulations permettent de valider les observations réalisées lors des premières simulations– section IV.3.2.1. En effet, on observe que les fautes impactant les tags les plus forts – *i.e.* les tags dont les taux de lectures sont élevés – sont plus facilement détectables que les fautes se produisant au niveau des tags avec des taux de lectures plus faibles. Comme indiqué précédemment, ceci est dû au fait qu'il se produit plus de décalages dans le profil lorsque les fautes se produisent dans la première moitié de celui-ci que lorsqu'elles se manifestent dans la seconde moitié. Ces différences de décalages sont amplifiées par l'impact des fautes sur les tags « faibles ou forts ». En effet, l'augmentation du nombre d'erreurs binaires dans la communication a des effets différents pour les tags forts et les tags faibles : les communications des tags forts présentent très peu d'erreurs au niveau bit et l'augmentation du nombre de ces erreurs a un fort impact – le taux de lectures d'un tag fort passe de 100% à 60% après l'injection de la faute – alors que les communications des tags faibles ont déjà beaucoup d'erreurs au niveau bit et l'augmentation du nombre d'erreurs a un faible impact – le taux de lecture d'un tag faible qui est de 16% en moyenne passe à 8% après injection de faute. Dans le premier cas, le taux d'erreurs de lecture passe de premier à 36^{e} élément du profil, engendrant alors 36 décalages et donc 36 potentiels dépassements de profil, alors que dans le second cas, le taux de lecture passe de 104^{e} à 110^{e} élément du profil, ne générant alors que 6 décalages.

Le Tableau IV-XI donne les taux de détections des différentes méthodes et de leurs combinaisons pour les différentes simulations réalisées : sans fautes, puis avec chacun des 6 modèles de fautes. Le symbole **&&** correspond à l'intersection de deux ou plusieurs méthodes et le symbole **||** correspond à l'union de deux ou plusieurs méthodes. Ainsi, la ligne « Profil **&&** RETR » contient le nombre de défaillances détectées à la fois par la méthode profil et la méthode RETR rapporté sur le nombre total de défaillances ; et la ligne « Profil **||** RETR » contient le nombre de défaillances détectées soit par la méthode profil, soit par la méthode RETR, soit par les deux, rapporté sur le nombre total de défaillances. Les différentes colonnes correspondent aux simulations réalisées : T0 correspond à la simulation sans injection de fautes, et les T1 à T6 correspondent aux simulations avec injection de fautes telles que décrites dans la section III.4.3. On peut ainsi voir que, pour la simulation avec l'injection de la faute T5 – 5 tags dont les taux d'erreurs binaires sont augmentés de 0,3 points –, la méthode

profil détecte seule 73% des défauts alors que la méthode RETR en détecte 36%. 32% des fautes sont détectées à la fois par la méthode profil et par la méthode RETR : il y a donc 36% des fautes qui sont détectées soit par la méthode profil, soit par la méthode RETR soit par les deux. Ici, la méthode ATTV ne détecte aucune défaillance. Il est alors possible de conclure que la méthode profil détecte plus de défaillances que les autres méthodes. Néanmoins, afin d'obtenir un taux de détection optimal, il est préférable d'utiliser conjointement plusieurs méthodes de test en ligne. Il est aussi à noter que plus la faute est importante, c'est-à-dire que plus il y a de tags touchés et que la communication est fortement perturbée, plus il est évidemment facile de détecter la défaillance.

Tableau IV-XI : Système #4 – Taux de détections des défaillances par les méthodes de test en ligne et par combinaison de ces méthodes

méthode	T0 – Sans faute	T1 – 10 Tags 5.10^{-3} – $7.10^{-3}\uparrow$	T2 – 10 Tags 2.10^{-3} – $5.10^{-3}\uparrow$	T3 – 10 Tags $2.10^{-3}\uparrow$	T4 – 5 Tags 4.10^{-3} – $6.10^{-3}\uparrow$	T5 – 5 Tags $3.10^{-3}\uparrow$	T6 – 2 Tags $5.10^{-3}\uparrow$
Profil	9%	100%	99%	90%	95%	73%	67%
ATTV	0%	0%	0%	0%	0%	0%	0%
RETR	0%	100%	98%	65%	72%	36%	7%
Profil && ATTV	0%	0%	0%	0%	0%	0%	0%
Profil && RETR	0%	100%	92%	63%	70%	32%	7%
Profil && RETR && ATTV	0%	0%	0%	0%	0%	0%	0%
Profil || ATTV	9%	100%	99%	90%	95%	73%	67%
Profil || RETR	9%	100%	99%	92%	97%	77%	67%
ATTV || RETR	0%	100%	98%	65%	72%	36%	7%
Profil || ATTV || RETR	9%	100%	99%	92%	97%	77%	67%

La Figure IV-19 reprend les données du tableau pour illustrer la complémentarité existante entre ces 3 méthodes sur le système #4.

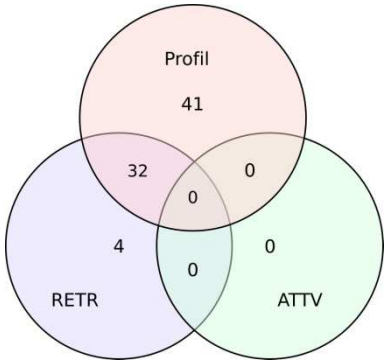


Figure IV-19 : Diagramme de Venn montrant la complémentarité des méthodes de test en ligne profil, ATTV et RETR sur le système #4

La Figure IV-20 donne les diagrammes de Venn, pour les quatre systèmes, des méthodes de test en ligne profil, ATTV et RETR. Elle représente les différents taux de détections moyens toutes défaillances confondues. On voit ainsi que, quelle que soit la variabilité du système, la méthode profil détecte toujours plus de défaillances que les autres méthodes. Mais il est toujours plus intéressant, pour détecter un maximum de défaillances, d'utiliser conjointement plusieurs méthodes.

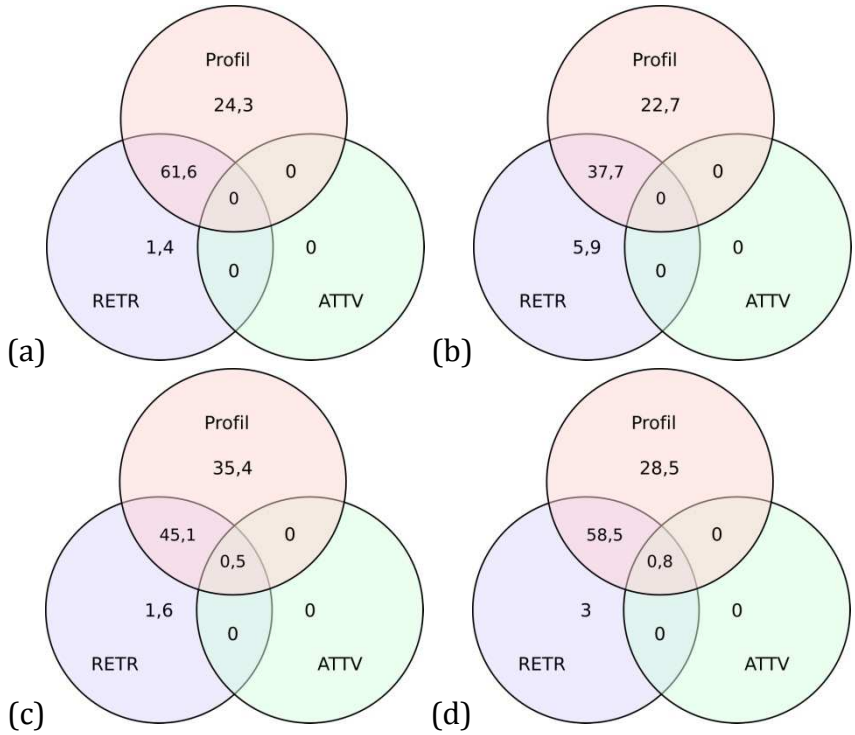


Figure IV-20 : Diagrammes de Venn montrant la complémentarité des méthodes de test en ligne profil, ATTV et RETR pour le (a) système #1 moyennement variant, (b) le système #2 peu variant, (c) le système #3 fortement variant et (d) le système #4 faiblement variant pour les 15 tags les plus forts et les plus faibles et moyennement variant pour les autres tags

Dans le Tableau IV-XI, on remarque aussi que le taux de fausses détections est assez élevé : 9% des systèmes sains sont détectés comme défaillants alors qu'ils fonctionnent correctement. Ce taux de fausses détections dépend de la variabilité du système. En effet, pour un système peu variable, le taux de faux positifs augmente encore en s'élevant à 12%. Ceci est dû au fait que, dans ce système, les différents profils sont fortement semblables, les taux de lectures les composants variant peu le profil limite est alors proche des profils sains. Ainsi, une faible variation sera plus facilement susceptible de passer en dessous de la limite alors que dans un système fortement variant, dans lequel le profil limite est plus éloigné des profils sains, les variations doivent être plus conséquentes pour passer en dessous du profil limite. En effet, dans un système peu variant, les éléments de la limite sont souvent à un ou deux points des éléments correspondants de la moyenne et quelques éléments sont même à moins d'un point de leur moyenne. Ainsi, une variation de seulement un ou deux points du taux de lecture dans le cas d'un système peu variant amène des points en dessous de la limite. La perte d'un point du taux de lecture correspond à un inventaire simple sur cent au cours d'un inventaire applicatif n'ayant pas identifié le tag ayant entraîné le dépassement de la limite. Dans le cas d'un système fortement variant, la limite est en général cinq à dix points en dessous de la moyenne, voire même quinze ou vingt points pour quelques éléments. Il faut donc beaucoup plus d'erreurs de lecture avant qu'un inventaire applicatif ne soit déclaré comme fautif.

Afin de diminuer ce taux important de faux positifs, une solution intrusive, contrairement aux autres méthodes décrites, est d'effectuer un second inventaire applicatif. Si ce second inventaire applicatif est toujours déclaré fautif, alors le système est dit fautif, sinon il est déclaré sain. Avec cette méthode, le taux de faux positif devient alors nul et le taux de détections reste inchangé. En effet, les variations des profils sains détectés comme fautifs sont tellement faibles que lors du second inventaire applicatif, moins de 1% des inventaires sont déclarés fautif alors qu'ils ne le sont pas. Le taux de détections reste quasiment inchangé car les variations induites par les fautes sont très importantes et restent dans la plupart des cas présentes lors du second inventaire, engendrant alors la détection du système comme fautif.

IV.4 Conclusion

Au cours de ce chapitre, nous avons étudié, à l'aide du simulateur SERFID présenté dans le chapitre précédent, trois méthodes de test en ligne. Ces méthodes, non intrusives, ont pour objectif la détection de défaillances dans les systèmes RFID. Ces défaillances ont comme manifestation principale la dégradation des communications entre les tags et les lecteurs. Les méthodes de test en ligne considérées sont donc basées sur l'étude des taux de lectures et la capacité du système à détecter la présence d'un tag.

Les deux premières méthodes présentées sont dérivées de méthodes classiques consistant à compter le nombre moyen de tags ou le nombre d'erreurs de lecture rapporté sur le nombre de tentatives de lecture par unité de temps. Respectivement nommées ATTV et RETR, ces méthodes ont pour objectif la détection de défaillances survenant principalement au niveau du lecteur ou de l'environnement du système. Ces méthodes, à la base conçues pour observer le système sur un intervalle de temps durant lequel plusieurs palettes de tags sont inventoriées, ont été paramétrées afin que l'unité de temps utilisée coïncide avec le temps de lecture d'une palette, autorisant alors la comparaison avec la méthode proposée, nommée « profil ».

Afin d'évaluer cette méthode de test en ligne non intrusive et de la comparer aux méthodes existantes, des expérimentations ont été réalisées. Ces expérimentations consistent à la mise en œuvre d'un lecteur pour la détection de cartons sur des palettes, chaque carton étant identifié par un tag. Des dysfonctionnements ont été introduits en modifiant l'emplacement et l'orientation des tags ainsi que le fonctionnement de la palette afin de tester les performances de la méthode profil. Puis, au vu de la complexité de mise en œuvre et pour l'étude de cas variés, des simulations ont été réalisées à l'aide du simulateur **SERFID** présenté dans le chapitre précédent. Ce simulateur nous a permis d'observer l'impact de certaines fautes sur le système complet. Il nous a aussi permis de valider les performances des méthodes de tests face à différentes fautes possibles. De plus, le temps de calcul de cette méthode, comme celui des deux autres, est très faible et négligeable par rapport aux temps d'inventaire. Ce temps est uniquement dépendant du nombre de tags présents dans l'inventaire.

Cette méthode a été publiée dans plusieurs conférences [FRI10, FRI11b, FRI12] et un journal [FRI11c].

En conclusion, nous avons pu observer que la méthode profil permet une meilleure détection de défaillances au sein du système – tags, lecteur et environnement – que les méthodes classiques ATTV et RETR. Il est apparu clairement que le caractère homogène du système était clairement indispensable au fonctionnement de cette méthode : en effet, il est nécessaire que les résultats des inventaires applicatifs soient similaires les uns aux autres, d'une palette à une autre. Cela est rendu possible par le fait d'utiliser toujours le même format de palette, contenant toujours les mêmes produits et plaçant les tags systématiquement de la même manière au sein de la palette. De plus, les fautes affectant les tags fonctionnant parfaitement sont facilement détectables, alors que les fautes affectant les tags les plus faibles sont difficiles à détecter. Pour finir, l'utilisation conjointe des différentes méthodes a été évaluée : chacune des méthodes, ayant des caractéristiques particulières, sont capables de

détecter certaines défaillances que les autres n'ont pas pu. Ainsi, en utilisant conjointement les trois méthodes, de bons taux de détection sont possibles. Ces taux de détections sont dépendants de la variabilité du système – *i.e.* de sa capacité à reproduire les mêmes profils d'une palette à une autre.

V Conclusion

V.1 Contribution

Ce document retrace le travail réalisé au cours de la thèse que j'ai effectuée ces dernières années au sein du Laboratoire de Conception et d'Intégration des Systèmes – LCIS. L'objectif de cette thèse a été d'étudier la sûreté de fonctionnement des systèmes RFID, en particulier la détection de fautes pendant le fonctionnement du système.

Dans un premier temps, les systèmes RFID ont été étudiés, afin d'en faire ressortir les caractéristiques principales : la télé-alimentation des tags et l'utilisation d'une communication passive, c'est-à-dire sans émetteur, pour la communication des tags vers le lecteur. Ce mode de fonctionnement particulier engendrant des problèmes de robustesse, nous nous sommes ensuite intéressés aux méthodes existantes permettant d'améliorer la robustesse des systèmes RFID.

Bien qu'il existe de nombreux outils permettant d'évaluer les performances des différents éléments d'un système RFID, ceux étudiés ne répondaient pas aux contraintes de modélisation et aux besoins d'injection de fautes nécessaires à notre travail. Il est donc apparu nécessaire de réaliser un outil permettant l'évaluation globale du système, de ses performances et de son comportement en présence de faute. Afin de déterminer le niveau de modélisation le plus adapté pour réaliser cet outil, nous avons tout d'abord étudié la robustesse des systèmes RFID. Pour cela, nous avons appliqué la méthode AMDE couramment utilisée [FRI10]. Cette méthode s'appuie sur une analyse systématique des modes de défaillances pouvant apparaître et de leurs effets. Nous avons alors étudié l'origine possible de ces modes de défaillances ainsi que les méthodes permettant de détecter l'apparition de ces défaillances, de les recouvrir ou de les éviter. Cette analyse a mené aux choix du niveau de modélisation transactionnel avec gestion du temps distribué (TLM-DT) et de modèles de fautes pouvant apparaître dans ces systèmes [FRI11a]. Le modèle proposé a été implanté dans un simulateur développé en **SYSTEMC** – représentant au total un peu plus de 20 000 lignes de code – nommé **SERFID**. Ce simulateur est capable de simuler un système possédant plusieurs tags et plusieurs lecteurs UHF respectant la norme EPC Classe 1 Génération 2 [EPCC1G2] et d'interfacer les lecteurs à un logiciel tiers, tel qu'un middleware. Il est possible d'injecter des fautes dans le modèle soit lors de la configuration de la simulation soit dynamiquement au cours de son exécution. Afin de s'assurer du bon fonctionnement de ce simulateur, des tests unitaires et globaux ont été appliqués afin de valider le respect de la norme aussi bien du point du vue des fonctionnalités implantées que des temps de communication et de vérifier les propriétés aléatoires du système – gestion et choix des slots par les tags. Une version permettant la simulation de système HF, respectant la norme ISO15693 [ISO15693-1, ISO15693-2, ISO15693-3] a également été développée. Un outil supplémentaire a été développé afin de remplacer le middleware et d'automatiser les différents tests nécessaires à l'étude du système et des méthodes de test en ligne. Cet outils, appelé « *RFID Simulator Manager* », permet ainsi de faire de la co-simulation.

Finalement, dans une dernière étape, une méthode de test en ligne permettant de détecter la présence de défaillances dans le système RFID est proposée [FRI11b, FRI11c, FRI12]. Cette méthode s'appuie sur l'observation des performances du système. En effet, lors d'un inventaire applicatif, le lecteur identifie les différents tags plus ou moins souvent, ceci étant dû aux caractéristiques intrinsèques du système et de sa sensibilité à l'environnement. Lors d'un inventaire applicatif, chaque tag est alors associé à son taux de lecture, témoin des performances de communication entre ce dernier et le lecteur. Les taux de lecture sont alors récupérés et triés : ils forment ainsi le profil du système. De par nos expérimentations, nous

avons constaté que ces profils varient peu d'une palette à une autre. Après une étude statistique du système, il est alors possible d'établir un profil limite. Ainsi, si tous les points d'un profil sont au-dessus des points de ce profil limite alors le système est considéré comme sain. Si un point ou plus d'un profil sont en dessous des points du profil limite, alors le système est considéré comme fautif. Cette méthode a été évaluée par expérimentation et par simulation, puis comparée à deux méthodes classiques : ATTV, mesurant le flux de tags passant devant un lecteur et RETR, mesurant le taux d'erreur global d'un lecteur. Les expérimentations et les simulations amènent à observer que la méthode proposée ici est plus performante que les méthodes RETR et ATTV. Néanmoins, ces deux dernières méthodes détectent des systèmes fautifs que la méthode profil ne détecte pas. Ainsi, il apparaît, au vu des expérimentations et simulations, qu'il est intéressant d'utiliser conjointement ces trois méthodes afin d'obtenir un taux de détection des systèmes fautifs optimal. Néanmoins, la méthode profil induit un taux de fausses détections de l'ordre de 10%. En cas de détection d'une défaillance, il est alors nécessaire de réaliser un inventaire applicatif supplémentaire afin de valider ou d'invalider le résultat du premier test. Les taux de fausses détections sont alors de l'ordre de moins 1% et le taux de détection reste inchangé. Afin de diminuer encore le taux de fausses détections, il est possible d'effectuer un troisième inventaire.

V.2 Perspectives

La problématique abordée dans cette thèse est large. En effet, la sûreté de fonctionnement des systèmes RFID impose l'étude de chaque composant du système, allant de la puce présente dans le tag jusqu'à l'infrastructure informatique permettant de gérer les données et de leur donner un sens vis-à-vis de l'application finale.

Les premières perspectives concernent directement le travail présenté ici. En effet, les phénomènes physiques liés aux échanges de signaux entres les tags et les lecteurs ont été modélisés avec des modèles simples à un haut niveau d'abstraction. Il est envisageable d'introduire des modèles plus détaillés, notamment au niveau du modèle de propagation de l'énergie et de celui de l'injection d'erreur binaire. Cela permettra de mieux prendre en compte les fautes dues à l'environnement, notamment celles liées à la télé-alimentation et aux perturbations électromagnétiques. En effet, il apparaît que les erreurs binaires dans des communications RF sont souvent groupées, ce qui n'est pas modélisé dans le modèle actuel.

Le second axe sera l'amélioration de la méthode profil. En l'état, cette méthode est intrusive : plusieurs inventaires applicatifs sont nécessaires afin d'éviter des taux de fausses détections prohibitifs. Il est possible, en modifiant la définition de la limite permettant de déclarer un système sain ou fautif, de la rendre non intrusive. En effet, la limite est définie suite à une analyse statistique des profils du système. Un profil est un ensemble ordonné de taux de lecture, chaque taux de lecture étant une variable aléatoire. Il est alors possible d'appliquer les théories mathématiques existantes sur les ensembles ordonnées de variables aléatoires [BAL07]. Ces théories mathématiques sont complexes et nécessitent de connaître la loi que suit chaque variable aléatoire, mais elles pourraient permettre de calculer une limite mieux adaptée car elles prennent en compte les spécificités aléatoires de chaque point du profil. Il est aussi apparu au cours des différentes études que les faux positifs de la méthode profil dépassent le profil limite de peu alors que les systèmes fautifs ont un écart important avec ce dernier. Il est alors possible d'effectuer une discrimination à partir du « niveau de dépassement » du profil limite : un système pourrait être déclaré fautif si plusieurs points sont en dessous du profil limite plutôt qu'un seul, ou encore si la somme des dépassements est suffisamment importante. Ainsi les cas de faux positifs pourraient être identifiés sans avoir besoin de réaliser un second inventaire. Néanmoins, l'impact de ces méthodes sur le taux de bonnes détections n'est pas prévisible et demande donc une étude détaillée.

Les autres perspectives concernent le projet **SAFERFID**. En effet, afin d'assurer la sûreté de fonctionnement des systèmes RFID, il est important d'étudier chaque composant du système. Ainsi, une thèse en cours s'intéresse à la robustesse du système RFID au niveau du middleware [KHE12]. Le travail concerne la détection et la localisation des défaillances dans le système grâce à la vue globale qu'en a le middleware. En effet, ce dernier agrège les données de tous les lecteurs. Il est alors capable, à l'aide des méthodes de détection locales proposées dans cette thèse appliqués à plusieurs lecteurs, de localiser la défaillance. Il pourra alors envisager des méthodes de recouvrement ou de reconfiguration. Le simulateur présenté dans ce document, **SERFID**, pourrait alors être utilisé afin de simuler une infrastructure comportant plusieurs lecteurs et évaluer les méthodes de détection et de reconfiguration implantées dans le middleware.

Un autre travail démarré dans ce projet concerne la définition et le développement d'un tag sur une carte FPGA couplée à un front-end RF [ABD12]. L'objectif est de proposer une plateforme permettant de valider l'architecture numérique d'un tag dans un environnement réel. L'avantage d'avoir un tag instrumenté est qu'il devient alors possible de visualiser les paramètres du système ainsi que les états internes du tag lors d'un fonctionnement avec d'autres tags. De plus, l'injection et l'observation de la propagation de fautes deviennent alors possibles dans un environnement réel. Par exemple, il sera possible d'injecter dans le tag instrumenté les modèles de fautes identifiés dans ce travail afin d'observer leur impact de manière expérimentale. Finalement, il sera aussi possible de proposer des solutions au niveau de l'architecture numérique du tag pour accroître sa robustesse.

VI Bibliographie

[ABD12] O. Abdelmalek, D. Hély, V. Beroulle, *Design and emulation of new robust architectures for UHF RFID Tags*, Colloque National du GDR SoC-SiP, 13-15 juin 2012

[AFS09] Agence Française de Sécurité Sanitaire et du Travail (AFSSET), *Les systèmes d'identification par radiofréquences (RFID), Evaluation des impacts sanitaires*, janvier 2009,

`http://www.afsset.fr/upload/bibliotheque/726108694775617668756800952202/RFID-Afsset_janvier_2009.pdf`

[AHM07] N. Ahmed, R. Kumar, R. S. French and U. Ramachandaran, *RF²ID: A reliable middleware framework for RFID deployment*, in proc. IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2007

[ALIEN] *Alien Interface Guide for ALR-9900, ALR-9800, ALR-8800 and ALR-9650– All fixed readers*, Alien Technology, July 2009

[ANG07] C. Angerer, B.Knerr, M.Holzer, A.Adalan, M. Rupp, *Flexible simulation and prototyping for RFID designs*, First international EURASIP Workshop on RFID Technology, pp.51-54, Sept. 2007

[ANG09] C. Angerer, R.Langwieser, *Flexible evaluation of RFID system parameters using rapid prototyping*, 2009 IEEE International Conference on RFID, pp. 42-47, 27-28 April 2009

[ARL06] J. Arlat, Y. Crouzet, Y. Deswarte, J.-C. Fabre, J.-C. Laprie, D. Powell, *Tolérance aux fautes*, dans « Encyclopédie de l'informatique et des systèmes d'information », (J. Akoka and I. Comyn-Wattiau, Eds.), Partie 1 : La dimension technologique des systèmes d'information – Section 2 : L'architecture et les systèmes (M. Banâtre, Ed.), ISBN 27117

[BAL07] N. Balakrishnan, *Permanents, Order Statistics, Outliers, and Robustness*. Revista Matemática Complutense, Amérique du Nord, 20, mar. 2007

[BOL07] L. Bolotnyy, G. Robins, *The case for multi-tag RFID systems*, International conference on Wireless Algorithms, Systems and Applications, 2007, pp.174-186

[CAU04] L. Cauffriez, J. Ciccotelli, B. Conrard, M. Bayart, *Design of intelligent distributed control systems: a dependability point of view*, Reliability Engineering and System Safety 84 (2004), pp.19-32

[CAV08] A. Cavalli, E. Martins, A. Morais, B. C. Moreira, *Une approche de test de robustesse basée sur l'injection de fautes et le monitorage*, Colloque Francophone sur l'ingénierie des protocoles (CFIP), 1-11 Feb. 2008, pp.1-13

[CHE07] T. Cheng, L. Jin, *Analysis and Simulation of RFID Anti-collision Algorithms*, The 9th International Conference on Advanced Communication Technology, February 12-14, 2007, volume 1, pp697-701

[COS00] C. da Costa, *Hardware for production test of RFID interface embedded into chips for smart cards and labels used in contactless applications*, IEEE International TEST Conference (ITC'00), 2000, pp.485-491

[DAS09] R. Das, P. Harrop, *RFID Forecasts, Players and Opportunities 2009-2019*, IdTechEx report, 2009

[DER07a] R. Derakhshan, M.E. Orlowska and X. Li, *RFID Data Management: Challenges and Opportunities*, 2007 IEEE International Conference on RFID, USA, 2007

[DER07b] V. Derbek, C. Steger, R. Weiss, D. Wischounig, J.Preishuber-Pfluegl, M.Pistauer, *Simulation Platform for UHF RFID*, Design, Automation & Test in Europe Conference & Exhibition, 2007, DATE'07, pp.1-6, 16-20 April 2007

[DON10] G. Dongjiu, W. Jun, T. Wenhong, *A scheme to enhance security of RFID middleware*, 2010 International conference on Communications, Circuits and Systems (ICCCAS), 28-30 July 2010, p.147-149

[EPCALE1] *The Application Level Events (ALE) Specification, Version 1.1.1, Part I: Core Specification*, EPCglobal Inc.

[EPCALE2] *The Application Level Events (ALE) Specification, Version 1.1.1, Part II: XML and SOAP Bindings,* EPCglobal Inc.

[EPCC1G2] *EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communication at 860 MHz – 960 MHz – Version 1.2.0*, EPCglobal Inc.

[EPCLLRP] *EPCglobal™ Low Level Reader Protocol (LLRP) – Version 1.1*, EPCGlobal Inc., October 13rd, 2010

[FIN99] K. Finkenzeller, *RFID Handbook – Radio-Frequency Identification Fundamentals and Applications*, Wiley 1999, ISBN 0 471 98851 0

[FIN03] K. Finkenzeller, *RFID Handbook – Fundamentals and Applications in Contactless Smart Cards and Identification – Second Edition*, Wiley 2003, ISBN 0 470 84402 7

[FLO08] C. Floerkemeier, R. Pappu, *Evaluation of RFIDSim – a physical and Logical layer simulation engine*, 2008 IEEE International Conference on RFID, pp.350-356, 16-17 april 2008

[FLO09] C. Floerkemeier, S. Sarma, *RFIDSim – a physical and Logical layer simulation engine*, IEEE Transactions on Automation Science and Engineering, vol.6, no.1, pp.33-43, Jan. 2009

[FOSSTRAK] http://www.fosstrak.org

[FRI10] G. Fritz, V. Beroulle, M.D. Nguyen, O. Aktouf, D. Hély, *Read-Error-Rate evaluation for RFID system on-line testing*, 2010 IEEE 16th International on Mixed-Signals; Sensors and Systems Test Workshop (IMS3TW), pp.1-6, June 7-9 2010

[FRI11a] G. Fritz, V. Beroulle, O. Aktouf, D. Hély, *SystemC Modeling of RFID Systems for Robustness Analysis*, 2011 IEEE 19th International Conference on Software Telecommunications and Computer Networks (SoftCOM) – Symposium on RFID Technology and Applications, Sept. 2011

[FRI11b] G. Fritz, B. Maaloul, V. Beroulle, O. Aktouf, D. Hély, *Read rate profile monitoring for defect detection in RFID Systems*, 2011 IEEE International Conference on RFID-Technology and Applications (RFID-TA), Sept. 2011

[FRI11c] G. Fritz, V. Beroulle, O. Aktouf, D. Hély, *RFID System On-line Testing based on the evaluation of the Tags Read-Error-Rate*, Journal of Electronic Testing (JETTA), Springer Netherland, vol. 27, pp.267-276, June 2011, DOI: 10.1007/s10839-010-5191-6

[FRI12] G. Fritz, V. Beroulle, O. Aktouf, D. Hély, *Evaluation of a new RFID system performance monitoring approach*, Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.1439-1442; March 12-16 2012

[GLO06] B. Glover & H. Bhatt, *RFID Essentials*, O'Reilly 2006, ISBN 0 596 00944 5

[GUI06] `http://www.guideinformatique.com/lettrefiche-rfid_cas_praitque_marks_spencer-407.html`

[HAN05] Y. Han, H. Min, *System modelling and simulation of RFID*, Auto-ID Labs Whitepaper, Sept. 2005

[IEC61000-4-4] *IEC 61000-4-4 – Electromagnetic compatibility (EMC) – Part 4: Testing and measurement techniques – Section 4: Electrical fast transient/burst immunity test*

[INO04] S. Inoue, D. Hagiwara, H. Yasuura, *A Systematic approach for the reliability of RFID Systems*, 2004 IEEE Region 10 Conference TENCON 2004, 21-24 Nov. 2004, 183 - 186 Vol. 2

[ISO15693-1] *ISO/IEC FDIS 15693-1 – Identification cards – Contactless integrated circuit(s) cards – Vicinity cards – Part 1: Physical characteristics*

[ISO15693-2] *ISO/IEC FCD 15693-2 – Identification cards – Contactless integrated circuit(s) cards – Vicinity cards – Part2: Radio frequency power and signal interface*

[ISO15693-3] *ISO/IEC CD 15693-3 – Identification cards – Contactless integrated circuit(s) cards – Vicinity cards – Part 3: Anticollision and transmission protocol*

[ISO7498-1] *ISO/IEC 7498-1 – Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

[JAC09] R. Jacobsen, K. F. Nielsen, P. Popovski, T. Larsen, *Reliable identification of RFID tags using multiple independent reader sessions*, 2009 IEEE International conference on RFID, pp.64-71, 27-28 April 2009

[JEN94] E. Jenn, J. Arlat, M. Rimen, J. Ohlsson, J. Karlsson, *Fault Injection into VHDL Models: The MEFISTO Tool*, Proc. 24th International Symposium Fault-Tolerant Computing, IEEE 1994, pp66-75

[JIN06] L. Jin, T. Cheng, *Analysis and Simulation of UHF RFID System*,2006 8th International Conference on Signal Processing, vol.4, 16-20 Nov. 2006

[KAR07] T. Karygiannis, B. Eydt, G. Barber, L. Bunn, T. Phillips, *Guidelines for securing radio frequency identifications (RFID) Systems*, Recommendations of the National Institure of Standards and Technology, April 2007

[KEF08] N. Kefalakis, N. Leontiadis, J. Soldatos, K. Gama and D. Donsez, *Supply Chain Management and NFC Picking Demonstrations using the AspireRfid Middleware Platform*, demonstration in the scope of the ACM Middleware 2008 conference, Leuwen, Belgium, December 1-5, 2008

[KHE12] R. Kheddam, O-E-K. Aktouf and I. Parissis, *An extended LLRP model for RFID system test and diagnosis*, IEEE Fifth International Conference on Software Testing, Verification and Validation. Montreal, Canada, 2012, pp. 529-538.

[KHO07] R. Khouri, V. Beroulle, T.P. Vuong, S. Tedjini, *UHF RFID tag-antenna matching optimization using VHDL-AMS behavioralmodeling*, Analog Integrated Circuits and Signal Processing, Springer Netherlands, Volume 50, Number 2 / February, 2007, ISSN : 0925-1030, pp. 81-162

[KNI02] J. Knight, *Dependability of Embedded Systems*, International Conference on Software Engineering (ICSE'02), May 19-25, 2002, pp.685-686

[KRI07] P. Krishna, D. Husak, *RFID Infrastructure*, IEEE Applications & Practice, September 2007, pp.4-10

[LAP96] J.-C. Laprie, et al., *Le Guide de la Sûreté de Fonctionnement, 2e édition*,CEPADUES, City, 1996

[LIU06] L. Liu, S. Lai, *ALOHA-based Anti-collision Algorithms Used in RFID System*, International Conference on Wireless Communications, Networking and Mobile Computing, 2006, WiCOM 2006, pp.1-4, 22-24 Sept. 2006

[LOD06] G. Lodewijks, H.P.M.Veeke, A.M.L De La Cruz, *Reliability of RFID in Logistic Systems*, IEEE International Conference on Service Operations and Logistics, and Informatics, 2006. SOLI '06, 21-23 June 2006, pp.971-976

[MAJ07] J-L. Ma, X. Wie, J. Liu, *Security Analysis of RFID Based on Multiple Readers*, Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2007. IIHMSP 2007, 26-28 Nov. 200, pp.265-268

[MAL08] W. Mallouli, F. Bessayah, A. Cavalli, A. Benameur, *Security Rules Specification and Analysis Based on Passive Testing*, Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008, Nov. 30 2008-Dec. 4 2008, pp.1-6

[MCM70] P. McManamon, *HF Markov Chain Models and Measured Error Averages*, IEEE Transactions on Communication Technology, Vol. COM-18, No. 3, June 1970, pp. 201-208

[MIR09] L. Mirowski, J. Hartnett, R. Williams, *Tyrell: A RFID simulation platform*, 2009 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp.325-330, 7-10 Dec. 2009

[MIS07] S. Misera, H. Theodor V., A. Sieber, *Fault Injection Techniques and their Accelerated Simulation in SystemC*, 10th Euromicro Conference On Digital System Design Architectures, Methods and Tools (DSD 2007), 2007

[MUR04] D. Murfett, TAGSYS Australia Pty Ltd, *The challenge of testing RFID integrated circuits*, proceedings of the Second IEEE International Workshop on Electronic Design, Test and Applications (DELTA'04)

[NAT07] U.S. Natarajan, H. Shanmugasundaram, P. Deshande, C.S. Wah, *Rapid UHF RFID silicon debug and production testing*, ITC 2007, pp.1-10, 21-26 October 2007

[NEG07] M. Negreiros, L. Carro and A.A. Susin, *Reducing Test Time Using an Enhanced RF Loopback*, Journal of Electronic Testing: Theory and Application (JETTA), Special issue on analog, mixed-signal and RF testing, volume 23, Number 6, December 2007

[NGU11] M.D. Nguyen, G. Fritz, O. Aktouf, V. Beroulle, D. Hély, *Towards middleware-based fault-tolerance in RFID systems*. Proceedings of the 13th European Workshop on Dependable Computing (EWDC '11), p49-52. May 2011

[NIE02] E. Niel, E. Craye, *Maîtrise des risques et sûreté de fonctionnement des systèmes de production*, Paris : Hermès science publications, 2002 .- (53-Mayenne : Impr. Floch) , 412 p. : ill.

[PAR01] D. Paret, *Identification radiofréquence et cartes à puce sans contact – Description*, Dunod 2001, ISBN 2 10 004263 7

[PAR06] N. Park, H. Lee, H. Kim, D. Won, *A Security and Privacy Enhanced Protection Scheme for Secure 900MHz UHF RFID Reader on Mobile Phone*, 2006 IEEE Tenth International Symposium on Consumer Electronics, 2006. ISCE '06, pp.1-5

[PAR07] J. Park, J. Na, M. Kim, *A Practical Approach for Enhancing Security of EPCglobal RFID Gen2 Tag*, Future Generation Communication and Networking (FGCN 2007), 6-8 Dec. 2007, pp.436-441

[PAR08] D. Paret, *RFID en ultra et super hautes fréquences – UHF-SHF – Théorie et mise en œuvre*, Dunod 2008, ISBN 972 2 10 049347 0

[PETRA] `http://www.iaik.tugraz.at/content/research/rfid/petra`

[PIA10] C. Piao, Z. Fan, C. Yang, X. Han, *Research on RFID security protocol based on grouped tags and re-encryption scheme*,2010 IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS), 25-27 June 2010, p.568-572

[PIR08] S. Piramuthu, *Anti-collision algorithm for RFID tags*, Mobile and Pervasive Computing (CoMPC), 2008

[PRE08] S. Preradovic, N.C. Karmakar and I. Balbin, *RFID Transponders*, IEEE microwave magazine, pp.90-103, October 2008

[RAH07] A. Rahmati, L. Zhong, M. Hiltunen, R. Jana, *Reliability for RFID-Based Object Tracking Applications*, 37th IEEE International Conference on Dependable Systems and Networks (DSN'07), 25-28 June 2007, pp.113-118

[RAK11] RiccoRakotomalala, *Tests de normalité – Techniques empiriques et tests statistiques – Version 2.0*, support de cours, Université Lumière Lyon 2, 1er octobre 2011

[RIFIDI] http://www.rifidi.org

[SafeRFID] `http://lcis.grenoble-inp.fr/ctsys/projet-saferfid-surete-de-fonctionnement-des-systemes-rfid-266203.kjsp?RH=LCIS2_CTSYS-PRO`

[SCH83] F. C. Schoute, *Dynamic frame length ALOHA*, IEEE Trans. Commun, vol. COM-31, no.4, pp. 565-568, Apr. 1983

[SEO05] K. Seong Leong, M. Leng Ng, *The reader collision problem in RFID systems*, 2005 IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications Proceedings, pp. 658-661

[SOF07] O. Soffke, P. Zhao, T. Hollstein, M. Glesner, *Modelling of HF and UHF Technolofy for System and Circuit Level Simulations*, 2007 3rd European Workshop on RFID Systems and Technologies (RFIS SysTech), pp.1-6, 12-13 June 2007

[TSA69] S. Tsai, *Markov Characterization of the HF Channel*, IEEE Transactions on Communication Technology, Vol. COM-17, N° 1, February 1969, pp24-32.

[WAN09] J. Wang, D. Yang, *Design of a Multi-Protocol RFID Tag Simulation Platform Based on Supply Chain*, International Conference on Management and Service Science, 2009. MASS '09, vol., no., pp.1-4, 20-22 Sept. 2009

[WOE09] H. Woellik, *A Simulation tool for RFID anti-collision algorithms based on ALOHA*, 16th International Conference on Systems, Signals and Image Processing, 2009, IWSSIP 2009, pp.1-4,18-20 June 2009

[ZDN05] `http://www.zdnet.fr/actualites/des-hopitaux-marseillais-mettent-la-rfid-en-tube-39243313.htm`

[ZOR94] M. Zorzi, R. Rao, *Capture and retransmission control in mobile radio*, IEEE Journal on Selected Areas Communications, vol. 12, no. 8, pp.1289-1298, Oct 1994

VII Glossaire

Anticollision Technique permettant de diminuer ou supprimer le nombre de collisions

ATTV « Average Tag Traffic Volume » Méthode de test en ligne s'appuyant sur l'étude du flux du tag en fonction du temps

Collision Evénement se produisant lorsque plusieurs équipements émettent en même temps

Communication Passive Mode de communication sans fil lorsque le composant souhaitant envoyer un signal n'a pas de système d'émission. Il doit alors utiliser des techniques de retro-modulation ou de modulation de charge

Disponibilité Attribut de la sûreté de fonctionnement caractérisant l'aptitude d'un composant ou d'un système à délivrer le service attendu à un instant t

Fiabilité Attribut de la sûreté de fonctionnement caractérisant l'aptitude d'un composant ou d'un système à fonctionner pendant un intervalle de temps $[t_0; t]$

Frame Regroupement de slots

HF High Frequency ou Haute Fréquence – De 3 à 30 MHz – En RFID, on utilise généralement la fréquence 13,56 MHz

Lecteur Composant du système RFID permettant de lire ou d'écrire des données dans le tag et de les transmettre au middleware

LF Low Frequency ou Basse fréquence – De 30 à 300 kHz – En RFID, on utilise généralement des fréquences inférieures à 135kHz

Maintenabilité Attribut de la sûreté de fonctionnement caractérisant l'aptitude d'un composant ou d'un système à être réparé

Middleware Logiciel intermédiaire faisant le lien entre les composants physiques – ici Tags et Lecteurs – et l'application finale – gestion de stock, …

Modulation de charge En RFID HF, permet de transmettre une information sans composant d'émission en modifiant la charge vue par le destinataire

RETR « Read Error to Total Read » Méthode de test en ligne se basant sur l'étude du nombre d'erreurs de lecture sur le nombre total de lectures par unité de temps

Rétro-modulation ou backscattering. Technique permettant de transmettre de l'information à partir de la réflexion d'un signal

Robustesse Sûreté de fonctionnement par rapport aux fautes externes

RFID Radio Frequencies IDentification – Identification par radio fréquences

Sécurité-immunité Attribut de la sûreté de fonctionnement caractérisant l'aptitude d'un composant ou d'un système à garantir l'intégrité des données qu'il contient et à en empêcher l'accès ou les manipulations non autorisées

Sécurité-innocuité Attribut de la sûreté de fonctionnement caractérisant l'aptitude d'un composant ou d'un système à ne pas conduire à des accidents catastrophiques, en terme humain, matériel ou financier

SHF Super High Frequency ou Super Haute Fréquence – de 3 à 30 GHz – En RFID, on utilise généralement la fréquence 5,9 GHz

Slot Unité temporelle durant laquelle un élément a le droit d'émettre

Sûreté de fonctionnement c'est la « propriété qui permet aux utilisateurs du système de placer une confiance justifiée dans le service qu'il leur délivre » [LAP96]

Tag Composant d'un système RFID attaché à un objet ou une personne contenant ses informations

TEB Taux d'erreur binaire – Rapport entre le nombre de bits erronés et le nombre de bits total

Télé-alimentation Opération consistant à envoyer suffisamment d'énergie grâce à une onde RF à un composant pour lui permettre de fonctionner

UHF Ultra-High Frequency ou Ultra Haute Fréquence – de 0,3 à 3 GHz – En RFID, on utilise généralement les fréquence 433 MHz, 860 à 960 MHz et 2,45 GHz

VIII Liste des publications

VIII.1 Revues internationales avec Comité de lecture

1. G. Fritz, V. Beroulle, O. Aktouf, D. Hély, *RFID System On-line Testing based on the evaluation of the Tags Read-Error-Rate*, Journal of Electronic Testing (JETTA), Springer Netherland, vol. 27, pp.267-276, June 2011, DOI: 10.1007/s10839-010-5191-6

VIII.2 Conférences internationales avec comité de sélection et publication des actes

2. G. Fritz, V. Beroulle, M.D. Nguyen, O. Aktouf, D. Hély, *Read-Error-Rate evaluation for RFID system on-line testing*, 2010 IEEE 16th International on Mixed-Signals; Sensors and Systems Test Workshop (IMS3TW), pp.1-6, June 7-9 2010
3. M.D. Nguyen, G. Fritz, O. Aktouf, V. Beroulle, D. Hély, *Towards middleware-based fault-tolerance in RFID systems*. in Proceedings of the 13th European Workshop on Dependable Computing (EWDC '11), p49-52. May 2011
4. G. Fritz, V. Beroulle, O. Aktouf, D. Hély, *SystemC Modeling of RFID Systems for Robustness Analysis*, 2011 IEEE 19th International Conference on Software Telecommunications and Computer Networks (SoftCOM) – Symposium on RFID Technology and Applications, Sept. 2011
5. G. Fritz, B. Maaloul, V. Beroulle, O. Aktouf, D. Hély, *Read rate profile monitoring for defect detection in RFID Systems*, 2011 IEEE International Conference on RFID-Technology and Applications (RFID-TA), Sept. 2011
6. G. Fritz, V. Beroulle, O. Aktouf, D. Hély, *Evaluation of a new RFID system performance monitoring approach*, Design, Automation & Test in Europe Conference & Exhibition (DATE), pp.1439-1442; March 12-16 2012

Annexes

Table des matières

A Classification détaillée des tags

Plusieurs catégories de tags se distinguent [PAR08, PRE08]. En effet, il est possible de les classer en fonction de plusieurs critères qu'illustre la Figure A-1.

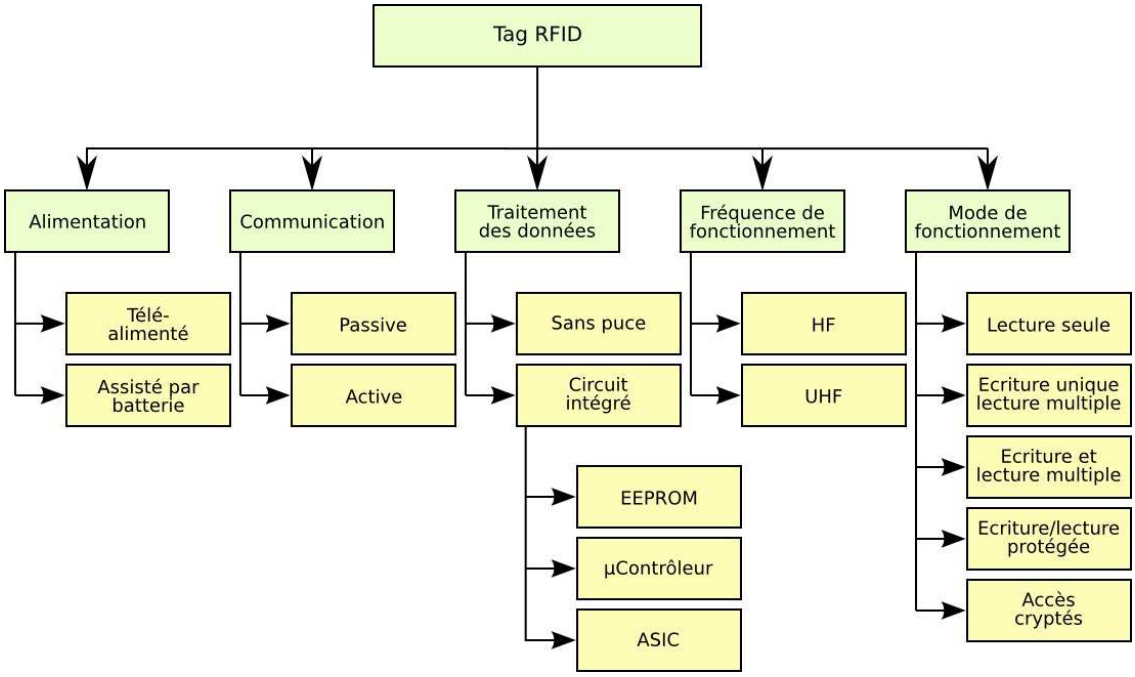


Figure A-1 : Classification des tags RFID

La première classification des tags RFID est faite par leur alimentation. En effet, comme tout circuit électronique, celui intégré dans le tag a besoin d'énergie afin de fonctionner correctement et d'assurer sa mission de communication. En RFID, il existe principalement deux techniques d'alimentation du tag :

1. l'alimentation est réalisée grâce à l'énergie transmise par l'onde provenant du lecteur. On parle alors de *télé-alimentation* ;
2. lorsque l'alimentation ne peut être réalisée par l'énergie transmise par l'onde provenant du lecteur, le tag est assisté par une batterie fournissant l'énergie nécessaire à son bon fonctionnement

Le premier cas est celui utilisé dans la plupart des applications RFID. En effet, cela permet d'avoir des tags peu chers, peu volumineux et très légers. Les tags reçoivent donc l'énergie nécessaire à leur alimentation grâce au champ magnétique/électromagnétique provenant du lecteur : ils sont donc *télé-alimentés*. L'énergie reçue par le tag doit être suffisante pour assurer son bon fonctionnement dans l'intégralité de ses fonctionnalités. Du fait des contraintes liées aux différentes normes et régulations, les distances de fonctionnement pour les tags télé-alimentés sont en général de quelques centimètres à quelques mètres. Ceci impose aussi des chalenges technologiques pour que l'électronique soit peu consommatrice. Mais cela permet d'avoir un équipement bon marché, totalement autonome, de faible dimension, ne nécessitant aucune maintenance particulière et d'une durée de vie importante.

La seconde technique d'alimentation est utilisée pour des cas plus particuliers de la RFID. En effet, l'utilisation d'une batterie pour alimenter le tag permet de libérer le lecteur de la contrainte de télé-alimentation du tag. Ceci permet alors d'atteindre des distances de

communication de plusieurs dizaines de mètre. En contrepartie, les tags deviennent plus volumineux, plus chers et surtout nécessitent une intervention régulière afin de remplacer l'élément énergétique, qu'il s'agisse d'une pile ou bien d'une batterie.

La seconde classification, couramment utilisée, est celle concernant le mode de communication du tag. Son mode de communication peut être *passif* ou *actif*.

Lorsque la communication est *passive*, le tag utilise l'onde émise par le lecteur afin de lui envoyer un message. Le tag arrive donc à se faire comprendre du lecteur *sans fonction émettrice à son bord*. Le principe de fonctionnement des communications passives est présenté dans le paragraphe II.1.1. Dans ce cas-là, les tags sont dits *passifs*. On rencontrera néanmoins couramment dans la littérature la confusion entre mode de communication et mode d'alimentation : en effet, les tags télé-alimentés sont considérés, comme passifs. Il est tout de même à noter, qu'à cause des contraintes liées à l'alimentation, un tag télé-alimenté utilise généralement une communication passive. Les communications passives sont en générale *Half-Duplex*, notée HDX, c'est-à-dire que les équipements envoient des données les uns après les autres. En effet, comme il sera vu par la suite, le tag utilise l'onde envoyée par le lecteur afin de communiquer. Ainsi, dans le cas d'une communication passive *Full-Duplex*, notée FDX, si l'onde envoyée par le lecteur contient déjà des données, celles-ci seront toujours présentes lors de l'envoi de données par le tag. Le lecteur devra alors trier les données présentes simultanément dans le signal afin de décoder le message provenant du tag. Ce tri de donnée nécessite des équipements complexes et coûteux à mettre en œuvre.

Contrairement aux tags passifs, les tags utilisant des techniques de communication actives possèdent un véritable émetteur. L'émission étant intégralement générée par le tag, la communication est *active*. Les tags *actifs* ont l'avantage d'avoir des distances de communication plus importantes que les tags passifs. Néanmoins, leur mode de fonctionnement se rapproche plus d'un système de communication RF « standard » et nécessite souvent l'assistance d'une batterie au sein du tag. Leur fonctionnement ne sera pas décrit ici.

On trouve aussi dans la littérature des tags dit *semi-actifs* ou *semi-passifs*. Ce terme, issu de la confusion décrite plus tôt, décrit généralement des tags passifs assistés par une batterie.

La troisième classification des tags concerne la manière dont ils gèrent les données. En effet, il existe actuellement une nouvelle catégorie de tags fonctionnant sans circuit électronique : ce sont des tags sans puce ou *chipless*. Ils exploitent alors les effets physiques dus à leur conception. Ils sont donc tous *passifs* et *télé-alimentés*. Le tag génère des données directement à partir de son architecture physique et de sa conception, données représentant l'identifiant unique de l'objet. Les tags chipless représentent actuellement une très faible partie des tags utilisés mais des études leur promettent une utilisation de plus en plus grande en raison de leur faible coût et de leur caractère difficilement reproductible, permettant l'authentification d'objet par exemple.

L'autre catégorie de tag, beaucoup plus répandu, est la catégorie des tags avec circuit intégré. Ces tags comportent alors de l'électronique embarquée fortement contrainte afin de répondre aux requêtes envoyées par le lecteur. Dans cette catégorie, il existe une grande disparité dans les fonctionnalités du tag : en fonction du circuit intégré, il sera alors possible d'avoir des fonctionnalités allant d'une simple sauvegarde d'un identifiant jusqu'à des applications de cryptographie, permettant alors l'utilisation des tags dans des applications critiques telles que le paiement sans contact. On peut distinguer néanmoins trois sous-catégories : les tags contenant simplement une EEPROM (en plus de la logique permettant de répondre au lecteur), simple équipement mémorisant une faible quantité de données ; les tags équipés d'un microcontrôleur, qui, de par leur équipement, peuvent effectuer des calculs ou embarquer des capteurs, étendant leurs fonctionnalités à davantage que la mémorisation d'un

simple identificateur ; et finalement les tags ayant une électronique dédiée – ASIC – : l'architecture du tag est alors étudiée spécifiquement pour une application et optimisée en vue de cette application. L'avantage de cette dernière catégorie est la possibilité de proposer des services spécifiques à une application, tout en garantissant une bonne autonomie et de faibles dimensions. De nos jours, la plupart des tags passifs télé-alimentés sont équipés d'ASIC.

Le quatrième critère correspond à la fréquence de fonctionnement du tag. Cette classification reprend la classification des systèmes RFID.

Enfin, la dernière classification de tags correspond à une classification en fonction du mode de fonctionnement vis-à-vis de l'accès aux données. Les tags les plus simples proposent uniquement un accès en lecture – tags en lecture seule. L'identifiant est alors stocké au moment de la production et ne pourra être changé au cours du temps. On trouve aussi sur le marché des tags permettant à l'utilisateur de configurer une seule fois, lors de l'initialisation, l'identifiant en fonction de ses besoins. Cet identifiant sera ensuite accessible en lecture seule. Ces tags sont qualifiés de tags à « Écriture unique, Lecture multiple ». Une évolution de ces tags est le tag à « Écriture et lecture multiple ». Ces tags permettent aux utilisateurs de modifier les données qu'ils contiennent au cours du temps. Pour des problèmes de confidentialité et d'intégrité des données, des tags proposent un accès protégé aux données, soit en lecture, soit en écriture, soit les deux. L'accès à la mémoire, en écriture ou en lecture, est alors soumis à condition – mot de passe, timing, … Une fois l'accès autorisé, les données circulent en clair. Si dans la plupart des cas cela ne pose pas de problème, il arrive dans certains cas particuliers – paiement sans contact, contrôle d'accès … – que les données doivent être masquées durant le transfert. On utilisera alors des tags à accès crypté. Ces tags utilisent des algorithmes cryptographiques propriétaires, ou normalisés symétriques – à clef privée type DES, AES – ou alors asymétriques – à clef publique type RSA, ECC.

B Autres codages bit

Les codages bit permettent des représentations physiques des bits. Dans cette annexe, différents codages bits sont décrits.

B.1 Codage bit NRZ

Le codage NRZ, pour Non-Retour à Zéro, est le plus simple des codages bit : le bit « **0** » est codé par un état bas du signal et le « **1** » par un état haut. Tous les bits ont la même durée, comme le montre la Figure B-1. Néanmoins, ce codage est peu utilisé en RFID. En effet, il y a très peu de transitions, surtout dans le cas de transmission de plusieurs bits identiques à la suite. De plus, en supposant l'équiprobabilité de la présence des bits – *i.e.* autant de « **1** » que de « **0** » dans le message – son efficacité énergétique est assez moyenne. En effet, seule la moitié des bits permettent de fournir le maximum d'énergie au tag. Dans la réalité, il y a rarement une succession de « **01** », mais plutôt des successions de « **1** » suivies de successions de « **0** ». L'alimentation du tag n'est donc pas constante et est totalement dépendante de la suite de bits dans le signal. Du plus, lors de longue séquences de **0** ou de **1**, il n'y a pas de transition, il est alors difficile pour le destinataire du message de se synchroniser avec l'émetteur.

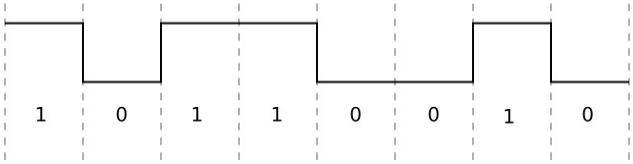


Figure B-1 : Codage bit « NRZ »

B.2 Codage bit Manchester

Afin d'augmenter le nombre de transition, un autre codage a été proposé : le codage *Manchester*. Ce codage, basé sur le précédent, divise le bit en deux parties : la première moitié du bit est identique au codage NRZ, la seconde moitié étant l'inverse comme le montre la Figure B-2. Ainsi, chaque bit contient une transition en son milieu. De plus, l'alimentation du tag est stable : il y a autant de moment à l'état haut qu'à l'état bas. Néanmoins, son efficacité énergétique est assez mauvaise. Ce codage a néanmoins un très gros avantage : il permet de détecter la présence de plusieurs émissions simultanées, comme l'explique la section 0. Il est donc principalement utilisé pour les communications tag vers lecteur.

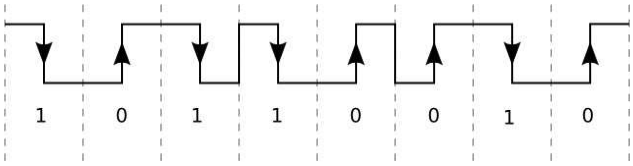


Figure B-2 : Codage bit « Manchester »

B.2.1 Manchester codé sous-porteuse

Une variante du codage bit *Manchester* permettant d'augmenter significativement le nombre de transitions a été proposée : le codage *Manchester* est couplé avec une ou deux sous-porteuses. Dans le cas du codage *Manchester codé sous-porteuse* avec *une* sous-porteuse, le signal du codage *Manchester* classique et celui de la sous-porteuse sont couplés. Ainsi, lorsque le signal *Manchester* classique est à l'état haut, la sous-porteuse est présente

dans le signal *Manchester codé sous-porteuse*. Dans le cas du codage bit *Manchester codé sous-porteuse* avec *deux* sous-porteuses, l'état haut du codage *Manchester* classique est remplacé par la première sous-porteuse, et l'état bas par la seconde sous-porteuse. La Figure B-3 illustre ces codages bits. Les nombreuses transitions rendent se codage idéal pour la communication tag vers lecteur, facilitant la détection, même dans un signal faible.

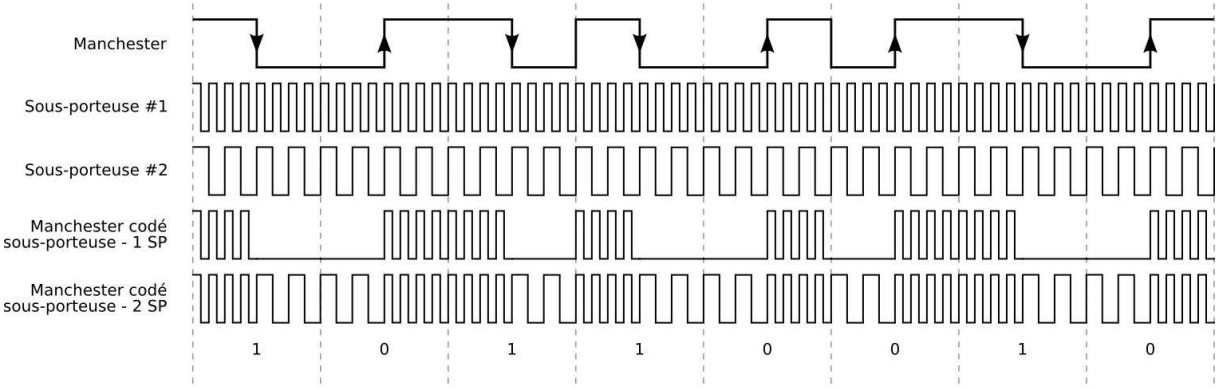


Figure B-3 : Codage bit « Manchester », « Manchester codé avec une sous-porteuse » et « Manchester codé avec deux sous-porteuses »

B.3 Codage Miller

Le codage Miller, représenté sur la Figure B-4, possède :

- une transition au milieu du bit pour un « **1** » ;
- aucune transition en milieu de bit pour un « **0** » ;
- une transition au début du bit pour un « **0** » si le bit précédant est aussi un « **0** ».

Ce codage bit possède donc de très régulières transitions, facilitant la synchronisation entre les différents acteurs de la communication.

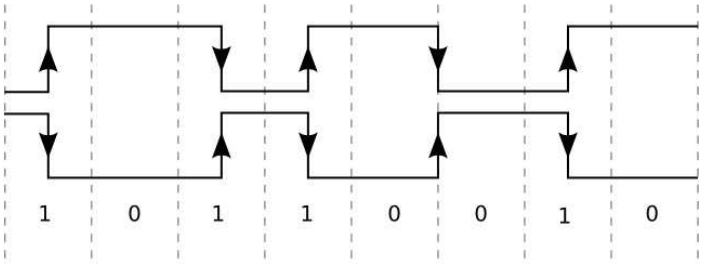


Figure B-4 : Codage bit « Miller »

B.3.1 Miller modifié

Le codage Miller modifié est basé sur le codage Miller vu précédemment. Les transitions sont remplacées par des impulsions. La Figure B-5 donne une représentation de ce codage.

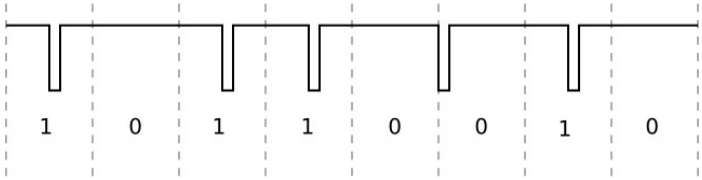


Figure B-5 : Codage bit « Miller modifié »

Ce codage a l'avantage d'avoir de longues phases durant lesquelles il ne se passe rien et de brefs instants durant lesquels il se produit une impulsion. Ce codage permet donc une bonne

télé-alimentation du tag. De plus, le fait d'avoir des transitions à quasiment chaque bit, rend la détection de ces derniers aisée.

B.3.2 Miller codé sous-porteuse

Une autre variante du codage *Miller* est le codage *Miller codé sous-porteuse*. Un sur-multiple de valeur *M* d'une sous porteuse est utilisée. Lorsque le codage Miller est à l'état haut, le sur-multiple de la sous-porteuse est utilisé, et lorsque le codage Miller est à l'état bas, l'inverse du sur-multiple de la sous-porteuse est utilisé. La Figure B-6 donne des exemples de séquence avec différents sur-multiples : $M = 2$, $M = 4$ et $M = 8$.

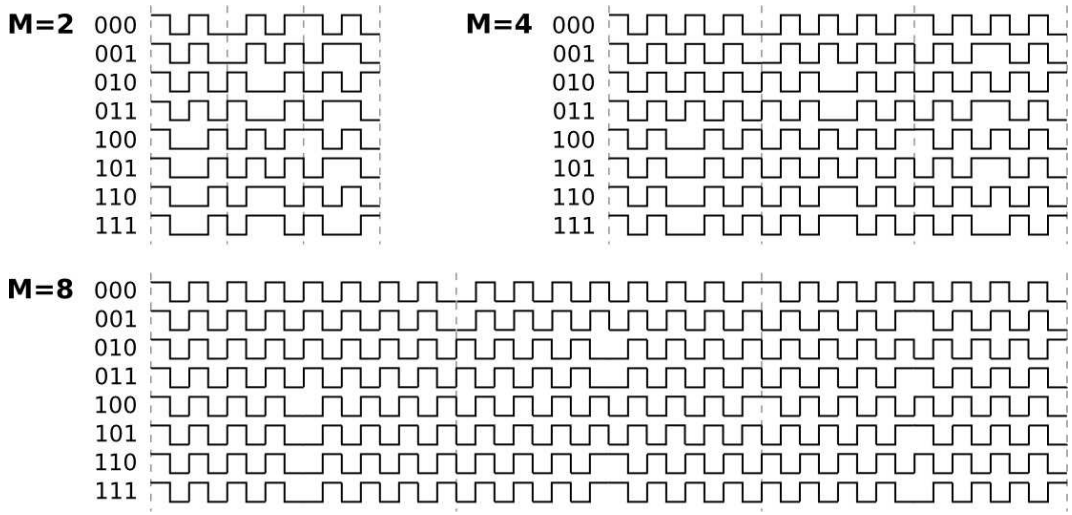


Figure B-6 : Codage bit « Miller codé sous-porteuse »

B.4 Codage « AC »

Le codage bit « AC » code les bits 0 avec une transition descendante au milieu du bit et les bits 1 avec deux transitions descendantes. La Figure B-7 donne un exemple d'une information codée avec ce dernier. Son nom vient du fait qu'il a été étudié afin de facilité la détection des collisions – AC pour AntiCollision. Il est donc principalement utilisé dans la liaison tag vers lecteur.

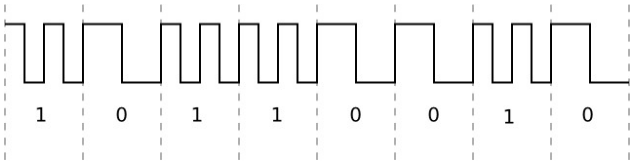


Figure B-7 : Codage « AC »

B.5 Codage PIE

Le codage PIE signifie « *Pulse Interval Encoding* ». Comme son nom l'indique, c'est l'intervalle entre deux impulsions qui va permettre d'identifier le bit codé : le bit « **1** » est deux fois plus long que le bit « **0** ». Ainsi, le « **0** » dure une unité de temps appelé *Tari* dans la norme EPC Classe 1 Génération 2 [EPCC1G2], et le « **1** » dure deux *Tari*. Au début de chaque symbole est présente une impulsion pour identifier le début de celui-ci. Pour décoder ce codage bit, il suffit de compter le temps entre deux impulsions : si le temps est égal à un Tari, le symbole représente un « **0** », s'il dure deux Tari, il représente un « **1** ». La Figure B-8 montre la représentation des bits avec ce codage. De par sa construction, ce codage permet d'être à l'état haut plus de la moitié du temps, offrant ainsi des phases d'alimentation plus

importantes que les codages précédents. L'inconvénient de ce genre de codage – dont la durée des symboles n'est pas fixe – est qu'il est difficile de déterminer le temps d'une communication. En effet, ce temps est dépendant du nombre de « **0** » et de « **1** » dans le message à transmettre.

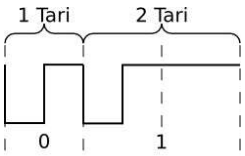


Figure B-8 : Codage bit « PIE »

B.6 Codage bit par position

Ce *codage de position* représente plusieurs bits par symbole. Le symbole est divisé en autant de bits qu'en est composé le symbole. Les positions des impulsions indiquent quels sont les bits à 1 dans le symbole. La Figure B-9 donne deux exemples de codage bit par position. Le premier symbole comporte 3 impulsions : une sur la zone du bit 6, une autre sur la zone du bit 4 et finalement une sur la zone du bit 2, codant alors l'octet « **0010 1010** ». Le second symbole est composé de deux impulsions. Une sur le bit 5 et une sur le premier bit, codant alors l'octet « **0001 0001** ».

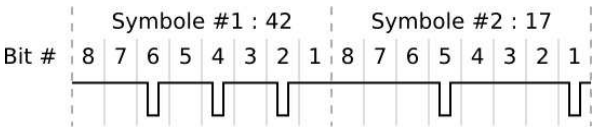


Figure B-9 : Codage bit par position

B.7 Codage de mots par position

Ce *codage de position* représente plusieurs bits par symbole. Le symbole est alors divisé en autant de parties qu'il y a de combinaisons binaires à coder. Afin d'identifier la combinaison binaire transmise, une impulsion est présente dans le symbole au niveau de la partie la codant. Pour illustrer ce codage bit, nous prendrons un cas particulier : le codage bit de position « *1 parmi 4* ». Celui-ci est capable de transmettre deux bits par symboles : il y a donc 4 combinaisons binaires possibles – **00**, **01**, **10**, **11**. Le symbole est donc divisé en quatre parties. Si l'impulsion du symbole se trouve dans son premier quart, le symbole représente « **00** », si elle se trouve dans le deuxième quart, le symbole représente « **01** », dans le troisième quart, « **10** » et finalement dans le dernier quart « **11** ». Les quatre symboles de ce codage bit sont représentés sur la Figure B-10. On trouve aussi dans les systèmes RFID le codage de position « *1 parmi 256* ». Le principe reste le même, à l'exception du nombre de bits codés par symbole : dans le cas du codage de position « 1 parmi 256 », chaque symbole code 8 bits, soit 256 combinaisons différentes. Le symbole est donc découpé en 256 parties. Le moment de l'impulsion dans le symbole déterminera quel est la combinaison de 8 bits codés. Ces codages permettent une bonne alimentation du tag : en effet, dans le cas du codage « 1 parmi 4 », l'état haut représente 75% du signal et dans le cas du codage « 1 parmi 256 », l'état haut représente 99% du signal. Néanmoins, la quantité de données pouvant être envoyée par intervalle de temps est très faible. Ces codages sont principalement utilisés dans des applications où les temps de communications ne sont pas critiques.

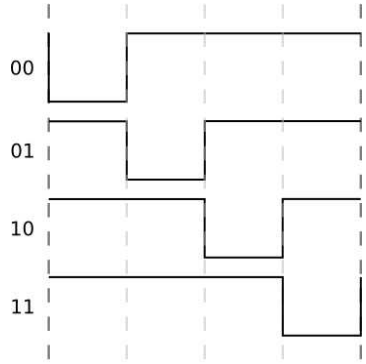


Figure B-10 : Codage de position « 1 parmi 4 »

C Principe physique de fonctionnement à 13,56 MHz – HF

Dans le cas de la RFID HF, les distances de fonctionnement, *i.e.* les distances entre les antennes du lecteur et du tag, sont relativement petites par rapport la longueur d'onde. En effet, d'après l'équation $\lambda = \frac{c}{v}$, la longueur d'onde utilisée par les systèmes RFID HF est de l'ordre de 22 mètres, ce qui est beaucoup plus grand que la distance entre les antennes du lecteur et du tag. Le système fonctionne alors en champ proche. Le système utilise alors un couplage magnétique pour télé-alimenter le tag et transférer les données [PAR01]. Le fonctionnement est similaire au fonctionnement d'un transformateur ; chaque antenne est alors une des bobines du transformateur. La **Erreur ! Source du renvoi introuvable.** montre le principe de télé-alimentation utilisée en champ proche. Le lecteur est la source d'énergie, connectée au primaire du transformateur et le tag est la charge, connectée au secondaire du transformateur.

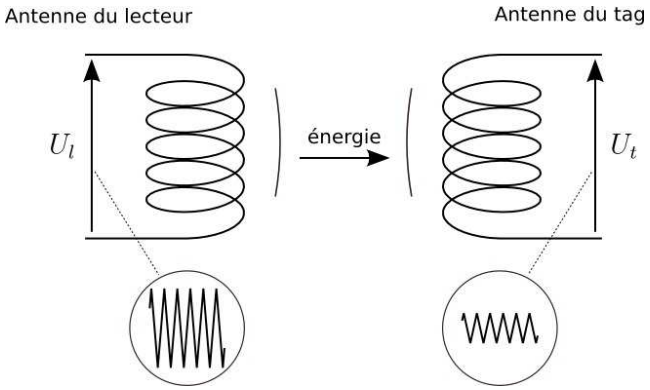


Figure C-1 : Transfert d'énergie en champ proche – couplage magnétique

Lorsque le lecteur souhaite envoyer des informations au tag, il module le signal qu'il émet en fonction des données à envoyer[1]. Le tag n'a plus qu'à démoduler le signal reçu afin de déterminer le message envoyé.

Dans le cas où le tag souhaite envoyer des informations au lecteur, il ne peut pas utiliser d'émetteur car les tags passifs n'en sont pas équipés. Il doit alors trouver un autre moyen que l'émission de signaux pour transmettre ce qu'il a à transmettre. La technique la plus utilisée est la *modulation de charge* et principalement la modulation de charge *résistive*. En effet, en champ proche, le tag correspond à une charge pour le lecteur. En venant modifier sa charge résistive, le tag modifie sa consommation d'énergie, ce qui tend à modifier la valeur du courant dans le circuit primaire, *i.e.* dans le circuit de l'antenne du lecteur. Le tag va alors utiliser cette modification de charge pour moduler le courant présent dans le circuit de l'antenne du lecteur. En observant cette modification, le lecteur est alors capable de décoder les informations envoyées par le tag – **Erreur ! Source du renvoi introuvable.**.

[1] Voir paragraphe II.1.2.2 qui traite de la modulation des signaux en RFID.

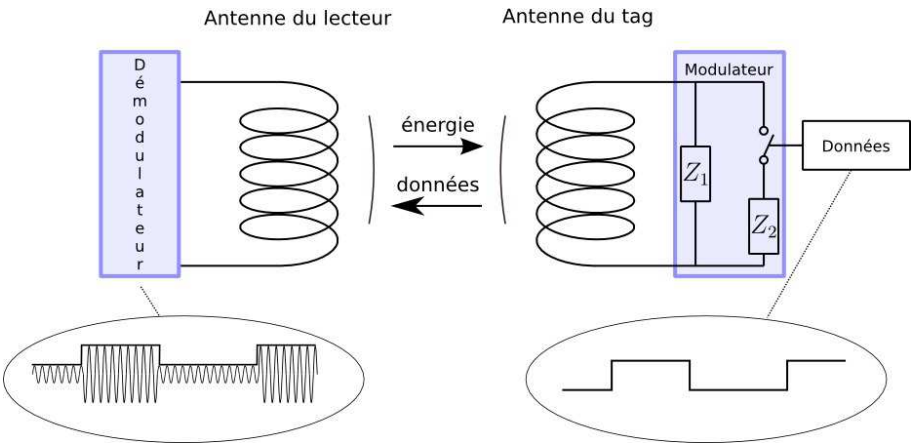


Figure C-2 : Modulation de charge

D Protocole anticollision probabiliste – cas de la norme ISO15693

La norme ISO-15693[ISO15693-3] est une variante de l'arbre binaire de recherche : elle prévoit le parcours d'un arbre en divisant les tags en sous-groupes codés sur 4 bits, à partir des bits de poids faibles de l'identifiant.

L'identifiant unique (UID), représenté sur la Figure D-1, est de 64 bits. Les 8 bits de poids forts sont toujours égaux à 0xE0 ; les 8 bits suivants correspondent à l'identifiant du fabricant du tag, et les bits restants correspondent à l'identifiant unique de ce tag attribué par le fabricant.

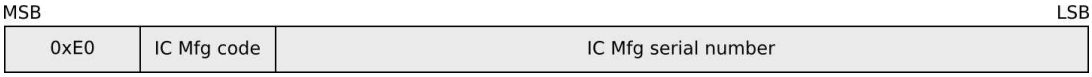


Figure D-1 : Identifiant Unique (UID) des tags ISO-15693

Pour réaliser l'inventaire – *i.e.* l'algorithme anticollision – le lecteur dispose de 2 commandes particulières et d'un signal :

1. la commande « **INVENTORY** » ;
2. la commande « **STAY QUIET** » ;
3. le signal « **END OF FRAME** », abrégé en « **EOF** ».

La première commande va permettre de configurer et de démarrer l'algorithme anticollision. Le lecteur va ainsi pouvoir vérifier la présence de tag dans son champ, qui est la première étape du parcours de l'arbre de recherche. Il va aussi pouvoir séparer les tags en 16 sous-groupes en fonction d'une partie de leurs identifiants. Pour cela, le lecteur va renseigner un masque de 0 à 64 bits et seuls les tags dont les bits de poids faibles de l'UID respectent ce masque participeront à l'inventaire.

La seconde commande va permettre d'endormir le tag afin que celui-ci ne participe plus aux inventaires suivants. Ainsi, les tags déjà identifiés ne perturberont pas l'inventaire en cours ni les inventaires futurs.

Enfin, le signal « **EOF** », présent à la fin de chaque transmission classique, est ici utilisé seul pour synchroniser les tags avec l'inventaire. A chaque fois que le lecteur est prêt à passer au groupe suivant, il envoie un « **EOF** » seul.

Ainsi, le lecteur aura divisé le temps en intervalles appelés « **SLOTS** » et chaque groupe de tag tentera de communiquer avec le lecteur dans le slot attribué à son groupe.

Pour chaque slot, les tags comparent leur identifiant au numéro de slot concaténé au masque transmis par le lecteur – Figure D-2. Les tags, dont le numéro de slot et le masque sont égaux à la partie correspondante de l'identifiant, transmettent alors leur identifiant. S'il n'y a qu'un seul tag, il est identifié ; s'il y en a plusieurs, il y a collision, ce que le lecteur devra résoudre.

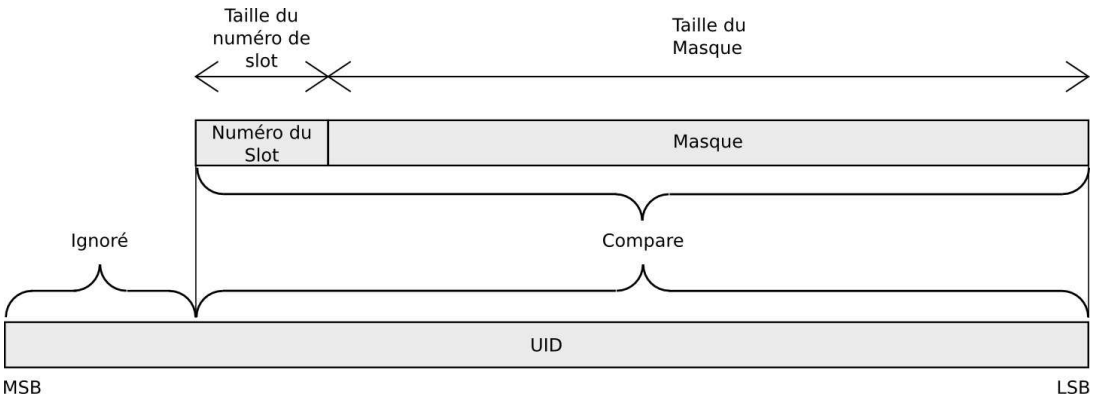


Figure D-2 : Détermination du numéro de slot en fonction du masque et de l'identifiant

La Figure D-3 donne un exemple d'exécution de cet algorithme. Quatre tags sont dans le champ du lecteur. Le Tableau D-I donne les identifiants uniques de ces tags. Ces identifiants ont volontairement été limités à 8 bits dans un souci de clarté.

Tableau D-I : Identifiant Unique des tags utilisés dans cet exemple

Numéro du tag	Identifiant unique
Tag #1	00100000
Tag #2	00100010
Tag #3	00110010
Tag #4	01000010

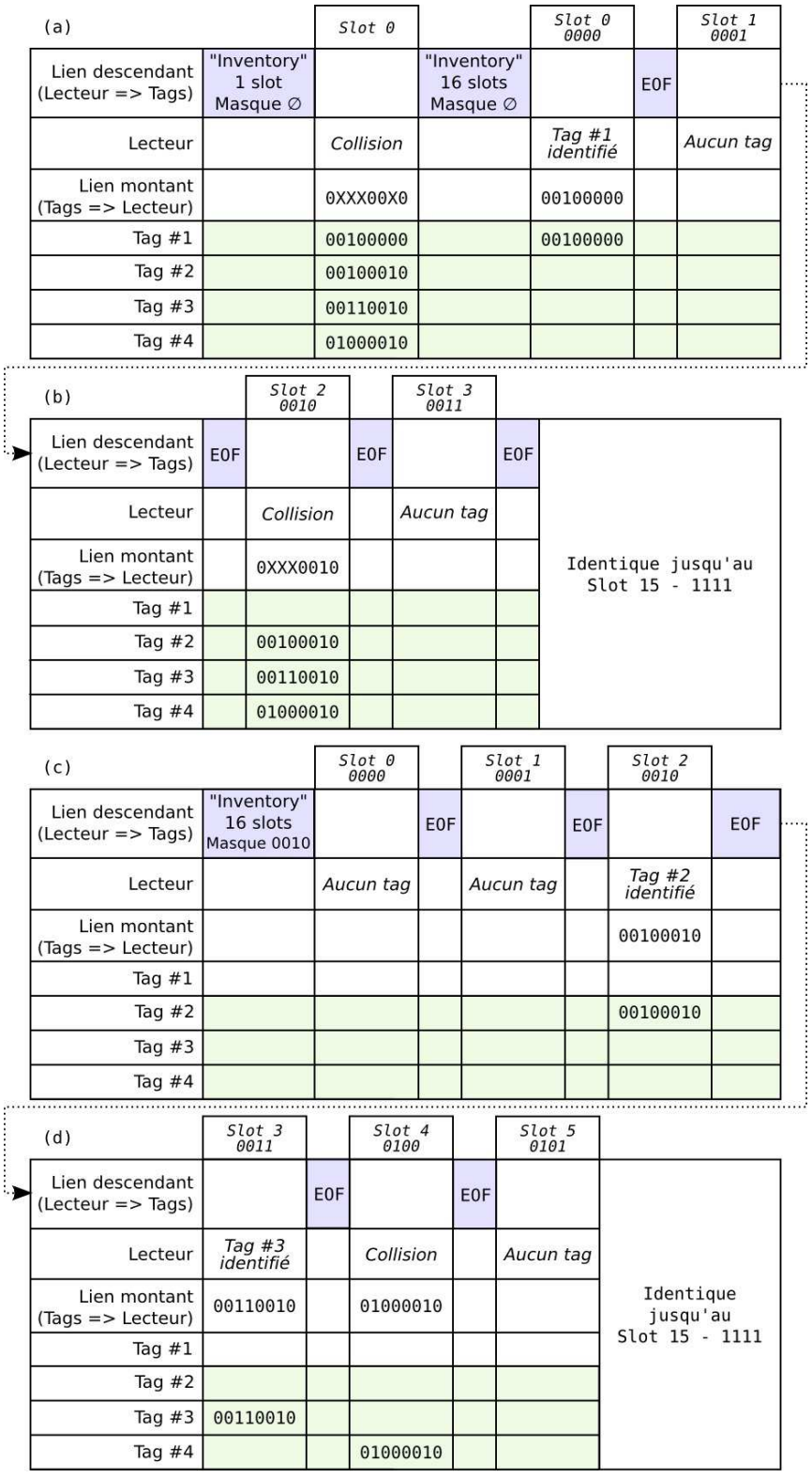


Figure D-3 : Exemple d'exécution de l'algorithme d'anticollision défini dans la norme ISO15693

On peut voir sur la Figure D-3 partie (a) que le lecteur vérifie dans un premier temps la présence de tags dans son champ. Pour cela, il envoie une commande « **INVENTORY** » indiquant qu'il effectuera un inventaire avec un seul slot et durant lequel tous les tags sont invités à communiquer – les tags participant à l'inventaire sont surlignés en vert. Il détecte alors une collision puisque 4 tags sont présents. On peut noter au passage, qu'il est possible de détecter la position de la collision. Ceci est dû à l'encodage bit utilisé dans cette norme [ISO15693-2] qui est le codage Manchester. Il est donc possible d'utiliser cette information afin d'optimiser le déroulement de l'algorithme.

Suite à cette détection, le lecteur divise alors tous les tags présents en 16 sous-groupes puis parcourt tous les slots afin que chaque groupe puisse communiquer avec lui. Il envoie donc une commande « **INVENTORY** » indiquant qu'il va diviser les tags en 16 sous-groupes, sans spécifier de masque particulier puisqu'il souhaite communiquer avec tous les tags. Juste après cette commande, le lecteur et les tags se trouvent dans la fenêtre temporelle – ou slot – numéro 0. Chaque tag va alors comparer son identifiant avec le numéro de slot concaténé au masque. Ici, la valeur du numéro de slot concaténé au masque vaut « **0000** » – slot 0 et pas de masque – et est de taille 4. Ainsi, chaque tag va comparer ses 4 bits de poids faibles avec cette valeur. Seul le tag #1 est concerné et envoie son identifiant. Comme il est le seul à être dans ce groupe, le lecteur est capable de l'identifier. Le lecteur envoie ensuite le signal « **EOF** » indiquant de passer au slot suivant. Dans ce slot, les tags comparent les 4 bits de poids faibles à la valeur du numéro de slot concaténée au masque qui vaut maintenant « **0001** ». Aucun tag ne correspond à cette contrainte. Le lecteur ne détecte aucun tag et passe alors au slot suivant en envoyant le signal « **EOF** » – Figure D-3, partie (b). La nouvelle valeur de comparaison est « **0010** » – slot 2 et pas de masque. Ici, 3 tags vont pouvoir communiquer. Or cela va engendrer une collision qui sera détectée par le lecteur. Le lecteur parcourt ensuite les slots restants en envoyant des signaux « **EOF** ». Aucun tag n'est présent, il ne détectera alors plus rien.

Afin de résoudre la collision du slot 2, le lecteur envoie une commande « **INVENTORY** » indiquant un inventaire à 16 slots durant lequel les tags doivent avoir les 4 bits de poids faibles égaux à « **0010** » - le masque est donc ici égal à « **0010** », Figure D-3, partie (c). Le lecteur parcourt alors les différents slots vides jusqu'au slot 2. Au slot 2, les tags doivent comparer les 8 bits de poids faibles à la valeur « **00100010** », ce qui correspond au numéro du slot 2 « **0010** » et au masque « **0010** ». Seul le tag #2 est présent et envoie son identifiant unique. Le lecteur passe donc au slot suivant. Ici, la valeur à comparer est « **00110010** » correspondant au slot 3 « **0011** » et au masque « **0010** ». Le tag #3 a son identifiant dont les 8 bits de poids faibles correspondent à cette valeur, et communique donc son identifiant. Il va de même pour le slot 4 – Figure D-3, partie (d).

Ainsi, tous les tags ont pu être identifiés. S'il y avait eu une ou plusieurs collision(s) dans le dernier inventaire, il aurait fallu réaliser un inventaire supplémentaire afin de résoudre la collision. Le masque aurait alors été de 8 bits, reprenant les 4 bits du premier masque et les quatre bits du slot collisionné.

Ces algorithmes ont l'avantage d'avoir un temps borné : le temps maximum d'inventaire est celui du temps de parcours de l'arbre. Il est aussi possible, lorsque l'on connait les tags présents, de prédire le temps que va prendre l'exécution de ce genre d'algorithme. De plus, leur caractère systématique rend la détection des tags sûre : chaque identifiant unique est parcouru afin de déterminer la présence de tag ou non.

Malheureusement, il apparaît clair que dans le cas d'un grand nombre de tags, le parcours systématique de chaque identifiant peut s'avérer assez long. Un autre type d'algorithme a donc été proposé afin d'améliorer le temps d'inventaire dans le cas d'un grand nombre de tags : les algorithmes probabilistes.

E AMDE pour les systèmes RFID

E.1 Découpage du système en fonction du modèle ISO/OSI

Tableau E-I : AMDE du système RFID – découpage en fonction du modèle ISO/OSI

Composant	Mode de défaillance	Cause possible	Composant(s) pouvant être à l'origine de la défaillance	Effet sur le système	Moyen de détection	Action correctrice		Evitement
						préventive	compensation	
Couche Physique	Non réception des signaux par un ou plusieurs tag(s) ou lecteur(s)	Tag(s) en dehors du champ du lecteur alors qu'il devrait y être	Antenne du lecteur, Canal	Perte d'information	Non détection de modification du champ, Analyse statistique	Répétition de la transmission jusqu'au succès	Redondance du tag	
		Perturbations EM	Canal, Antenne		Utilisation d'acquitement au niveau 2, Analyse statistique			Utilisation de puissance plus forte, de protocole physique plus robuste
		Agression extérieure (déformation, coupure, chocs, âge)	Antenne		Utilisation d'acquitement au niveau 2, Analyse statistique			Prendre des précautions, qualité des antennes, qualité physique
		défaillance interne	Antenne, Démodulateur, Canal, Alimentation		Analyse statistique			
	Mauvaise réception des	Perturbations EM	Canal, Antenne					

Composant	Mode de défaillance	Cause possible	Composant(s) pouvant être à l'origine de la défaillance	Effet sur le système	Moyen de détection	Action correctrice		Evitement
	signaux par un ou plusiers tag(s) ou lecteur(s)	Agression extérieure (déformation, coupure, chocs, âge)	Antenne					
		Défaillance interne (cf. Tag/Démodulateur)	Antenne, Démodulateur, Canal, Alimentation					
	Non émission des signaux par un ou plusieurs tag(s) ou lecteur(s)	Défaillance interne (cf. Tag/modulateur)	Antenne, Modulateur, Canal, Alimentation		Analyse statistique			
		Tag(s) en dehors du champ du lecteur alors qu'il devrait y être	Antenne du lecteur, Canal		Analyse statistique			
		Agression extérieur (déformation, coupure, chocs, âge)	Antenne		Analyse statistique			
	Mauvaise émission des signaux par un ou plusiers tag(s) ou lecteur(s)	Défaillance interne (cf. Tag/modulateur)						
		Agression extérieur (déformation, coupure, chocs, âge)						

Composant	Mode de défaillance	Cause possible	Composant(s) pouvant être à l'origine de la défaillance	Effet sur le système	Moyen de détection	Action correctrice		Evitement
	Emission continue (dialogue excessif)	Défaillance interne du tag	Modulateur	Canal surchargé, perte d'information, dégradation des performances	Analyse statistique	Désactivation automatique du tag		
						Retrait physique du tag du champ par une tierce partie		
		Défaillance interne du lecteur	Démodulateur, Antenne		Analyse statistique	Désactivation automatique du lecteur		
						Retrait physique du lecteur par une tierce partie		
		Répétition continue de tentatives d'émission	Logique interne du tag ou du lecteur		Comptage des erreurs d'acquitement, Analyse statistique			Limitation personnalisé du nombre de tentatives d'émission
	Absence de modulation	Agression extérieure (déformation, coupure, chocs, âge)	Modulateur	Le tag ne répond pas				
	Mauvaise modulation	Agression extérieure (déformation, coupure, chocs, âge)		Le tag émet une onde parasite				

Composant	Mode de défaillance	Cause possible	Composant(s) pouvant être à l'origine de la défaillance	Effet sur le système	Moyen de détection	Action correctrice		Evitement
	Aucune démodulation	Agression extérieure (déformation, coupure, chocs, âge)	Démodulateur	Le tag ne reçoit rien				
	Mauvaise démodulation	Agression extérieure (déformation, coupure, chocs, âge)		Le tag reçoit du bruit				
Couche Liaison	Transmission d'une trame erronée	Perturbations dûes à l'environnement (EM)	Canal	Information reçue non valide	Code détecteur d'erreur, Analyse statistique	Redondance temporelle efficace	Utilisation de code correcteur d'erreur	
						Redondance temporelle naturelle		
						Adéquation des contraintes temporelles de l'application		
		Erreur interne d'un composant de la couche Liaison	Logique interne du tag ou du lecteur		Redondance matérielle, Analyse statistique		Redondance matérielle	
Couche application du lecteur	Non détection de la disparition d'un tag	Bug logiciel ?	Logique interne					Inventaire régulier
	Non détection de l'apparition d'un tag							

Composant	Mode de défaillance	Cause possible	Composant(s) pouvant être à l'origine de la défaillance	Effet sur le système	Moyen de détection	Action correctrice		Evitement
	Détection d'un tag qui ne devrait pas être présent							
Couche application du tag	N'exécute aucune commande			Le tag ne répond pas	Acquittement, Analyse statistique			
	Exécute une autre commande que celle envoyée par le lecteur			Le tag ne répond pas correctement	Acquittement			Utilisation d'un code pseudo-aléatoire pour authentifier la communication (EPCGlobal)
	Exécute la commande envoyée par le lecteur plus une autre commande non désirée							
	Exécute la commande destiné à un autre tag							
	Mémoire erronée	Inversion/collage de bit(s)		Perte d'information	Code CRC (utilisé notamment par EPCGlobal)	Redondance d'information	Utilisation de code correcteur d'erreur	

E.2 Découpage fonctionnel du système

E.2.1 Tag

Tableau E-II : AMDE du système RFID – Découpage fonctionnel – Composant « tag »

Composant	Mode de défaillance	Cause possible	Effet sur le système	Moyen de détection	Action correctrice		Evitement
					préventive	compensation	
Antenne (Couche 1)	Non réception des signaux	Tag(s) en dehors du champ du lecteur alors qu'il devrait y être	Le tag ne répond pas	Analyse statistique			
		Défaillance interne du composant		Analyse statistique			
	Non émission des signaux	Tag(s) en dehors du champ du lecteur alors qu'il devrait y être		Analyse statistique			
		Défaillance interne du composant		Analyse statistique			
Alimentation	N'alimente pas alors qu'elle devrait	Puissance fournie par l'antenne insuffisante		Analyse statistique			
		Mauvaise orientation du tag				Changement par un tiers de l'orientation du tag	Utiliser plusieurs lecteurs, Utiliser un tunnel
	Alimente alors qu'elle ne devrait pas			Utilisation de codage de bit permettant la detection des collisions (codage Miller Codé sous porteuse)			
Mémoire (Couche 7)	Lecture erronée	Inversion/collage de bit(s)	Perte d'information	code CRC (utilisé notamment par EPCGlobal)	Redondance d'information	Utilisation de code correcteur d'erreur	
	Lecture impos-						

Composant	Mode de défail-	Cause possible	Effet sur le	Moyen de	Action correctrice		Evitement
	sible						
Logique de contrôle (Couche 2 - 7)	N'exécute aucune commande		Le tag ne répond pas	Acquittement			
	Exécute une autre commande que celle envoyée par le lecteur		Le tag ne répond pas correctement	Acquittement			Utilisation d'un code pseudo-aléatoire pour authentifier la communication (EPCGlobal)
	Exécute la commande envoyée par le lecteur plus une autre commande non désirée						
	Exécute la commande destiné à un autre tag						
Modulateur (Couche 1)	Absence de modulation		Le tag ne répond pas				
	Mauvaise modulation		Le tag émet une onde parasite				
Démodulateur (Couche 1)	Aucune démodulation		Le tag ne reçoit rien				
	Mauvaise démodulation		Le tag reçoit du bruit				
CRC (Couche 2)	Valide toujours la communication	Défaillance interne	Prise en compte d'informations erronées				
	Invalide toujours la communication		Perte d'information				

E.2.2 Lecteur

Composant	Mode de défaillance	Cause possible	Effet sur le système	Moyen de détection	Action corrective		Évitement
					préventive	par compensation	
Antenne (Couche1)	Non réception des signaux	Défaillance interne	Pertes d'information	Analyse statistique			
		Perturbation EM					
		Agression extérieur					
	Non émission des signaux	Défaillance interne					
		Perturbation EM					
		Agression extérieur					
Logique de contrôle (Couche 2-7)	N'arrive pas à résoudre les collisions	Défaillance interne					
		Perturbation EM					
Gestion des antennes (Couche 1-2)	Ne sélectionne pas la bonne antenne	Défaillance interne					
Liaison avec la station	N'émet rien	Défaillance interne		Analyse statistique			
		Défaillance de la connexion					
	Ne reçoit rien	Défaillance interne					
		défaillance de la connexion					
Modulateur (Couche 1)	Absence de modulation		Le lecteur ne répond pas				
	Mauvaise modulation		Le lecteur émet une onde parasite				
Démodulateur (Couche 1)	Aucune démodulation		Le lecteur ne reçoit rien				
	Mauvaise démodulation		Le lecteur reçoit du bruit				

F Exemple de composant SystemC dans SERFID

SERFID a été développé en **SystemC**, qui est une bibliothèque **C++**. Ainsi, la définition des composants peut-être, comme en **C++**, séparé en deux fichiers : les fichiers d'en-tête, dont l'extension en .h, contenant la déclaration du composant – entrées/sorties, processus internes, *etc.* – et les fichiers de définitions contenant le code **SystemC** décrivant les processus internes.

F.1 Canal de transmission classique

Ce composant permet de transférer une information entre un tag et un lecteur, en modifier la quantité d'énergie transmise d'un composant vers un autre, et en modifier la qualité intrinsèque du signal.

F.1.1 Fichier d'en-tête

Code F-1 : Fichier d'en-tête du composant « Channel » assurant la liaison entre les lecteurs et les tags

```
#include <systemc>

#include "Environment/GlobalEnvironment.h"
#include "Common/types.h"
#include "Common/Data/Data.h"
#include "Common/Information/Information.h"

/**
 * Represent the link between a tag and a reader
 *
 * Transmit power and information from reader to tag
 * Transmit power and information from tag to reader
 */
SC_MODULE(channel)
{
/* Déclaration des éléments privées du module */
protected:
      ::sc_core::sc_signal<t_distance> s_r; //!< distance in meter between
transmitter and receiver
      ::sc_core::sc_signal<bool> s_active; //!< determine if the channel
can transmit (true) or not (false)

      double m_lambda; //!< wavelength in meter
      double m_n; //!< pathloss exponent

      /**
       * Change data in the channel
       *
       * @param d Data in the channel
       */
      virtual void change_data(Data &d);
      /**
       * Change power from the information in the channel
       *
       * @param i Information in the Channel
       */
      virtual void change_power(Information &i);
      /**
       * Change quality of the information in the channel
```

```
	 *
	 * @param i Information in Channel
	 */
	virtual void change_quality(Information &i);

/* Déclaration des éléments publics du module */
public:
	// ios for reader
	::sc_core::sc_in<Information> inf_reader_tag_in; //!< information
which reader send to tag
	::sc_core::sc_out<Information> inf_tag_reader_out; //!< information
which reader receive from tag

	// ios for tag
	::sc_core::sc_out<Information> inf_reader_tag_out; //!< information
which tag receive from reader
	::sc_core::sc_in<Information> inf_tag_reader_in; //!< information
which tag send to reader

	/**
	 * SC_THREAD to transmit information from inf_reader_tag_in to
inf_reader_tag_out
	 */
	virtual void prc_transmit_inf_reader_tag();

	/**
	 * SC_THREAD to transmit information from inf_tag_reader_in to
inf_tag_reader_out
	 */
	virtual void prc_transmit_inf_tag_reader();

	/**
	 * Constructor
	 */
	SC_CTOR(channel):
		m_lambda(0.34),
		m_n(2)
	{
		s_r.write(1);
		s_active.write(true);

		/* Déclaration du processus de transfert de l'information
		 * du lecteur vers le tag et de sa liste de sensibilité :
		 * le processus se réveille lorsque l'information provenant
		 * du lecteur change, lorsque le channel devient actif ou
		 * inactif, ou lorsque la distance entre le tag et le
		 * lecteur change */
		SC_THREAD(prc_transmit_inf_reader_tag);
		sensitive << inf_reader_tag_in << s_active << s_r;

		/* Déclaration du processus de transfert de l'information
		 * du tag vers le lecteur et de sa liste de sensibilité :
		 * le processus se réveille lorsque l'information provenant
		 * du lecteur change, lorsque le channel devient actif ou
		 * inactif, ou lorsque la distance entre le tag et le
		 * lecteur change */
		SC_THREAD(prc_transmit_inf_tag_reader);
		sensitive << inf_tag_reader_in << s_active << s_r;
	}
```

```
        /**
         * Set the wavelength (meter)
         *
         * @param w new wavelength
         */
        void setWavelength(double w);

        /**
         * Get the wavelength (meter)
         *
         * @return Wavelength
         */
        double getWavelength();

        /**
         * Set pathloss exponent
         *
         * @param n new Pathloss exponent
         */
        void setPathlossExponent(double n);

        /**
         * Get pathloss exponent
         *
         * @return Pathloss Exponent
         */
        double getPathlossExponent();

        /**
         * Set distance between tag and reader (meter)
         *
         * @param r new distance
         */
        void setDistance(t_distance r);

        /**
         * Get distance between tag and reader (meter)
         *
         * @return distance
         */
        t_distance getDistance();

        /**
         * Checks if the channel is enabled.
         */
        bool isEnabled();

        /**
         * Enable channel, i.e. authorize the transfer of power and infor-
mations
         */
        void enable();

        /**
         * Disable channel, i.e. unauthorize the transfer of power and infor-
mations
         */
        void disable();

    /**
```

```
     * Toggle state of channel i.e enable it when it is disabled and disa-
ble it when it is enabled
     */
      void toggle();
   };
```

F.1.2 Fichier de définition

Code F-2 : Fichier de définition du composant « Channel » assurant la liaison entre les lecteurs et les tags

```
#include "channel.h"

#include <iostream>
#include <cmath>

#define PI 3.14159265

using namespace sc_core;
using namespace sc_dt;

/* Pas de changement des données */
void channel::change_data(Data &d)
{}

/* Diminution de la puissance en fonction de la distance entre le
 * lecteur et le tag et de la longueur d'onde */
void channel::change_power(Information &i)
{
      double D = pow(((4*PI*s_r.read())/m_lambda),m_n);
      i.setPower(i.getPower()/ D);
}

/* Diminution de la qualité du signal en fonction de l'environnement
 * global */
void channel::change_quality(Information &i)
{
      i.setQuality((i.getQuality()/2)*GlobalEnvironment::getInstance().getQ
uality());
}

/* Transmission de l'information du lecteur vers le tag */
void channel::prc_transmit_inf_reader_tag()
{
      while(1)
      {
          // Attente d'un évènement sur la liste de sensibilité
            wait();
            // Si le composant est actif
            if (s_active.read())
            {
                // Récupération de l'information envoyé par le lecteur
                  Information i=inf_reader_tag_in.read();

                  // Modification de la qualité
                  change_quality(i);

                  // Modification des données
                  Data d=i.getData();
                  change_data(d);
```

```
                    i.setData(d);

                    // Modification de la puissance
                    change_power(i);

                    // Envoie de l'information vers le tag
                    inf_reader_tag_out.write(i);
              }
              // Si le composant est inactif
              else
              {
                    // Envoie d'un message vide vers le tag
                    inf_reader_tag_out.write(Information::empty);
              }
        }
}

/* Transmission de l'information du tag vers le lecteur */
void channel::prc_transmit_inf_tag_reader()
{
        while(1)
        {
            // Attente d'un évènement sur la liste de sensibilité
              wait();
              // Si le composant est actif
              if (s_active.read())
              {
                  // Récupération de l'information envoyé par le tag
                    Information i=inf_tag_reader_in.read();

                    // Modification de la qualité
                    change_quality(i);

                    // Modification des données
                    Data d=i.getData();
                    change_data(d);
                    i.setData(d);

                    // Modification de la puissance
                    change_power(i);

                    // Envoie de l'information vers le lecteur
                    inf_tag_reader_out.write(i);
              }
              // Si le composant est inactif
              else
              {
                    // Envoie d'un message vide vers le lecteur
                    inf_tag_reader_out.write(Information::empty);
              }
        }
}

/* Modification de la distance */
void channel::setDistance(t_distance r)
{
        s_r.write(r);
}

t_distance channel::getDistance()
{
```

```
	return s_r.read();
}

/* Activation du composant */
void channel::enable()
{
	s_active.write(true);
}

/* Désaction du composant */
void channel::disable()
{
	s_active.write(false);
}

bool channel::isEnabled()
{
	return s_active.read();
}

/* Invertion de l'action du composant */
void channel::toggle()
{
	s_active.write(!s_active.read());
}

void channel::setWavelength(double w)
{
	m_lambda = w;
}

double channel::getWavelength()
{
	return m_lambda;
}

void channel::setPathlossExponent(double n)
{
	m_n=n;
}

double channel::getPathlossExponent()
{
	return m_n;
}
```

F.2 Canal de transmission spécifique : injection d'erreur binaire

Ce composant est une extension du précédent. Il permet de transférer les informations entre un tag et un lecteur, en injecter des erreurs binaires dans les données. Pour cela, il hérite de toutes les fonctionnalités du composant « channel » et redéfinit certaines fonctionnalités en fonction des besoins.

F.2.1 Fichier d'en-tête

Code F-3 : Fichier d'en-tête du composant « channel_with_ber » permettant de transférer des informations entre un lecteur et un tag avec injection d'erreurs binaire

```
#include "channel.h"
#include <ctime>
#include "Common/Data/Data.h"
#include "Common/Utils/Random.h"

/**
 * Specific link between reader and tag : exchanged information are modi-
fied :
 * bits are flips randomly according bit-error-rate (ber)
 */
class channel_with_ber: public channel // Héritage du composant "channel"
{
// Définition des éléments privées du composant
private:
      double m_ber;//!< The Bit Error Rate of the channel
      Random m_generator;//!< Random generator

      /**
       * Create an unique generator for this component
       *
       * @return a reference of an unique instance of Random class
       */
      static Random& getUniqueGenerator()
      {
            static Random _generator(SEED_CHANNEL_WITH_BER_UNIQUE);

            return _generator;
      }

protected:
      // Redéfinition de la fonction changeant les données de l'information
      virtual void change_data(Data &d);

// Définition des éléments publics du composant
public:

      // Indique que ce composant est un composant SystemC
      SC_HAS_PROCESS(channel_with_ber);

    /**
     * Contructor
     *
     * @param name Name of the component
     * @param ber Bit error rate
     * @param randomize seed for the generator
     */
      channel_with_ber( ::sc_core::sc_module_name name, double ber=1, int
*randomize=NULL ):
            channel(name),
            m_ber(ber)
      {
            if (randomize==NULL)
            {
                  m_generator.randomize(SEED_CHANNEL_WITH_BER);
            }
```

```
            else
            {
                  m_generator.randomize(*randomize);
            }
      }

    /**
     * Set the seed
     *
     * @param r the new seed
     */
    void setRandomize(long r)
    {
      m_generator.randomize(r);
    }

    /**
     * Return seed of the internal generator
     *
     * @return seed of the internal generator
     */
    long getRandomize()
    {
      return m_generator.getSeed();
    }

    /**
     * Set new Bit Error Rate
     *
     * @param ber new Bit Error Rate
     */
    void setBer(double ber)
    {
      m_ber=ber;
    }

    /**
     * Get Bit Error Rate
     *
     * @return Bit Error Rate
     */
    double getBer()
    {
      return m_ber;
    }
  };
```

F.2.2 Fichier de définition

Listing F-4 : Fichier de définition du composant « channel_with_ber » permettant de transférer des informations entre un lecteur et un tag avec injection d'erreurs binaire

```
#include <cstdlib>

using namespace std;
using namespace sc_core;
using namespace sc_dt;

// Redéfinition de la fonction modifiant les données
void channel_with_ber::change_data(Data &d)
{
      // Pour chaque champ de la donnée
      for (map<string, sc_uint_base*>::iterator it=d.begin();
                  it!=d.end(); it++)
      {
            int size = (*it).second->length();
            // Pour chaque bit du champ de la donnée
            for (int i=0; i < size; i++)
            {
                  // On détermine si il faut inverser le bit, en fonction
d'une loi uniforme
                  if (m_generator.uniform<double>(0,1) <=
(m_ber+GlobalEnvironment::getInstance().getBer()))
                  {
                        (*(*it).second)[i] ^= 1;
                  }
            }
      }
   }
```

G Paramètres de communication utilisés dans SERFID

Tableau G-I : Paramètres de la communication analogique entre le tag et le lecteur

Nom	Norme	Simulateur	Description
Tari	6,25µs<Tari<25µs	6,25µs	Intervalle de temps référence pour la communication lecteur->tag
Data-0	Tari	6,25µs	Temps d'un symbole '0' (encodage PIE) sens lecteur->tag
Data-1	1,5Tari < Data-1 < 2Tari	12,5µs	Temps d'un symbole '1' (encodage PIE) sens lecteur->tag
RTcal	2,5Tari < Tari < 3Tari	18,75µs	Symbole de calibration lecteur->tag
TRcal	1,1RTcal <TRcal< 3RTcal	37,5µs	Symbole de calibration tag->lecteur
DR	8 ; 64/3	8	Ratio de division du TRcal
M	1 ; 2 ; 4 ; 8	2	Nombre de cycles de sous-porteuse par symbole
TRext	0 (non) ; 1 (oui)	0 (non)	Utilisation d'un signal pilote
BLF	[EPCC1G2] Tableau 6.9	192kHz	Fréquence de la rétro-modulation
Tpri	1/BLF	5,21µs	Période rétro-modulation
T1	[EPCC1G2] Tableau 6.13	59,3µs	Temps entre la fin de transmission du lecteur et le début de transmission du tag
T2	[EPCC1G2] Tableau 6.13	15,625µs	Temps entre la fin de transmission du tag et le début de transmission du lecteur
T3	[EPCC1G2] Tableau 6.13	15,625µs	Temps d'attente du lecteur après **T1** avant de commencer une nouvelle commande
T4	[EPCC1G2] Tableau 6.13	37,5µs	Temps min entre 2 commandes du lecteur
RDR	26,7kbps < RDR < 128kbps	107kbps	Vitesse de communication lecteur->tag (en supposant l'équiprobabilité des données)
TDR	5kbps < TDR < 640kbps	96kbps	Vitesse de communication tag->lecteur
ID EPC		128bits	Taille de l'identifiant EPC
Codage bit lecteur → tag		PIE	Cf. section II.1.2.1, page 17
PIE Premable	12.5 + Tari + RTcal + TRcal	75µs	Préambule de communication Lecteur->Tag
PIE Frame-Sync	12.5 + Tari + RTcal	37.5µs	Frame de synchronisation Lecteur->Tag
Codage bit tag → lecteur		Miller codé sous-porteuse	Cf. annexe B
Miller Preamble	(4+12*TRext)*M/BLF + 6/TDR	104,2µs	Préambule utilisé lorsque la modulation utilisé par le tag est Miller
Miller EoS	2 bits	20,8µs	Trame de fin de communication lorsque la modulation utilisée par le tag est Miller

H Résultats de simulation de validation du nombre de slots

Dans cette annexe, les résultats de validation fonctionnelle du simulateur sont présentés. Pour cela, le nombre moyen de slots collisions, vides ou singletons est analysé en fonction du nombre de slots et du nombre de tags. Pour rappel, les équations (II.5), (II.6) et (II.7) permettent de calculer le nombre théorique de slots vides, collisions ou singletons et sont rappelées ici :

$$S(N,L) = N\left(1-\frac{1}{L}\right)^{N-1}$$
$$V(N,L) = L\left(1-\frac{1}{L}\right)^{N}$$
$$C(N,L) = L - S(N,L) - V(N,L)$$

Où ***S*** correspond au nombre de slots singletons, ***V*** au nombre de slots vides et ***C*** du nombre de slots collisions dans un frame de ***L*** slots en présence de ***N*** tags.

H.1 5 tags – 16 slots (Q=4)

Pour obtenir ces données, 250 tirages ont été réalisés.

Tableau H-I : Nombre de slots singletons, vides ou collisions pour un frame de 16 slots et 5 tags à inventorier

Type de slots	Théorie	Simulation			
		Moyenne	Ecart type	Min	Max
S(5,16)	3,9	3,84	1,20	1	5
V(5,16)	11,6	11,59	0,62	11	13
C(5,16)	0,5	0,572	0,59	0	2

H.2 0 tags – 256 slots (Q=8)

Pour obtenir ces données, 250 tirages ont été réalisés.

Tableau H-II: Nombre de slots singletons, vides ou collisions pour un frame de 256 slots et 0 tags à inventorier

Type de slots	Théorie	Simulation			
		Moyenne	Ecart type	Min	Max
S(0,256)	0	0	0	0	0
V(0,256)	256	256	0	256	256
C(0,256)	0	0	0	0	0

H.3 10 tags – 256 slots (Q=8)

Pour obtenir ces données, 250 tirages ont été réalisés.

Tableau H-III : Nombre de slots singletons, vides ou collisions pour un frame de 256 slots et 10 tags à inventorier

Type de slots	Théorie	Simulation			
		Moyenne	Ecart type	Min	Max
S(10,256)	9,65	9,552	0,87	6	10
V(10,256)	246,17	246,25	0,47	246	248
C(10,256)	0,17	0,22	0,43	0	2

H.4 100 tags – 256 slots (Q=8)

Pour obtenir ces données, 100 tirages ont été réalisés.

Tableau H-IV : Nombre de slots singletons, vides ou collisions pour un frame de 256 slots et 100 tags à inventorier

Type de slots	Théorie	Simulation			
		Moyenne	Ecart type	Min	Max
S(100,256)	67,88	68,24	5,83	55	80
V(100,256)	173,1	172,79	3,14	167	181
C(100,256)	15,02	14,97	2,82	9	21

H.5 256 tags – 256 slots (Q=8)

Pour obtenir ces données, 100 tirages ont été réalisés.

Tableau H-V : Nombre de slots singletons, vides ou collisions pour un frame de 256 slots et 256 tags à inventorier

Type de slots	Théorie	Simulation			
		Moyenne	Ecart type	Min	Max
S(256,256)	94,36	93,77	8,27	74	117
V(256,256)	94	94,18	5,21	82	106
C(256,256)	67,64	68,05	4,24	57	77

H.6 1000 tags – 256 slots (Q=8)

Pour obtenir ces données, 101 tirages ont été réalisés.

Tableau H-VI : Nombre de slots singletons, vides ou collisions pour un frame de 256 slots et 1000 tags à inventorier

Type de slots	Théorie	Simulation			
		Moyenne	Ecart type	Min	Max
S(1000,256)	20,04	20,62	4,20	9	32
V(1000,256)	5,11	5,18	1,86	0	10
C(1000,256)	230,85	230,30	4,27	220	241

I Outils mathématiques

I.1 Calcul CRC

Les différentes manières de calcul du CRC-16 sont présentées ici. Le Code I-1 donne l'implémentation en langage C du calcul du CRC-16 utilisée par les normes ISO-15693 [ISO15693-3] et EPC Classe 1 Génération 2 [EPCC1G2].

Code I-1 : Algorithme de calcul du CRC, implémentation en langage C

```
#define POLYNOMIAL      0x8408          // x^16 + x^12 + x^5 + 1
#define PRESET_VALUE    0xFFFF
#define CHECK_VALUE     0xF0B8
#define CALC_CRC            1
#define CHECK_CRC           0

/** Fonction qui calcule ou vérifie le crc16 d'une suite d'octets
 *
 * @param size            nombre d'octets contenus dans le tableau
 * @param array           tableau d'octets
 * @param calculate_or_check   CALC_CRC : calcul du crc; CHECK_CRC : vérifie le crc
 *
 * @return La valeur de CRC
 */
unsigned int calculate_crc(int size, unsigned char array[], int
calculate_or_check)
{
  unsigned int current_crc_value;
  int i,j;

  current_crc_value = PRESET_VALUE;

  for (i=0; i < size; i++)
  {
    current_crc_value = current_crc_value ^ ((unsigned int) array[i]);
    for (j=0; j < 8; j++)
    {
      if (current_crc_value & 0x0001)
      {
        current_crc_value = (current_crc_value >> 1) ^ POLYNOMIAL;
      }
      else
      {
        current_crc_value = (current_crc_value >> 1);
      }
    }
  }

  if (calculate_or_check==CALC_CRC)
  {
    return (~current_crc_value);
  }
  else
  {
    return (current_crc_value==CHECK_VALUE);
  }
}
```

La Figure I-1 donne un exemple de circuit de calcul de CRC-5. Ce circuit consiste en un registre de décalage avec des bouclages dépendant du polynôme de division. Après avoir initialisé le circuit, les données sont injectées via le signal **DATA** bit après bit à chaque front

d'horloge, en commençant par le bit de poids fort. La valeur du code CRC associé à la donnée est présente sur le signal **CRC[4..0]** une fois tous les bits injectés dans le circuit de calcul. Afin de valider la cohérence entre une donnée et son code CRC associé, ces derniers sont injectés dans le circuit de calcul en commençant par le bit de poids fort de la donnée. Si le signal **CRC[4..0]** est égal à **00000B**, c'est que le CRC et la donnée sont cohérents.

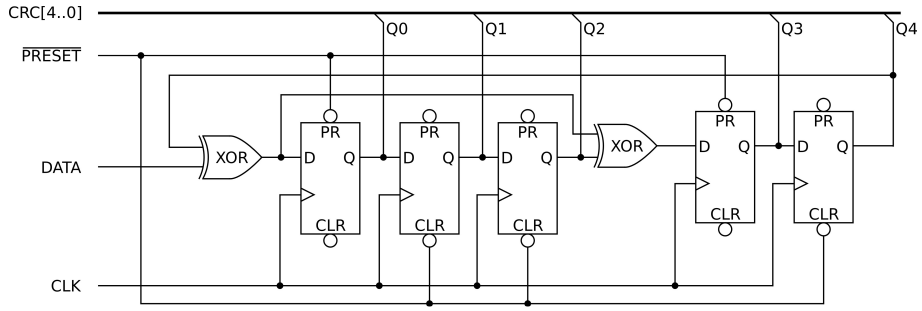


Figure I-1 : Exemple de circuit de calcul de CRC-5

I.2 Droite de Henry

La droite de Henry est une méthode graphique permettant de vérifier le comportement gaussien d'une distribution et de lire graphiquement la moyenne et l'écart type de cette distribution. [http://fr.wikipedia.org/wiki/Droite_de_Henry]

I.2.1 Principe

Soit X une variable gaussienne de moyenne $\bar{x}$ et de variance σ^2 et N une variable de loi normale centrée réduite. On a les égalités suivantes :

$$\begin{aligned} P(X < x) &= P\left(\frac{X-\bar{x}}{\sigma} < \frac{x-\bar{x}}{\sigma}\right) \\ &= P(N < t) \\ &= \Phi(t) \\ \text{avec} \quad t &= \frac{x-\bar{x}}{\sigma} \end{aligned} \tag{I.1}$$

où Φ est la fonction de répartition de la loi normale centrée réduite.

Pour chaque valeur x_i de X, on peut, à l'aide de la table de la fonction Φ :

- Calculer $P(X < x_i)$
- En déduire t_i tel que $\Phi(t_i) = P(X < x_i)$

Si la variable X est gaussienne, les points de coordonnées $(x_i; t_i)$ sont alignés sur la droite d'équation $t = \frac{x-\bar{x}}{\sigma}$.

I.2.2 Exemple

Lors de 10 inventaires applicatifs composés de 100 inventaires d'un groupe de tags, on obtient les taux de lecture suivants pour le 98e tag :

Tableau I-I : Résultat des inventaires applicatifs

	1er inv.	2e inv.	3e inv.	4e inv.	5e inv.	6e inv.	7e inv.	8e inv.	9e inv.	10e inv.
Nombre de détections	20	6	9	11	3	14	9	7	17	15

Afin de vérifier que cette distribution est gaussienne, on calcule pour différents x_i la probabilité $P(X < x_i)$ et on détermine le t_i associé à l'aide de la table de la fonction de répartition de la loi normale centrée réduite :

Tableau I-II : Probabilité et t_i pour 4 valeurs de la distribution étudiée

x_i	$P(X < x_i) = \Phi(t_i)$	t_i
4	0,10	-1,28
8	0,30	-0,525
12	0,60	0,255
16	0,80	0,84

On trace alors les points de coordonnées $(x_i; t_i)$:

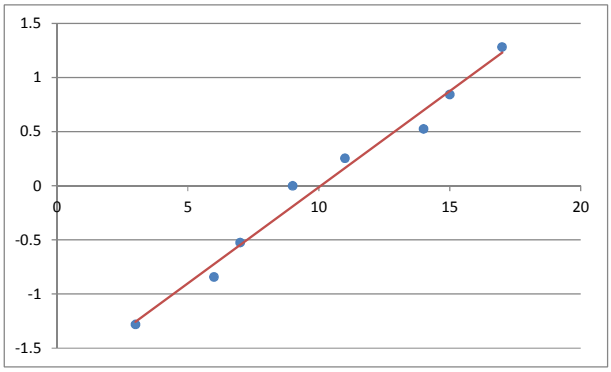


Tableau I-III : Droite de Henry de la distribution étudiée