INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche LILLE - NORD EUROPE

Université Lille1 Sciences et Technologies

cnrs dépasser les frontières

UNIVERSITÉ LILLE 1

Numéro d'ordre : 40441

Passage à l'échelle des intergiciels RFID pour l'informatique diffuse

Thèse de doctorat (spécialité Informatique) présentée le 06 décembre 2010

par LOÏC SCHMIDT

Composition du jury

Rapporteurs :	Michel Banâtre, Directeur de recherche, INRIA Rennes - Bretagne Atlantique
	Marcelo Dias de Amorim, Chargé de recherche HdR., LIP6/CNRS - Université de Paris 6
Examinateurs :	Stéphane Ducasse, Directeur de recherche, INRIA Lille - Nord Europe
	Hakima Chaouchi, Maître de Conférences, Telecom Sud Paris
	Patrice Ribout, Responsable de projet RFID pour le groupe Oxylane, Lille
Directeur :	David Simplot-Ryl, Professeur, Université Lille 1
Encadrant :	Nathalie Mitton, Chargée de recherche, INRIA Lille - Nord Europe

À Christopher Llyod et Robert Zemeckis ...

Table des matières

1 Introduction

Depuis plusieurs années, beaucoup imaginent une nouvelle ère à Internet, celle de l'Internet des objets où chaque objet manufacturé se voit attribuer un identifiant unique. Cette identifiant ne se présente plus sous une forme graphique (comme le code à barres par exemple) mais est électronique et capable de communiquer sans fil avec des lecteurs appropriés. Ainsi, il est possible de lire plusieurs identifiants à la fois et sous certaines conditions de calculer le prix d'un caddie entier en quelques secondes ou de faire un inventaire d'un rayon en quelques minutes. Cette technologie est déjà très utilisée dans les systèmes antivol. Cet identifiant se présente sous forme d'une étiquette électronique appelée étiquette RFID pour *Radio Frequency IDentification*. L'arrivée de ces RFID dans le processus de traçabilité et de gestion des biens soulèvent de nouvelles problématiques techniques telles que leur intégration dans l'Internet qui permettra d'accéder aux informations relatives à l'objet identifié, recueillies durant sa vie. Pour ce faire, certaines entités (des serveurs pour la gestion de stocks des entreprises par exemple) ont pour charge la récupération et le stockage des données. Ces entités vont ensuite donner accès à ces informations à l'ensemble du réseau par l'utilisation d'un intergiciel spécifique. Un intergiciel, comme son nom l'indique, et un logiciel intermédiaire entre le réseau et le matériel d'une part, et les applications d'autre part. Cet intergiciel permet de collecter, de filtrer et d'agréger les données. Cela permet aussi une gestion des différents lecteurs plus facile. L'expansion du nombre de RFID ainsi que la quantité de données qu'ils génèrent amènent à s'assurer de la capacité de l'architecture logicielle de l'intergiciel à gèrer cet afflux d'information. Nos travaux ont porté sur le passage à l'échelle des intergiciels RFID. Dans la prochaine section, nous présentons la technologie RFID. Nous exposons ensuite l'intérêt d'un intergiciel pour la gestion des étiquettes électroniques.

1.1 La technologie RFID

L'identification par fréquence radio (Radio Frequency IDentification - RFID) est une méthode pour identifier un objet grâce à une étiquette électronique et un lecteur. L'étiquette électronique, composée d'une puce électronique et d'une antenne permettant, par radio, de recevoir des requêtes d'un lecteur et de lui répondre. Ces étiquettes peuvent être collées sur des objets afin de les identifier. Elles peuvent même être implantées dans des animaux ou des humains. Si l'on se sert encore de code à barres pour la traçabilité, les étiquettes électroniques sont déjà utilisées dans des solutions antivol. La figure 1.1 montre des étiquettes électroniques. On peut voir sur chacune d'elle la puce électronique (en foncé), et les antennes, qui font le contour des étiquettes.

Ces étiquettes vont communiquer avec un lecteur. Ce dernier va initialiser l'échange en envoyant une requête à l'étiquette. Celle-ci va répondre par radio. C'est l'avantage de la RFID sur le code à barres, il n'y a pas besoin de voir l'étiquette pour récupérer les informations. On peut ainsi en inventorier plusieurs en quelques secondes (jusqu'à 250 étiquettes par seconde [1]). La réponse de l'étiquette au lecteur est bien souvent son identifiant. Cet identifiant sert plus à identifier le « propriétaire » de l'étiquette que l'étiquette en elle-même. Ce « propriétaire » est

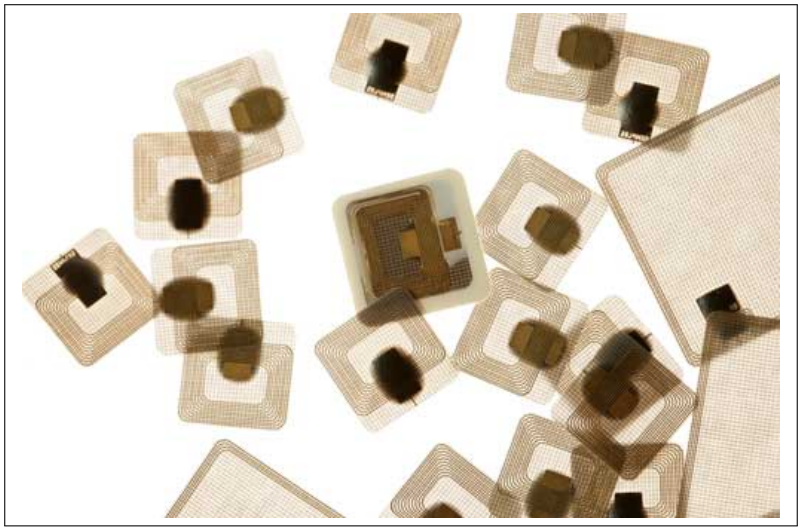


FIGURE 1.1 – *Des étiquettes électroniques*

l'objet sur lequel l'étiquette est apposée, ou l'usager de la carte de transport qui vient d'être lue, *etc.*

Il existe trois types d'étiquettes RFID :

- Les étiquettes dites passives : elles n'ont pas de batterie et ne peuvent transmettre qu'en utilisant l'énergie transmise par le lecteur. En effet, l'onde électromagnétique de la requête va créer une induction dans l'antenne qui tiendra lieu de bobine et va alimenter le circuit, permettant le traitement de la requête et l'envoi de la réponse ;
- Les étiquettes dites actives : elles ont une batterie et s'en servent pour alimenter la communication. Cette batterie peut aussi servir à alimenter des capteurs comme un thermomètre ou un capteur de lumière ;
- Les étiquettes dites semi-passives : elles ont une batterie mais ne s'en servent que pour alimenter les capteurs. La communication est alimentée par la requête du lecteur, comme pour les étiquettes passives.

Avec la RFID et la possibilité d'équiper d'un identifiant les objets, un nouveau concept est né, l'Internet des Objets. L'Internet des Objets [2] est : *« un réseau de réseaux qui permet, via des systèmes d'identification électronique normalisés et unifiés, et des dispositifs mobiles sans fil, d'identifier directement et sans ambiguïté des entités numériques et des objets physiques et ainsi de pouvoir récupérer, stocker, transférer et traiter, sans discontinuité entre les mondes physiques et virtuels, les données s'y rattachant »* [1]. Au delà de l'étiquette et de son identifiant, l'Internet des Objets est aussi une architecture réseau permettant d'enregistrer et de récupérer les données relative à cet identifiant, permettant ainsi la traçabilité de l'objet attaché à cette étiquette. C'est ce que l'on appelle un intergiciel.

1.2 Pourquoi un intergiciel ?

Un intergiciel est un logiciel intermédiaire entre le réseau et les applications, permettant la communication entre des applications hétérogènes. Le but d'un intergiciel est d'accomplir les tâches techniques pour les applications métiers et les échanges de données. Dans les système RFID, un tel intergiciel doit gérer les lecteurs, souvent hétérogènes, doit traiter les évènements issus des lecteurs de RFID, et doit être connecté aux applications métiers. Dans certains cas, il n'y a pas

1. *Etude « l'internet des objets »* par Pierre-Jean Benghozi et Sylvain Bureau (Pôle de recherche en Economie et Gestion de l'Ecole Polytechnique) et Françoise Massit-Folléa (programme Vox Internet II)

besoin d'intergiciel, comme dans le cas d'une petite et unique application telle que « compter le nombre d'identifiants lus », ou encore « lire les étiquettes dans le champs du lecteur »comme dans l'exemple de la figure 1.2. Ici l'application n'a qu'un seul lecteur et ne fait que lire les étiquettes passant sous le portail.

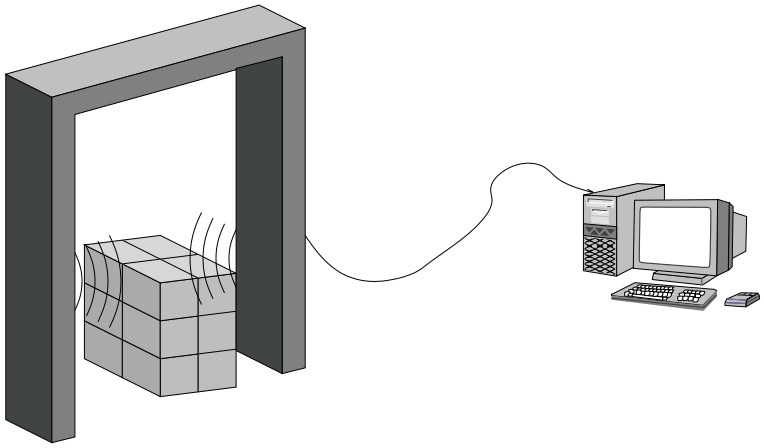


FIGURE 1.2 – *Exemple d'application RFID*

Le besoin d'intergiciel intervient si l'on complexifie légèrement cette architecture en y ajoutant des lecteurs, des applications nécessitant l'accès aux données recueillies par ces lecteurs. En effet, partager les informations arrivant des lecteurs entre plusieurs applications métiers nécessite de spécifier les messages des lecteurs en fonction des applications. Une application peut n'avoir besoin que des identifiants des produits passant sous le portail, tandis que d'autres vont requérir des informations plus larges. Par exemple, une application d'un transporteur ne fait pas juste lister les identifiants (ID), il a aussi besoin de savoir lequel de ces ID est celui de la palette et lesquels sont ceux des produits, afin de faire une association contenant-contenu. Dans la plupart des cas, les détails physiques ne sont pas utiles pour la logique métier. Par conséquent, un intergiciel devient obligatoire quand les applications de RFID deviennent complexes. C'est un point d'accès unique aux données collectées. Il associe la lecture d'objet avec l'activité et/ou la localisation. Il homogénéise l'accès aux données au travers d'interfaces standards permettant :

- La gestion des lecteurs (séparation ente les lecteurs physiques et logiques) ;
- La gestion des données (formatage, agrégation, filtrage) ;
- Les appels à des fonctions des applications ou l'envoi de résultat ;
- La gestion de règles et d'exceptions ;
- La collecte de données et le traitement en mode temps-réel ou en asynchrone.

Un intergiciel RFID est considéré comme un composant ajoutant de l'intelligence essentielle à tous systèmes RFID et peut être lié avec d'autres systèmes d'information de l'entreprise tels que les bases de données externes, les systèmes d'information de partenaires ou les systèmes de gestion d'un hangar, *etc.*. La figure 1.3 montre la position d'un intergiciel RFID ainsi que deux concepts clés d'un point de vue extérieur : l'interface avec les lecteurs et l'interface avec les applications. En effet, l'intergiciel est le lien entre les lecteurs et les applications métiers. Mais connecter les lecteurs directement sur les applications métiers soulèvent quelques problèmes :

- Les applications sont dépendantes des lecteurs. Cela veut dire que si l'on doit changer un ou des lecteurs et que ces nouveaux lecteurs n'utilisent pas les mêmes protocoles de communications que les précédents, alors une partie du code des applications connectées doit subir des modifications.
- Toutes les applications connectées aux lecteurs reçoivent la même information sur les étiquettes. De plus, cette information est brute (c'est une liste d'identifiants). Néanmoins,

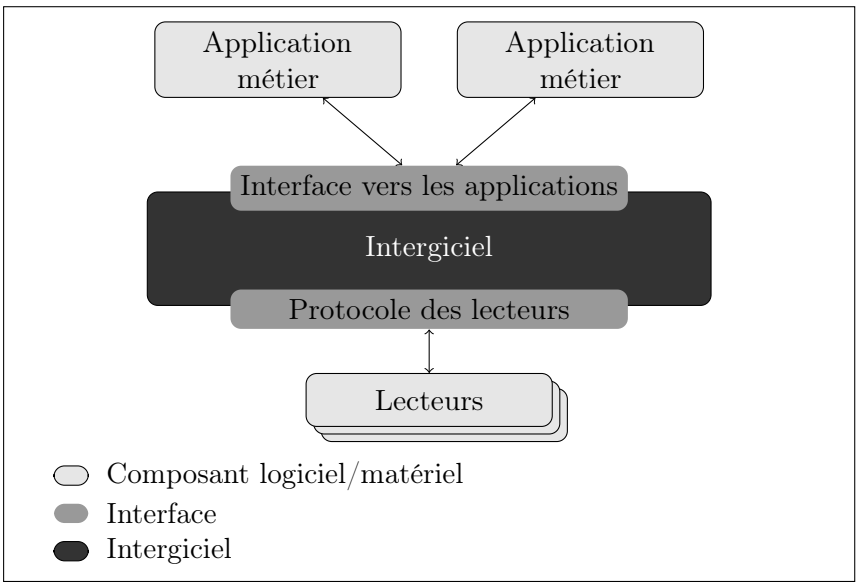


FIGURE 1.3 – *Un intergiciel RFID d'un point de vue extérieur*

seule une partie de cette information peut être nécessaire, et cela dépend des applications. Les applications sont inondées par trop de données.
- Enfin, si il y a un grand nombre de lecteurs et/ou d'applications, le système est amené à tomber en panne à cause de la saturation et de la surcharge de travail.

Sans intergiciel, connecter directement les lecteurs aux applications ne passe pas à l'échelle et consomme inutilement des ressources.

L'intergiciel va donc abstraire les lecteurs pour faciliter leur interrogation par les applications. Pour cela, les applications doivent savoir comment interroger l'intergiciel. Et ce dernier doit savoir comment interroger les lecteurs. Afin de faciliter le déploiement de la technologie RFID, de nombreux acteurs se sont réunis et ont fondé le Auto-ID Center en 1999. Après avoir défini une série de standards pour les intergiciels RFID, EPCGlobal [3] et l'Auto-ID Labs [4] ont émergé de ce centre avec chacun un rôle particulier. Le premier est de gérer le réseau et les standards ainsi créés. Quant au second, il continue d'étudier la technologie RFID afin de l'améliorer. EPCGlobal étant maintenant une partie de GS1, gérant des codes à barres, ces standards se démocratisent et les intergiciels sont donc pour la plupart compatibles avec ces standards. En effet, les besoins en traçabilité des entreprises dépassent le cadre de leurs hangars seuls, et la coopération et l'échange entre entreprises nécessitent la définition d'un langage commun pour offrir une inter-opérabilité. C'est le rôle d'un standard. Or, déjà aujourd'hui, les standards GS1 sur les codes à barres sont largement utilisés pour l'identification des produits. L'avenir des standards d'EPCGlobal, définissant l'architecture d'un intergiciel complet permettant l'échange d'informations entre les entreprises et utilisant la technologie EPC, présentent donc un intérêt grandissant dans le secteur industriel.

1.3 Organisation du document

Le passage à l'échelle d'intergiciels RFID se conformant aux standards d'EPCGlobal constitue le cœur de nos travaux. Ce sont les seuls standards allant de l'identification au partage des informations issues des identifiants RFID. De plus, certains travaux sont effectués dans le cadre du projet européen ASPIRE dont le but est de fournir un intergiciel conforme aux standards d'EPCGlobal et open source, et d'autres dans le cadre des projets nationaux ICOM et WINGS ayant GS1 dans le consortium. Nous commençons dans le chapitre 2 par présenter l'architecture

d'un réseau EPCGlobal ainsi que les solutions intergicielles existantes. Nous verrons les concepts clés à l'intérieur de la boite noire représentée sur la figure 1.3 et comment ils sont utilisés pour répondre aux fonctions de l'intergiciel RFID. Nous considérons ensuite séparément certains composants de cette architecture pour en dégager les modifications à y apporter pour un passage à l'échelle efficace.

Nos travaux s'articulent autour de trois axes que nous présentons dans les trois chapitres suivants :

Chapitre 3 : A partir du standard d'EPCGlobal nous avons conçu un composant de filtrage et d'agrégation des données reçues des lecteurs passant à l'échelle (standard ALE d'EPCGlobal). Ce passage à l'échelle est permis par l'utilisation des tables de hachage distribuées, technique déjà utilisée dans les logiciels de partage de fichiers. Nous montrons que cette solution, permettant le passage à l'échelle tout en assurant la transparence aux couches supérieures à l'intergiciel, entraine un surcoût acceptable pour de meilleures performances.

Chapitre 4 : Nous proposons un moyen efficace de gestion des bases de données (standard EPCIS d'EPCGlobal) de manière distribuée pour des applications de gestion de stocks, cette solution étant basée sur des tables de hachage distribuée. Cette solution offre également une gestion efficace des requêtes de recherche dans cette base distribuée. Son niveau de tolérance aux pannes est paramétrable grâce à un contrôle du taux de réplication des données enregistrées.

Chapitre 5 : Nous présentons une distribution du serveur fournissant le service de nommage des objets (le standard ONS décrit ce service).

Le chapitre 6 est consacré aux travaux d'intégration d'autres systèmes d'identification standardisé au sein de l'architecture EPCGlobal. Nous revenons également sur un composant de filtrage et d'agrégation de données développé afin d'être embarqué dans un lecteur mobile aux capacités limitées. Le dernier chapitre apporte une vue générale des travaux présentés.

État de l'art

2

Afin de rendre les interigiciels RFID inter-opérables, il faut se mettre d'accord sur les interfaces et les composants de cet intergiciel. Il existe actuellement des standards définissant le rôle de plusieurs composants pour des intergiciels RFID et décrivant les interfaces de connexions entre ces composants. Ces standards, développés par EPCGlobal [1], permettent de d'interconnecter les différents composants d'un intergiciel RFID afin de collecter, de filtrer, d'agréger, d'enregistrer, de consulter et d'échanger les données d'étiquettes RFID. Cette architecture d'intergiciel est décrite dans le standard Architecture Framework d'EPCGlobal [5].

Ce chapitre est organisé comme suit. Nous allons dans un premier temps, dans la section 2.1, présenter une vue d'ensemble d'un intergiciel RFID, les différents besoins auxquels il répond, ainsi que les intergiciels et standards d'intergiciel existants (principalement ceux d'EPCGlobal). Dans un second temps, nous entrerons en détails sur le fonctionnement de certains standards d'EPCGlobal. Nous ne présentons ici que les standards auxquels nous nous sommes intéressés durant ces travaux. La section 2.2 se concentre sur les standards de structure et traduction des données des étiquettes : Tag Data Standard (TDS)[6] et Tag Data Translation (TDT)[7]. Nous expliquerons ensuite dans la section 2.3 le fonctionnement du standard Application Level Event (ALE)[8], définissant une interface pour le composant qui va filtrer et agréger les données reçues des lecteurs. Enfin, dans la section 2.4, le standard de l'EPC Information Services (EPCIS)[9], chargé de la persistance des données, sera présenté. Nous finirons par le standard de l'Object Name Service (ONS)[10]. Nous verrons comment ce dernier standard permet l'échange des données à travers les entreprises.

2.1 Les intergiciels RFID et EPCGlobal

2.1.1 EPCGlobal

Comme nous l'avons vu dans le chapitre précédent, la connexion directe entre les lecteurs et les applications n'est pas une solution efficace. En revanche, l'utilisation d'un intergiciel entre les deux afin de gérer cette connexion répond à ces problèmes. Pour ce faire, l'intergiciel est divisé en plusieurs blocs ayant chacun un rôle particulier. La figure 2.1 montre l'intérieur de notre boite noire de la figure 1.3. Les deux composants logiciels « Filtrage et agrégation » (F&A) et « Gestion des lecteurs » répondent aux problèmes ci-dessus. En effet, le composant de gestion des lecteurs ainsi que l'interface du protocole des lecteurs permettent de résoudre le problème d'hétérogénéité et de contrôle des lecteurs connectés. Quant au composant F&A, il offre la spécialisation des données (filtrage et agrégation), et permet ainsi de n'envoyer sur le réseau que les données utiles aux applications. En effet, les informations sur les étiquettes ne sont envoyées qu'aux application en ayant fait la demande par l'interface. Donc, plutôt que d'envoyer les informations brutes,

1. http ://www.epcglobalinc.org

elles sont filtrées et agrégées par le composant F&A avant d'être envoyées vers les applications concernées. C'est ainsi que les applications reçoivent des données directement exploitables.

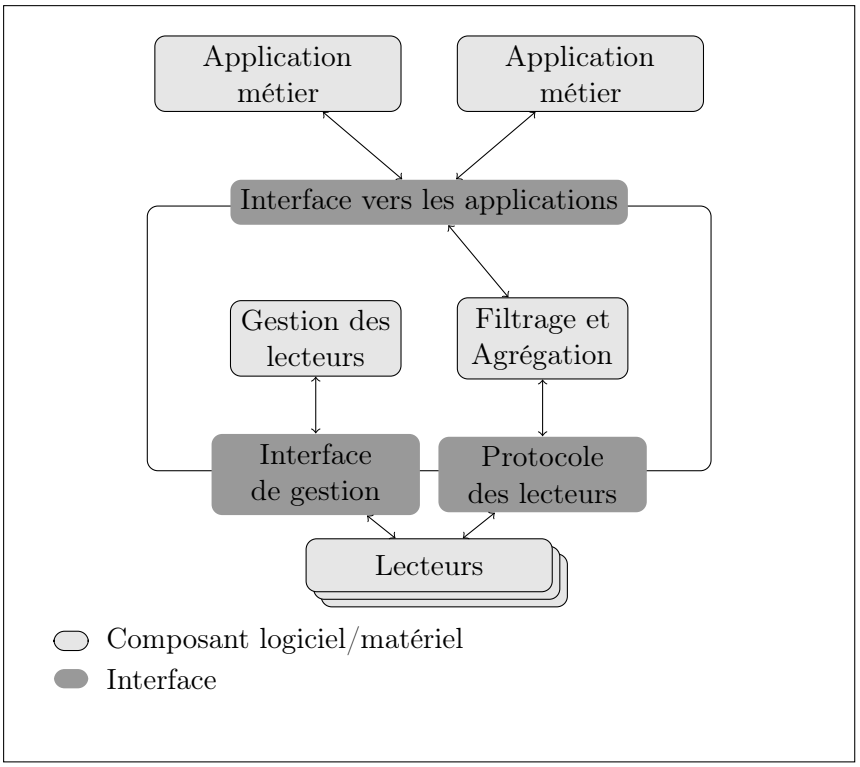


FIGURE 2.1 – *À l'intérieur de l'intergiciel RFID*

Nous avons souligné ici l'importance des deux composants de gestion des lecteurs et de filtrage et d'agrégation, mais ce sont les deux interfaces qui sont les points importants. En effet, le composant de gestion des lecteurs permet de gérer la configuration de ces derniers, mais sans les interfaces de gestion et de protocole, un développement est nécessaire à chaque ajout/modification de lecteur. Il en est de même pour l'interface des applications.

Afin de permettre l'échange des lecteurs et une connexion facilitée des applications métiers, ces interfaces doivent être standardisées. En effet, si toutes les applications se connectaient à l'intergiciel de la même façon, un gain de développement serait évident. C'est pour cela qu'EPCGlobal [3] propose toute une série de standards. La particularité des standards EPCGlobal est qu'ils définissent aussi bien la façon dont les données sont enregistrées dans l'étiquette RFID que la façon dont on peut retrouver des données relatives à ces étiquettes. Pour cela, EPCGlobal définit la structure même de l'identifiant contenu dans les étiquettes, appelé EPC (pour Electronic Product Code), avec ses standards Tag Data (TDS [6]) et Tag Data Translation (TDT [7]). Ces deux standards sont expliqués en détails dans la section 2.2. La figure 2.2 dépeint l'architecture logicielle du réseau EPCGlobal [5]. Nous y retrouvons tout en bas les interfaces des lecteurs (Interfaces de protocole et de gestion des lecteurs [11][12][13]) et des applications (Interface ALE [8]), ainsi que nos composants de filtrage et d'agrégation et de gestion des lecteurs. Plus de détails sont données sur l'interface ALE en section 2.3.

Il manque plusieurs aspects importants dans notre intergiciel de la figure 2.1 et que l'on retrouve dans les standards d'EPCGlobal, car il ne s'agit pas juste de récupérer des données des identifiants lus par des lecteurs. Ces données, afin de les rendre les plus utiles possible, doivent être enregistrées dans une base de données et doivent pouvoir être partagées. C'est le rôle des standards EPCIS [9] et ONS [10] d'EPCGlobal. Le premier définit deux interfaces permettant (i) de récupérer les données envoyées depuis l'interface ALE et (ii) d'interroger une base données

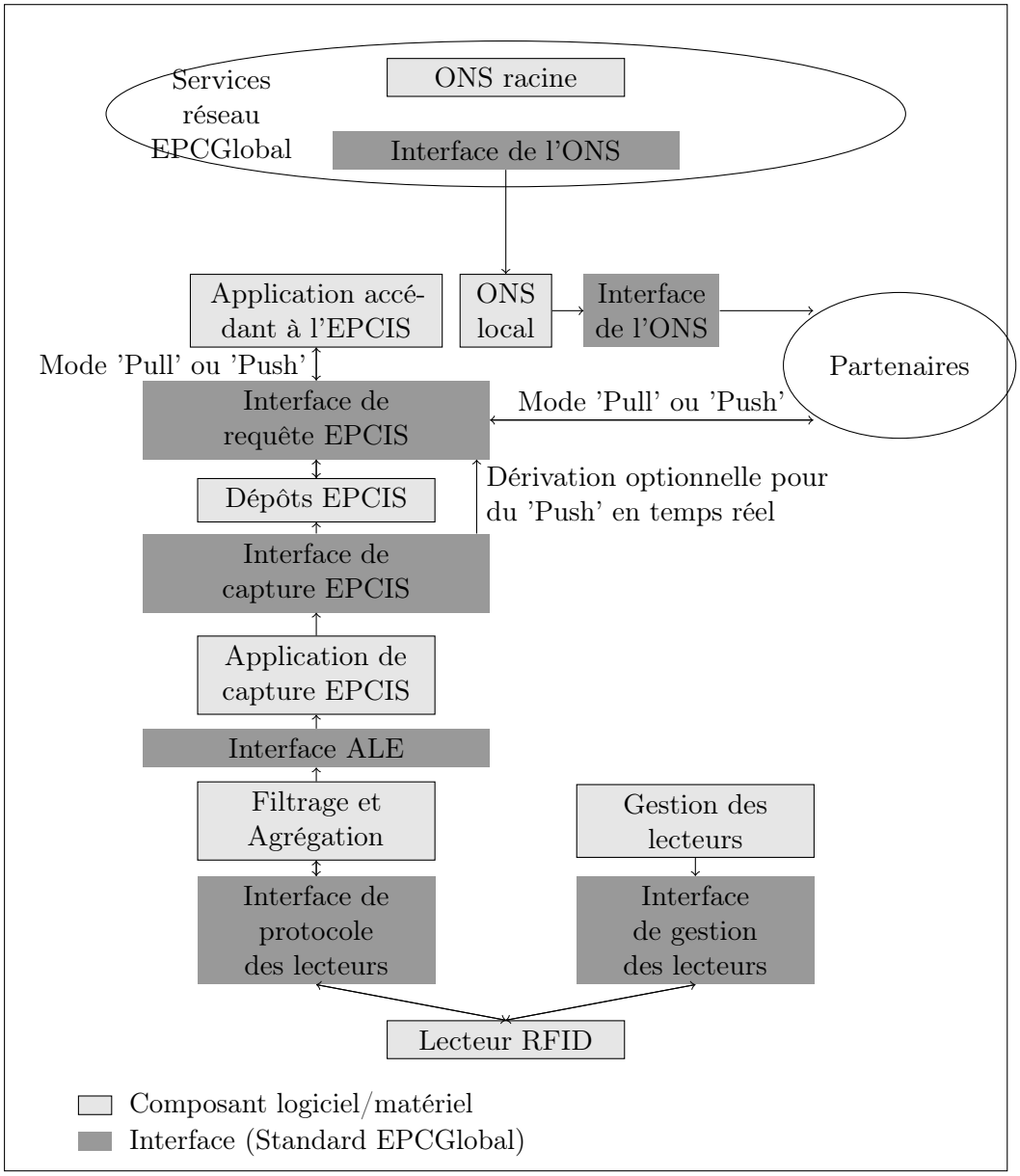


FIGURE 2.2 – *Architecture réseau d'EPCGlobal*

ayant enregistré les données récupérées. C'est la base de données de l'intergiciel. Ce standard est détaillé dans la section 2.4. Une fois ces données stockées et une possibilité de les récupérer offerte, il faut un moyen de retrouver où sont enregistrées les données, afin de ne pas questionner toutes les bases de données avant de trouver celle qui contient les informations que l'on souhaite. L'ONS, basé sur le fonctionnement des DNS, associe un identifiant EPC à une série de services, notamment celui des EPCIS. La section 2.5 présente le fonctionnement de ce standard.

Avant d'entrer dans les détails de plusieurs de ces standards, il est intéressant de connaître les différents intergiciels existants, qu'ils soient propriétaires ou libres.

2.1.2 Les intergiciels existants

Il existe de nombreux intergiciels disponibles sous différentes licences. Parmi les solutions open-source on peut citer : *AspireRFID*, *Rifidi*, *CUHK RFID System* 1.0,*LogicAlloy*, et *LLRP Toolkit*. *AspireRFID* [14] offre un intergiciel compatible avec les standards EPCGlobal et développé dans

le carde du projet FP7 Aspire. Il offre aussi une série d'outils facilitant le développement, le déploiement et la gestion des applications pour la RFID et les capteurs et permettant ainsi aux consultants en RFID de déployer des solutions sans avoir besoin de programmer à bas niveau. Les outils génèrent tous les fichiers nécessaires au déploiement de ces solutions en utilisant l'intergiciel AspireRFID. Il implémente plusieurs spécifications de consortiums tels que EPCGlobal, NFC Forum, JCP et OSGi Alliance. Le logiciel Fosstrak [15] est composé de quatre briques implémentant des standards EPCGlobal. On y retrouve un composant de filtrage et d'agrégation fournissant une interface ALE. Le moteur de traduction des identifiants TDT est implémenté aussi, ainsi que le dépôt EPCIS et un outil de gestion des les lecteurs LLRP. *Rifidi* [16] est en fait deux applications outre un intergiciel compatible EPCGlobal. L'une d'elles, Rifidi Emulator un simulateur de matériel. Il simule un ou plusieurs lecteurs utilisant le standard LLRP d'EPCGlobal. Rifidi Designer est un outil visuel de conception et de simulation de processus RFID en 3D. Le dernier, Rifidi Tag Streamer permet de générer des lecteurs et des étiquettes RFID virtuels afin de tester un système RFID. *CUHK RFID System* 1.0 [17] implémente le standard ALE d'EPCGlobal. Il offre une implémentation du composant de filtrage et d'agrégation fournissant une interface ALE aux applications. Il permet la connexion de lecteur via IP ou RS-232 et offre une console de gestion permettant de configurer, de contrôler et de surveiller les lecteurs. *LogicAlloy* [18] est un serveur ALE compatible EPCGlobal. Il est fournit avec un simulateur d'évènements permettant de tester le système RFID et supporte une demi-douzaine de protocoles de lecteur. Le serveur ALE permet d'envoyer le rapport avec différents protocoles (HTTP, FTP, SOAP, *etc.*). *LLRP Toolkit* [19] fournit une librairie permettant de faciliter le développement d'application utilisant LLRP et l'intégration de lecteur LLRP dans des systèmes existants. Implémentant le standard LLRP, cette librairie est utilisée dans beaucoup de projets tels que Fosstrak (dans son outil de gestion des lecteurs LLRP) ou encore les fabricants Intermec et Impinj. Les développeurs de Rifidi sont impliqués dans le développement de la librairie LLRP Toolkit. Elle est disponible dans plusieurs langages de programmation (Perl, C, C++, Java, .Net, *etc.*).

Le développement rapide de la RFID et le potentiel économique qui y est associé a attiré de grands acteurs de l'informatique tels que IBM, Oracle, Microsoft, SAP, Sun Microsystems, et HP qui proposent chacun leur solution propriétaire. Il en existe d'autres dont le développement est abandonné, ou a fusionné avec d'autres projets comme UJF RFID Suite a fusionné avec AspireRFID. La plupart des solutions existantes utilisent les standards EPCGlobal dont nous présentons les quatre utilisés durant nos travaux.

2.2 Standard TDT d'EPCGlobal (Tag Data Translation)

Au cœur de l'architecture EPCGlobal, il y a l'identifiant, ou Electronic Product Code (EPC). Tous les standards d'EPCGlobal sont définis afin de gérer, de manipuler ces identifiants. Il convient donc de commencer par les standards des données des étiquettes (Tag Data Standard - TDS) [6] et de la traduction vers d'autres formats de cet identifiant (Tag Data Translation - TDT) [7].

Tag Data Standard (TDS)

La structure d'un EPC est importante afin de comprendre en quoi il est unique. En effet, un des fondements de l'architecture d'EPCGlobal est cette unicité des identifiants, permettant de différencier de façon certaine un objet d'un autre. Basé sur le système d'identification de GS1 utilisé dans les codes à barres [20], le standard TDS définit différents types d'EPC (tels que le

SGTIN [2], SSCC [3], SGLN [4], GID [5], *etc.*). Tous ces identifiants ont des rôles particuliers (*e.g.* SGTIN pour les biens à vendre, SSCC pour les palettes, SGLN pour les localisations, *etc.*).

La structure de ces identifiants suit les mêmes règles pour tous les identifiants (sauf le GID et le DoD [6], mais ce dernier est réservé au département de la défense des États-Unis d'Amérique). La figure 2.3 représente la structure d'un SGTIN de longueur 96 bits. Il est composé de six champs.

Entête	Filtre	Partition	Préfixe de compagnie	Référence d'objet	Numéro de série
8 bits	3 bits	3 bits	20-40 bits	24-4 bits	38 bits

FIGURE 2.3 – *Structure d'un SGTIN*

- Entête : identifie le type d'EPC auquel nous avons affaire (SGTIN, DoD, SSCC, *etc.*) ;
- Filtre : indique une valeur utilisée pour du filtrage rapide ;
- Partition : indique la longueur du préfixe de compagnie et de la référence de l'objet ;
- Préfixe de compagnie : identifie le fabriquant du produit (préfixe donné par GS1) ;
- Référence d'objet : identifie la classe de l'objet ;
- Numéro de série : identifie l'objet en lui-même dans sa classe.

Le préfixe de compagnie et la référence de l'objet forme ce que l'on appelle le GTIN, utilisé par GS1 dans ses codes à barres. L'avantage des identifiants SGTIN est le numéro de série, permettant d'identifier uniquement l'objet en question, contrairement aux codes à barres, munis pour la plupart d'un GTIN, et n'identifiant ainsi qu'une classe d'objet (*i.e.* bouteille d'eau d'un litre et demi de telle marque, mais pas une bouteille en particulier). Dans le préfixe de la compagnie, trois chiffres dépendent du pays de l'entreprise. Par exemple, en France, ces trois digits forment un numéro compris entre 300 et 379 [7]. Ces trois chiffres ne permettent pas de savoir dans quel pays est fabriqué le produit, mais plutôt le pays de l'entreprise ou dans lequel l'entreprise s'est enregistré chez GS1.

Tag Data Translation (TDT)

Chacun de ces identifiants peut être écrit sous différents formats, ayant tous une utilité à un moment dans l'architecture d'EPCGlobal. La figure **??** donne un exemple d'un SGTIN dans les différents formats définis dans le standard TDT. En effet, ce standard défini comment on passe d'une représentation à une autre.

Representation	**Value**
TAG-ENCODING URI	urn :epc :tag :sgtin-64 :1.0037000.030241.1041970
PURE-IDENTITY URI	urn :epc :id :sgtin :0037000.030241.1041970
ONS HOSTNAME	030241.0037000.sgtin.onsepc.com
LEGACY	gtin=00037000302414 ;serial=1041970
BINARY	10001000000000000010000011101100010000010000011111111001100011001 0

FIGURE 2.4 – *EPCglobal Tag Data representation*

Le format *tag-encoding* d'un EPC est surtout utilisé au niveau de l'ALE (Application Level

2. Serialized Global Trade Item Number
3. Serial Shipping Container Code
4. Serialized Global Location Number
5. General Identifier
6. Department of Defense
7. http ://www.gs1.org/barcodes/support/prefix_list

Event) pour définir des filtres et/ou des groupes. La représentation *pure-identity* peut servir dans l'ALE et dans l'EPCIS. L'EPC doit être sous sa forme *ons-hostname* afin de procéder aux requêtes d'ONS. La forme binaire de l'EPC sert à la communication avec le lecteur, pour les écritures ou les lectures d'étiquettes RFID. Enfin, le format *legacy* est défini afin d'avoir une version plus rétro-compatible pour les applications métiers. En effet, dans ce format, on y retrouve le GTIN, identifiant utilisé dans les codes à barres actuellement encore très répandus. Cette étape de traduction peut donc avoir lieu à n'importe quel niveau de l'architecture, la TDT est un outils très important dans le réseau EPCGlobal.

On ne passe pas d'un format à l'autre comme ça. Il y des pertes d'information entre les différentes représentation d'un EPC. Ainsi, par exemple, le format *ons-hostname* n'inclut pas le numéro de série. On ne peut donc pas traduire un EPC de sa forme *ons-hostname* vers du *tag-encoding* sans informations supplémentaires. La figure 2.5 montre les paramètres additionnels nécessaires lors de l'encodage d'un EPC.

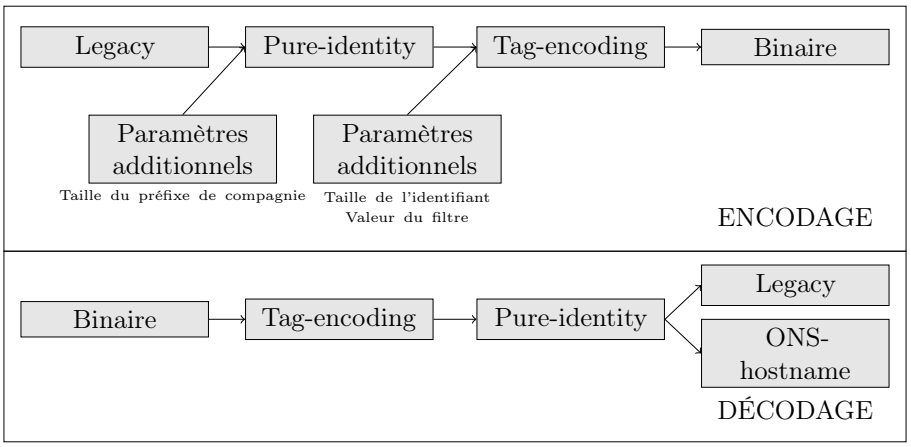


FIGURE 2.5 – *Décodage et encodage d'un EPC d'un format à un autre.*

Les formats binaire et *tag-encoding* contiennent toutes les informations d'un EPC. Il n'y a donc pas besoin de paramètres pour traduire un EPC à partir d'un de ces deux formats, quelque soit le format de sortie. En revanche, pour l'encodage, des paramètres supplémentaires sont requis, tels que la taille du code EPC, le filtre (ces deux informations ne sont pas présentes dans les formats *ons-hostname*, *pure-identity* et *legacy*). La longueur du préfixe de compagnie est indispensable pour la traduction à partir du format *legacy*.

Ainsi, ces deux standards définissent les différents types d'EPC pour différents usages, la façon dont sont structurés ces EPC, et la façon dont ils vont être restructurés, formatés afin d'être sous une représentation plus efficace en fonction de la couche de l'intergiciel dans laquelle ils sont traités.

2.3 Standard ALE d'EPCGlobal (Application Level Event)

Le standard ALE d'EPCGlobal (Application Level Event)[8] définit les interfaces fournies par le composant de filtrage et d'agrégation (F&A) aux couches supérieures de l'architecture (EPCIS, applications métiers, *etc.*, voir figure 2.2). Ces interfaces sont au nombre de cinq :

interface de lecture : permet de configurer le composant F&A afin d'executer une opération de lecture sur diverses sources ;

interface d'écriture : offre la possibilité d'écrire dans les étiquettes RFID via les lecteurs ou les imprimantes ;

interface de lecteur logique : sert à définir des lecteurs logiques, utilisables pour les opérations de lectures et d'écritures, regroupant une ou plusieurs sources physiques de lecture/écriture ;

interface de spécification de la mémoire des étiquettes : permet de définir des noms symboliques associés aux champs de données des étiquettes ;

interface de contrôle d'accès : contrôle l'accès des clients aux autres interfaces.

Afin d'utiliser ce composant, les couches supérieures vont définir des spécifications permettant de configurer le composant. Ces spécifications sont propres à l'interface utilisée. Ainsi, il existe des spécifications de lecture, d'écriture, de configuration de lecteurs logiques, *etc.*

Interface de lecture

L'interface de lecture fournit des méthodes pour gérer les spécifications d'opérations de lecture. Ces spécifications, appelée ECSpecs, définissent les lecteurs logiques inclus dans l'opération, le type d'identifiant à filtrer et à agréger, *etc.*. Le listing 2.1 est un exemple de fichier XML d'ECSpecs : il définit ce que le F&A doit faire pour générer le rapport. Les lignes 3 à 6 définissent les lecteurs logiques nécessaires à l'opération. La partie *boundarySpec* (lignes 7 à 11) configure le composant pour exécuter cette opération toutes les $10000ms$, et cette opération dure $9500ms$. La dernière partie, lignes 12 à 20, concerne ce qu'il faut que le rapport contienne. La ligne 14 indique qu'il faut remonter les identifiants des étiquettes présentes dans les champs des lecteurs (« CURRENT »). On peut aussi avoir « ADDITIONS » ou « DELETIONS », pour remonter respectivement les identifiants des étiquettes arrivant dans le champ du lecteur ou quittant ce champ durant la période de lecture ou depuis le dernier rapport. Les lignes 15 à 17 représentent les patrons de groupement des identifiants. Ici, les « X » dans les champs de filtre et de préfixe de compagnie et les « * » pour le numéro de la classe de l'objet et son numéro de série indiquent que l'on veut grouper les identifiants par filtre et compagnie identique, mais objets différents. Par conséquent, ce patron groupe les identifiants par fabriquant. Enfin la ligne 18 sert à spécifier le(s) format(s) sous le(s)quel(s) les identifiants doivent apparaître dans le rapport (ici, sous leur forme *tag-encoding*). En d'autres termes, cette spécification décrit une opération d'inventaire avec deux lecteurs logiques, qui se répète toutes les 10 secondes pour une durée de 9,5 secondes en groupant les produits par fabriquant.

Listing 2.1 – *Exemple d'ECSpecs*

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <ns2:ECSpec xmlns:ns2="urn:epcglobal:ale:xsd:1">
3     <logicalReaders>
4        <logicalReader>LogicalReader1</logicalReader>                    } Lecteurs
5        <logicalReader>LogicalReader2</logicalReader>                    }
6     </logicalReaders>
7     <boundarySpec>
8        <repeatPeriod unit="MS">10000</repeatPeriod>                     }
9        <duration unit="MS">9500</duration>                              } Période
10       <stableSetInterval unit="MS">0</stableSetInterval>               }
11    </boundarySpec>
12    <reportSpecs>
13       <reportSpec>
14          <reportSet set="CURRENT"/>                                    } ID à remonter
15          <groupSpec>
16             <pattern>urn:epc:pat:sgtin-96:X.X.*.*</pattern>            } Groupe
17          </groupSpec>                                                  }
18          <output includeTag="true"/>                                   } Format de sortie
19       </reportSpec>
20    </reportSpecs>
21 </ns2:ECSpec>
```

Un fois que la spécification est envoyée au F&A puis validée, l'intergiciel va effectuer l'opération et envoyer le rapport (appelé ECReports). Le listing 2.2 est un exemple d'un fichier XML décrivant un ECReports qui pourrait être une réponse pour l'ECSpecs 2.1. Il indique à quelle spécification il correspond (ligne 2), puis il liste les identifiants des étiquettes présentes dans le champ des deux lecteurs (lignes 10, 13, 16 et 23). Ces identifiants sont regroupés en fonction de leur fabriquant. Ici, les trois premiers objets remontés proviennent tous du fabriquant dont le préfixe de compagnie est 211298, et le dernier de la compagnie 211265. L'attribut « name » des éléments « group » représente le patron que vérifient les identifiants du groupe.

Listing 2.2 – *Exemple d'ECReports*

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <ns2:ECReports totalMilliseconds="32024" specName="specCURRENT"          } Spécification
3                 date="2010-05-28T11:22:10.721+02:00"
4                 ALEID="ETHZ-ALE630889259" xmlns:ns2="urn:epcglobal:ale:xsd:1">
5     <reports>
6        <report>
7           <group name="urn:epc:pat:sgtin-96:1.211298.*.*">
8              <groupList>                                                  }
9                 <member>                                                  }
10                   <tag>urn:epc:tag:sgtin-96:1.211298.0070875.1</tag>     }
11                </member>                                                 }
12                <member>                                                  }
13                   <tag>urn:epc:tag:sgtin-96:1.211298.0070875.2</tag>     } Groupe 1
14                </member>                                                 }
15                <member>                                                  }
16                   <tag>urn:epc:tag:sgtin-96:1.211298.0070876.1</tag>     }
17                </member>                                                 }
18             </groupList>                                                 }
19          </group>                                                        }
20          <group name="urn:epc:pat:sgtin-96:1.211265.*.*">                }
21             <groupList>                                                  }
22                <member>                                                  }
23                   <tag>urn:epc:tag:sgtin-96:1.211265.0056643.8</tag>     } Groupe 2
24                </member>                                                 }
25             </groupList>                                                 }
26          </group>                                                        }
27       </report>
28    </reports>
29 </ns2:ECReports>
```

L'exemple ci-dessus illustre l'utilisation du composant F&A dans un cas bien précis. De nombreuses autres spécifications peuvent être utilisées. Par exemple, ici nous utilisons le mode *« define, subscribe »*, mais il existe aussi le mode immédiat. Nous reportons aussi les identifiants des étiquettes présentes dans le champ des lecteurs (*reportSet set="CURRENT"*, ligne 14 de l'ECSpecs 2.1), mais le composant peut aussi rapporter les identifiants ajoutés ou supprimés du champ, typiquement pour le cas de requêtes telles que « Préviens moi dès qu'un produit sort du hangar ». De nombreux autres paramètres sont définis dans le standard d'EPCGlobal.

Interface d'écriture

L'interface d'écriture fonctionne de la même façon que celle de lecture. Un client fournit une spécification d'une opération (*e.g.* CCSpecs) et reçoit une réponse (*e.g.* CCReports). C'est par cette interface que les clients peuvent écrire dans les étiquettes électroniques.

Interface des lecteurs logiques

L'interface des lecteurs logiques permet de définir des groupements de une ou plusieurs sources de lectures/écritures. Par exemple, on peut définir un lecteur logique appelé « Hangar 1 » et qui contiendrait tous les lecteurs physiques présents dans ce hangar. Il est ainsi plus facile de définir

une spécification avec ce seul lecteur logique plutôt que de mettre tous les lecteurs physiques dans le fichier XML. Ces spécifications de lecteurs logiques sont appelées LRSpecs.

Il existe deux types de lecteurs logiques : les non-composites et les composites. Un lecteur logique non-composite définit et paramètre une et une seule source. Le listing 2.3 présente la déclaration d'un tel lecteur logique. La ligne 3 indique que ce n'est pas la déclaration d'un lecteur composite. Il faut ensuite préciser quel est le type d'adaptateur utilisé, ici un adaptateur LLRP (lignes 7 et 8). Il peut exister plusieurs adaptateurs, ceux-ci permettant au F&A de savoir comment communiquer avec le lecteur. Il faut ensuite donner un nom unique au lecteur (lignes 15 et 16), ainsi qu'une adresse IP (lignes 19 et 20), un port (lignes 23 et 24). Les lignes 27 et 28 indique au F&A que le lecteur est initialisé.

Listing 2.3 – *Exemple de LRSpecs pour un lecteur logique non-composite*

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <ns3:LRSpec xmlns:ns2="urn:epcglobal:ale:wsdl:1" xmlns:ns3="urn:epcglobal:ale:xsd:1">
3     <isComposite>false</isComposite>                                  } Non-composite
4     <readers/>
5     <properties>
6        <property>
7           <name>ReaderType</name>                                    } Adaptateur
8           <value>LLRPAdaptor</value>
9        </property>
10       <property>
11          <name>Description</name>                                   } Description
12          <value>LLRP reader</value>
13       </property>
14       <property>
15          <name>PhysicalReaderName</name>                            } Nom (ID)
16          <value>LogicalReader1</value>
17       </property>
18       <property>
19          <name>ip</name>                                            } Adresse IP
20          <value>localhost</value>
21       </property>
22       <property>
23          <name>port</name>                                          } Port
24          <value>5084</value>
25       </property>
26       <property>
27          <name>clientInitiated</name>                               } Initialisé
28          <value>true</value>
29       </property>
30    </properties>
31 </ns3:LRSpec>
```

Une fois les lecteurs non-composites définis, on peut définir des lecteurs composites. Dans le fichier 2.4, on définit un lecteur logique composé de deux autres lecteurs logiques. Ainsi, on pourra utiliser ce lecteur composite directement dans les spécifications.

Listing 2.4 – *Exemple de LRSpecs pour un lecteur logique composite*

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <ns3:LRSpec xmlns:ns2="urn:epcglobal:ale:wsdl:1" xmlns:ns3="urn:epcglobal:ale:xsd:1">
3     <isComposite>true</isComposite>                                   } Composite
4     <readers>
5        <reader>LogicalReader1</reader>                                } Lecteurs
6        <reader>LogicalReader2</reader>
7     </readers>
8     <properties/>
9  </ns3:LRSpec>
```

Interface de mémoire des étiquettes

L'interface pour la mémoire des étiquettes permet au F&A de traiter des données non-EPC. En effet, la définition d'un filtre ou d'un groupe pour les identifiants se fait sur les champs de cet identifiant. Par exemple, si l'on veut grouper les identifiants EPC SGTIN-96 par leur préfixe d'entreprise, on ajoute un patron de groupe définit comme suit : *urn :epc :pat :sgtin-96 :*.X.*.*.* Cette interface permet de définir d'autres structures pour les mémoires des étiquettes, offrant la possibilité de spécifier un banc mémoire, une taille de champ et un nom symbolique pour ce champ. Une fois défini, on peut alors faire du filtrage sur ce champ.

Interface de contrôle d'accès

Enfin, l'interface de contrôle d'accès ajoute des mécanismes de droits et d'accès pour les clients. On peut alors définir un ou des rôles aux clients, chacun associé à des droits d'accès et d'utilisation de fonctionnalités du F&A.

Lors de l'installation d'un tel système, nous nous retrouvons dans une configuration dans laquelle le F&A récupère tous les évènements des lecteurs. La figure 2.6 illustre une telle architecture. Les lecteurs physiques récupèrent les données des étiquettes électroniques. Ces lecteurs physiques, configurés en lecteurs logiques, sont connectés au composant F&A qui offre l'interface standard ALE aux applications métiers.

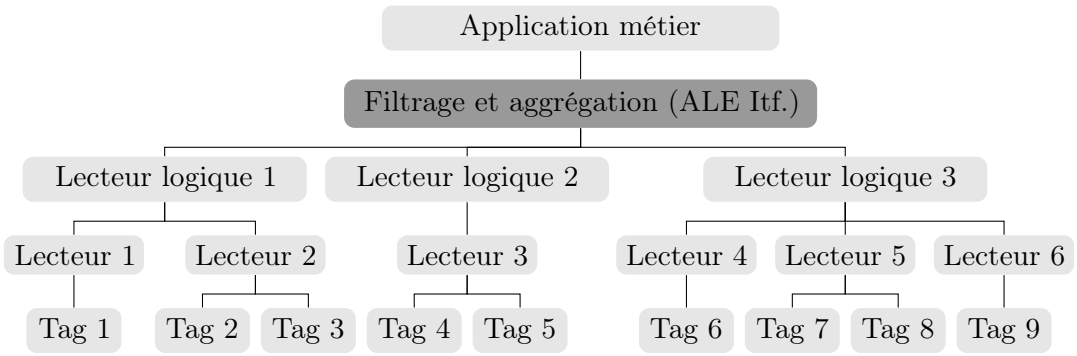


FIGURE 2.6 – *Architecture typique*

Le composant F&A offrant l'interface ALE d'EPCGlobal est un composant clé dans l'intergiciel. C'est ce qui permet d'abstraire les communication avec les lecteurs physiques. Une application métier peut ainsi questionner des lecteurs sans en connaître plus que le nom du lecteur logique (*i.e.* protocole de communication, pilote du lecteur, *etc.*).

2.4 Standard EPCIS d'EPCGlobal (EPC Information Services)

Après avoir filtré et agrégé les données lues par les lecteurs, les évènements du composant F&A (Filtrage et Agrégation, voir section précédente) peuvent être enregistrés dans une base de données. En effet, une application métier pourrait avoir besoin d'un historique des évènements, c'est là même le cœur de la traçabilité. Pour cela, EPCGlobal a défini le standard EPCIS[9], pour EPC Information Services, services d'informations sur les EPC. Ce standard spécifie la structure des données de l'EPCIS, ainsi que celles qui seront échangées via les services.

Il définit ensuite les interfaces pour les services de l'EPCIS. La figure 2.7 positionne ces interfaces entre les composants. Nous avons en bas de la figure l'interface ALE du composant

F&A, émettrice d'évènements. Ces évènements vont être récupéré par une application de capture, qui va les travailler afin de les envoyer au dépôt pour de les enregistrer via l'interface de capture de l'EPCIS. Une application métier ayant besoin d'un enregistrement de ce dépôt va émettre une requête à travers l'interface de requête de l'EPCIS. Cette interface est en fait double. Il y a une interface de contrôle et une interface de rappel pour les requêtes asynchrones.

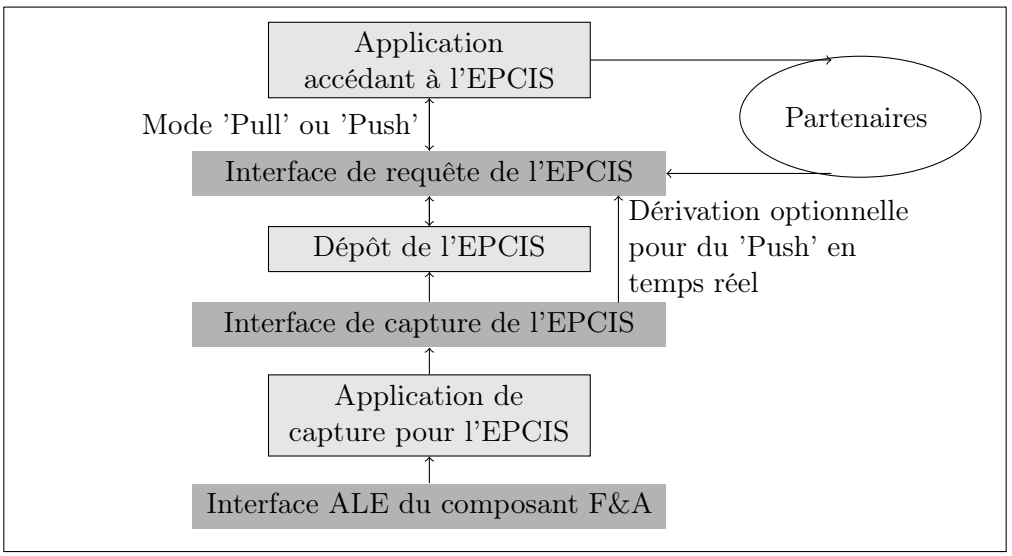


FIGURE 2.7 – *Interfaces de l'EPCIS*

L'*application de capture pour l'EPCIS* retravaille les évènements reçu par l'interface ALE. C'est le composant qui va analyser et réagir sur les problèmes métiers. Par exemple, il peut écarter une mauvaise caisse dans le système d'un convoyeur. Cette application capture les ECReports et CCReports à travers l'interface ALE et les réarrange (*i.e.* elle transforme les évènements ALE en évènements EPCIS). Elle ajoute un contexte métier et crée un ou plusieurs évènements métiers. Elle permet aussi l'agrégation avec d'autres sources de données que celles venant du composant F&A comme des données entrées par des utilisateurs ou des systèmes d'information métiers. Son rôle est comparable à celui du composant de filtrage et d'agrégation à ceci près qu'elle connait le contexte métier.

L'*interface de capture de l'EPCIS* est l'interface définissant la manière dont les évènements EPCIS vont être délivré aux couches supérieures du système (au dépôt de l'EPCIS, aux applications accédant à l'EPCIS, ou à un partenaire). Elle consiste en une méthode *capture(...)* avec comme seul argument une liste d'évènements EPCIS. Cette méthode ne retourne rien ou *void*. Le standard définit un moyen par le protocole HTTP d'envoyer les évènements d'une application de capture vers les niveaux supérieurs du système à travers cette interface de capture.

Le *dépôt EPCIS* enregistre les évènements EPCIS afin d'offrir la persistance aux données relatives aux EPC. Ces données peuvent être accessibles aux applications métiers ou aux partenaires au travers de l'interface de requête de l'EPCIS. Il agit comme une base de données et peut être implémenté de différentes façons en utilisant des systèmes de base de données bien connus comme MySQL, Oracle, *etc.*. Un évènement EPCIS est un enregistrement de quelque chose s'étant passé dans le monde réel. Dans bien des cas, il est déclenché par la lecture d'une étiquette RFID. Un évènement a quatre dimensions :

- **Quoi** : quel objet est concerné ?
- **Quand** : quand cela est-il arrivé ?
- **Où** : où cela s'est-il produit ?
- **Pourquoi** : de quel processus métier est-il issu ?

Un évènement peut être soit :

- **un évènement d'objet** : Observation d'une liste d'identifiant pendant une étape précise à un endroit et une date donné (*i.e. Ces objets ont été lus au centre de distribution* #9 *à* 10*h*04 *en réception*) ;
- **un évènement d'agrégation** : Association physique entre une liste d'identifiant et un identifiant père à un endroit et une date donné (*i.e. Ces objets ont été rassemblés dans la palette* #12 *au palettiseur* #4 *à* 12*h*32) ;
- **un évènement de quantité** : Déclaration d'une certaine quantité d'un type de produit à un endroit et une date donné (*i.e.* 200 *bouteilles d'eau de telle marque ont été identifiées dans le stock du magasin* #234 *aujourd'hui à* 15*h*20) ;
- **un évènement de transaction** : Association d'objets enregistrés avec une transaction donnée (*i.e. Commande* #123 *a été assemblée avec les objets x, y et z*).

L'*interface de requête de l'EPCIS* est divisée en deux parties : l'*interface de contrôle des requêtes* et l'*interface de rappel*. Les applications métiers et les partenaires récupèrent les données relative à un EPC au travers de l'interface de contrôle en mode synchrone ou asynchrone. En revanche, en mode asynchrone, la réponse à la requête est envoyée en *« push »* via l'interface de rappel. Cette dernière interface est aussi utilisée pour envoyer directement les données des applications de capture aux applications métiers (la flèche de « dérivation optionnelle » sur la figure 2.7).

Enfin, les *Applications accédant à l'EPCIS* sont les applications métiers qui utilisent des données relative aux EPC. C'est le module générant et envoyant les ECSpecs et CCSpecs (voir section 2.3). Par exemple, une telle application peut être le service de facturation d'une entreprise vérifiant qu'une commande a bien été expédiée avant d'éditer la facture. Cela peut aussi être une application déployée chez un partenaire, ou encore une partie du système de l'entreprise. Par exemple, cela peut être le système d'une entreprise de transport vérifiant qu'une commande est prête avant de venir la chercher. Des partenaires peuvent ainsi accéder au données relative aux EPC d'une autre entreprise à travers l'interface de requête. Mais ces partenaires doivent pour cela connaître l'adresse du dépôt EPCIS. Le standard présenté dans la section suivante sert justement de lien entre les EPCIS. Il s'agit du service de nommage des objets (Object Name Service - ONS [10]).

2.5 Standard ONS d'EPCGlobal (Object Name Service)

Le standard EPCGlobal

Dans l'architecture EPCGlobal, chaque composant logiciel a un rôle bien précis. Ainsi, le rôle de l'ALE est de filtrer et d'agréger les évènements reçus des lecteurs afin de construire le rapport et ainsi d'envoyer un évènement applicatif au réseau. Le rôle de l'EPCIS est la persistance des évènements applicatifs.

Dans le réseau EPCGlobal, le service de nommage des objects (Object Name Service - ONS [10]) joue un rôle important dans le partage des données des étiquettes RFID. Il va permettre de récupérer des URI de service sdisponibles en fonction de l'EPC contenu dans la requête. On peut ainsi retrouver l'URI du service de l'EPCIS contenant des enregistrements concernant l'EPC envoyé par exemple. Selon le standard de l'ONS, différentes URI peuvent être renvoyées, fournissant un des services suivants :

- epcis : la spécification du service d'information de l'EPC (http, https) ;
- ws : un service générique basé sur les Web Services, les services disponibles étant décrits dans le fichier WSDL de l'URI (http, https) ;
- html : une adresse d'une page html (http, https, ftp) ;

– xmlrpc : un service HTTP POST attendant une connexion compatible XML-RPC (http, https).

Une association entre une adresse compréhensible et des services associés n'est pas sans rappeler le fonctionnement du système de nom de domaine (Domain Name System - DNS [21]). En effet, celui-ci associe à la base une adresse type nom de domaine à l'adresse IP du serveur associé. Les types de réponse d'un DNS ne sont pas uniquement des adresses IP et peuvent aussi être des enregistrements NAPTR (Naming Autorithy Pointer). C'est ce type d'enregistrement qu'utilise l'ONS. Avec les NAPTR, le fonctionnement de l'ONS est identique à celui d'un DNS.

Pour cela, les identifiants EPC vont être traduits dans la forme *ons-hostname*. Comme expliqué dans la section 2.2, un identifiant EPC pour les objets à vendre est appelé SGTIN (Serialized Global Trade Number) et est composé d'un numéro identifiant le fabriquant du produit (*Company Prefix*), d'un numéro identifiant le type de produit (*Item Reference*) et d'un numéro de série (*Serial Number*). Prenons l'exemple d'un SGTIN dans sa forme *pure-identity* : *urn* : *epc* : *id* : *sgtin* : 0614141.112345.400. En analysant les différents champs de cette forme, nous savons qu'il s'agit de la forme *pure-identity* (: *id*) d'un SGTIN (: *sgtin*), que le préfixe de la compagnie est 0614141, la classe du produit est 112345 et le numéro de série de ce produit est 400. La forme *ons-hostname*, comme la forme d'une adresse d'un site web, commence par les éléments les plus précis pour finir par les plus vastes. Ainsi, l'EPC devient 112345.0614141.*sgtin.id.onsepc.com* dans sa forme *ons-hostname*. On remarquera que dans la version actuelle du standard, le numéro de série n'est pas présent dans cette forme. Une fois ce format obtenu, on peut effectuer une opération de recherche dans l'ONS. La figure 2.8 donne un exemple d'une telle opération dans l'ONS Racine, l'ONS local et l'EPCIS. La compagnie A recherche l'EPCIS contenant des informations sur un EPC. L'application va alors demander à l'ONS local de l'entreprise. Ne connaissant pas cet EPC, l'ONS local va interroger l'ONS Racine. L'ONS Racine connait l'ONS local responsable du préfixe de compagnie 0614141 et va rediriger la requête vers cet ONS local de la compagnie B. Cet ONS local connait l'adresse du service de l'EPCIS. La compagnie A peut désormais contacter l'EPCIS de la compagnie B (en supposant qu'elle y ait des droits).

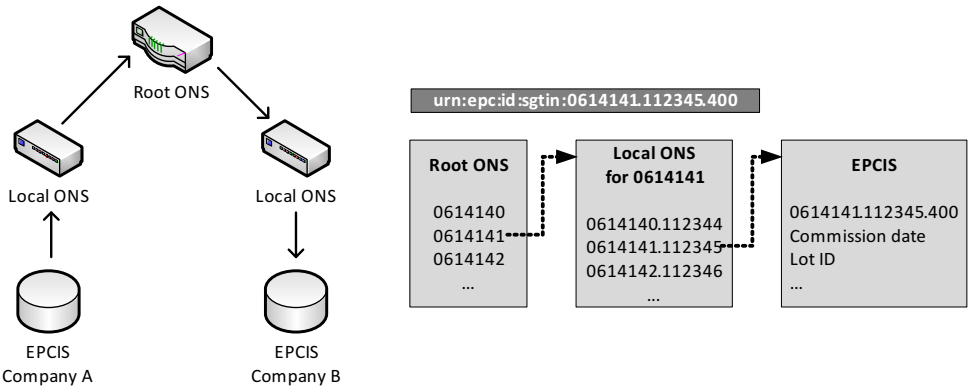


FIGURE 2.8 – *Service de recherche du standard de l'ONS*

Contraintes

Du fait de la structure hiérarchique de l'architecture des ONS, l'ONS racine est sous le contrôle d'une seule entité gouvernementale, ce qui n'est pas sans poser des problèmes politiques. En effet, une telle entité pourrait effectuer des modifications et les répliquer sur les autres serveurs. Certains serveurs racines n'autoriseront pas les mises-à-jour, mais d'autres, appartenant aussi à la-dite entité, le peuvent. On verra ainsi à long terme une fragmentation du réseau, paradoxalement aux principes de base de l'Internet. La société pour l'attribution des noms de domaine et des numéros

sur Internet (Internet Corporation for Assigned Names and Numbers - ICANN) responsable des domaines de haut niveaux (Top-Level Domain - TLD) qui, malgré l'importance qu'a pris Internet aujourd'hui et les inquiétudes de certains gouvernements sur ce sujet de gouvernance, reste contrôlée par le gouvernement des États-Unis. Cet exemple pour les noms de domaine, et le fait que VeriSign, responsable des deux domaines « .com » et « .net », héberge un ONS racine, font réagir certains acteurs de l'Internet sur les enjeux politiques d'une telle architecture.

Dans une vision globale de l'Internet des objets, où chaque objet serait équipé d'une puce RFID, les requêtes vers les ONS vont croître en fonction du développement de la RFID. En effet, plus il y a d'objets équipés d'une puce, plus il y a d'identifiants, et plus il y a de requêtes ONS afin de retrouver des informations sur ces objets. De plus, en cas de panne d'un serveur ONS, il faut que le système puisse continuer à répondre aux requêtes. Il faut donc penser au passage à l'échelle et à la robustesse de l'architecture des ONS.

Le contexte industriel, visant à faciliter les échanges de biens et nécessitant donc une interconnexion et un partage des données entre les entreprises, impose la conformité avec un standard. En effet, afin de pouvoir récupérer des informations sur n'importe quel RFID, les bases de données des constructeurs/transporteur/distributeur doivent pouvoir être questionnées. Une interface standard permettant de retrouver les services accessibles pour un EPC est indispensable afin de faciliter l'intégration de nouvelles entreprises, de nouveaux produits, *etc.*. Il est donc important d'assurer une rétro-compatibilité avec le système ONS déjà mis en place.

2.6 DHT et P2P

2.6.1 Routage indirect et tables de hachage distribuées

Dans le contexte des systèmes auto-organisés, fournir un service de localisation efficace et passant à l'échelle n'est pas un problème trivial, dû à la spontanéité des réseaux. Cela requiert une association dynamique entre l'identification et la localisation d'un nœud, et une spécification d'un mécanisme de gestion de cette association. De plus, il est nécessaire de minimiser le coût des messages de contrôle pour le routage et la découverte de localisation. Une solution efficace est le principe de *routage indirect* [22][23][24].

Une opération de routage indirect est réalisée en deux étapes : *(i)* d'abord localiser la cible et ensuite *(ii)* communiquer avec cette cible. La principale différence avec le routage classique est la façon dont la cible est localisée. Au lieu d'utiliser une grande table de routage contenant toutes les informations et adresses des cibles, dans le routage indirect, cette table est distribuée parmi les nœuds et accessible via une fonction de hachage. Les tables de hachage distribuées (Distributed Hash Tables - DHT) représentent la base du routage indirecte. Elles associent les informations et les données de localisation, établissant ainsi une couche de routage indépendante de la localisation. Cela permet au réseau de dissocier l'information sur la localisation d'un nœud de cette localisation elle-même. Avec cette approche, l'information peut être totalement distribuée, ce qui est important afin de fournir le passage à l'échelle.

La figure 2.9 présente les mécanismes utilisés pour enregistrer ou retrouver des informations avec le routage indirect et les DHT. Supposons sur le nœud k veuille enregistrer un contenu C. Durant l'opération d'enregistrement (figure 2.9(a)), le nœud k hache ce contenu C et enregistre les informations de ce contenu dans le nœud correspondant (dans la figure 2.9(a), le nœud i est responsable de l'espace d'adressage $[20, 30]$ et $Hash(C) = 25$, donc le nœud i est responsable de ce contenu). Lorsqu'un nœud veut récupérer du contenu, il calcule d'abord le hash de ce contenu et contacte alors le nœud correspondant. La figure 2.9(b) illustre le nœud j hachant le contenu C et contactant le nœud i, responsable de l'espace d'adressage $[20, 30]$. Le nœud i répond alors à j avec les informations sur C (figure 2.9(b)).

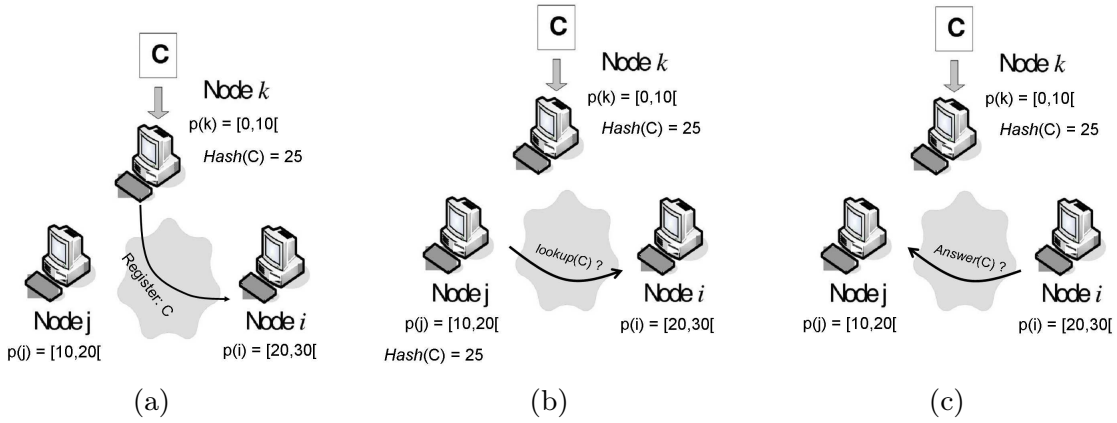


FIGURE 2.9 – *(a) L'enregistrement par le nœud k, stockant le contenu C, des informations de C sur le nœud i qui est le nœud responsable de l'espace d'adressage* $[20, 30]$*. (b) La phase de recherche du nœud j pour contacter le nœud responsable de C. (c) Réponse à la requête de recherche avec les informations sur C.*

La dynamique du réseau nécessite pour le système la capacité à gérer les arrivées, les départs et les pannes des nœuds.

Arrivée d'un nœud : Lorsqu'un nœud veut joindre le réseau, il doit d'abord contacter un nœud déjà présent dans le réseau. Ensuite, une partie de l'espace d'adressage logique lui est alors assignée. Cette arrivée implique une mise à jour des informations de routage dans le réseau. Enfin, il va récupérer toutes les paires $(clé, valeur)$ sous sa responsabilité depuis le nœud qui en était le précédent responsable.

Départ d'un nœud : Lorsqu'un nœud veut quitter le réseau, il en notifie le système avant. Cette notification permet au système de gérer le départ de ce nœud en *(i)* réassignant la partition de l'espace d'adressage logique du nœud partant aux autres nœuds et en *(ii)* enregistrant toutes les paires $(clé, valeur)$ gérées précédemment par le nœud partant dans les nœuds correspondants.

Panne d'un nœud : Lorsqu'un nœud tombe en panne, les informations qu'il stockait sont perdues. Afin de palier ce problème, certains systèmes de DHT utilisent la réplication des données et enregistrent plusieurs copies sur différents nœuds. En effet, une panne d'un nœud entraîne une perte temporaire des données jusqu'à ce qu'elles soient rafraichies.

2.6.2 Quelques systèmes de DHT

Les systèmes pair-à-pair (peer-to-peer - p2p) basés sur les DHT sont très répandus dans le domaine du partage de fichiers. Associant la localisation et l'information, de tels systèmes offrent un moyen facile de partager des données. Un nouveau nœud avec des données à partager doit juste enregistrer ses données dans la base de données et il pourra ensuite être questionné par les autres nœuds du réseau afin de récupérer ses données partagées. Le premier protocole à présenter est Napster [25]. Ce réseau fonctionne avec une DHT qui lie les adresses des nœuds avec les fichiers qu'ils partagent. Le principal problème de Napster est que cette table de liens est complètement centralisée. Une surcharge du réseau peut apparaître alors lorsque trop de nœuds sont connectés. En effet, toutes les requêtes de recherche pour un fichier sont envoyées au serveur qui gère cette table. Le passage à l'échelle d'un tel réseau réside uniquement dans la fiabilité et la robustesse du serveur. Ce problème de centralisation est résolu dans Gnutella [26], un système de partage de fichier p2p totalement décentralisé. Mais un nouveau problème apparait dans Gnutella, lors d'une requête de recherche (*i.e.* récupérer au moins une adresse d'un nœud partageant ce fichier).

En effet, un nœud doit diffuser sa requête à tout le réseau pour savoir où est stocké le fichier demandé, ce qui peut, comme la centralisation de Napster, causer une surcharge du réseau.

Les systèmes suivants proposent des solutions décentralisées et des mécanismes de recherche plus intelligents que la diffusion vers tous les nœuds. Solutionnant les deux problèmes ci-dessus, ils sont de plus en plus utilisés car plus efficaces.

Chord

Dans Chord [27], les auteurs proposent un espace d'adressage à une dimension organisé en cercle. Chaque nœud est responsable d'une partition de cet espace d'adressage et va maintenir une table de routage afin de résoudre les requêtes de recherche. Le tableau 2.10 explique comment remplir ces tables. Soit m le nombre de bits utilisés pour les clés et les identifiants des nœuds et n l'identifiant du nœud dont on construit la table. Ce nœud n devra maintenir une table d'au plus m éléments ainsi que l'identifiant de son successeur, à savoir le plus petit identifiant supérieur à n d'un nœud présent dans le réseau. La figure 2.11 illustre un réseau de trois nœuds et les

Notation	**Définition**
$finger[k].début$	$(n + 2^{k-1}) \mod 2^m, 1 \leq k \leq m$
$.intervalle$	$[finger[k].début, finger[k+1].début)$
$.nœud$	premier nœud $\geq n.finger[k].début$
$successeur$	prochain nœud sur le cercle ; $finger[1].nœud$

FIGURE 2.10 – *Définition de la table de routage des nœuds dans Chord*

tables correspondantes avec des identifiants codés sur 3 bits ($m = 3$). Si par exemple le nœud

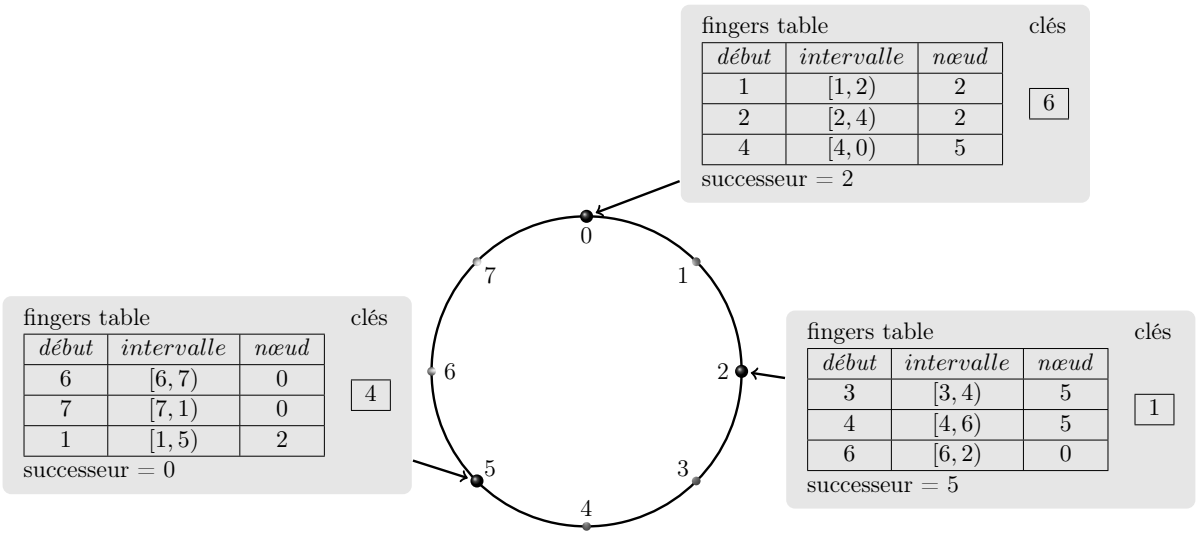


FIGURE 2.11 – *Réseau p2p Chord avec trois nœuds*

2 souhaite retrouver le nœud responsable de la clé 4, il va regarder dans sa table. L'identifiant 4 appartient à l'intervalle $[4, 6)$, 2^e entrée de la table. Le nœud 2 va alors analyser le troisième champ $finger[2].node$, soit 5. L'identifiant 5 étant supérieur à la clé 4, celui-ci est le responsable de cette clé. Chaque nœud va donc devoir maintenir des données sur $O(\log N)$ autres nœuds, et les requêtes de recherche se font en $O(\log N)$ messages.

CAN

La particularité de CAN [28] réside dans son espace virtuel : un d-tore à coordonnées Cartésiennes à d-dimensions. Grâce à son espace d'adressage, couplé à un système de coordonnées géographiques, un avantage de CAN est qu'il peut lier l'espace virtuel à l'espace physique avec $d = 3$. Le problème est la complexité additionnelle due aux d dimensions de CAN. Chaque nœud maintient une table avec $O(d)$ enregistrements et la complexité d'une requête de recherche est $O(dN^{1/d})$ avec N le nombre de nœud dans le système. La taille des informations maintenues par un nœud ne dépendant pas de N, l'espace utilisé ne grandira pas en augmentant le nombre de nœud.

Pastry

Pastry [29] utilise un espace virtuel en cercle, comme dans Chord. Le protocole de routage dans Pastry est basé sur le préfixe de l'identifiant. En d'autres termes, les requêtes de routage sont transférées au nœud possédant l'identifiant avec le plus grand préfixe commun avec l'identifiant de la donnée recherchée jusqu'à atteindre le bon nœud.

Tapestry

Tapestry [30] se focalise sur la proximité. Afin de réduire la latence du réseau, la requête est envoyée aux nœuds les plus proches de celui qui a initié cette requête. Cette recherche de la proximité augmente la complexité de certaines opérations comme celle de l'arrivée, du départ ou d'une panne d'un nœud dans le réseau.

Pour résumer, les trois concepts majeurs d'un système p2p sont la décentralisation, la charge du réseau et la dynamique de celui-ci (*i.e.* les nœuds qui arrivent ou partent du réseau). Le tableau 2.12 récapitule les coûts de chacun des systèmes présentés. La ligne *Espace* indique la quantité de données nécessaire qui doit être stockée et maintenue dans chaque nœud. Les lignes *Recherche* et *Arrivée* sont les coûts pour respectivement la phase de recherche (*i.e.* recherche d'un nœud contenant des données connues) et la phase d'arrivée (ou de départ) d'un nœud dans le système.

	Chord	CAN	Pastry	Tapestry
Espace	$\log N$	d	$\log N$	$\log N$
Recherche	$\log N$	$dN^{1/d}$	$\log N$	$\log N$
Arrivée	$\log^2 N$	$dN^{1/d} + d\log(N)$	$\log^2 N$	$\log^2 N$

FIGURE 2.12 – *Coûts des algorithmes p2p*

Ce chapitre a dressé la description des fonctionnalités offertes par un intergiciel RFID et a souligné la nécessité de son utilisation. Nous avons vu comment, à travers des standards définissants les données dans les étiquettes jusqu'au service de nommage des objets permettant l'échange d'information sur ces étiquettes, ces intergiciels se connectent entre eux afin d'offrir une traçabilité tout au long de la chaîne d'approvisionnement.

Ces standards, développés par EPCGlobal, définissent diverses couches de l'intergiciel : les standards TDS et TDT décrivent la structure et la traduction des identifiants EPC dans les étiquettes RFID ; les standards RM, RP et LLRP décrivent la gestion et les communications avec les lecteurs ; le standard ALE décrit comment filtrer et agréger les données de façon uniforme pour les couches supérieures de l'intergiciel ; le standard EPCIS décrit comment, à partir d'évènements reçu par l'interface ALE, enregistrer et récupérer les données en y ajoutant un contexte métier ;

enfin, le standard ONS décrit comment un utilisateur peut retrouver l'adresse des services d'un EPCIS d'une entreprise afin de l'interroger à partir d'un identifiant EPC.

Mais nous allons voir aussi que ces standards possèdent leurs limitations. Les chapitres suivants présenterons les problèmes et défis de déploiement, de développement de certains de ces standards compte tenu des besoins actuels comme le passage à l'échelle, la neutralité, *etc.*.

3 MACHINE A EVENEMENTS DISTRIBUEE

Le composant de filtrage et d'agrégation (F&A) tient un rôle primordial dans l'architecture de l'intergiciel. Comme expliqué dans les sections 2.1 et 2.3, il va analyser les spécifications reçues des couches supérieures, récupérer les évènements des lecteurs impliqués dans ces spécifications et construire les rapports à envoyer à ces mêmes couches. Mais un problème peut survenir avec l'architecture de la figure 2.6 lorsque trop de lecteurs sont connectés au F&A, ou qu'ils lisent trop d'étiquettes RFID. En d'autres termes, lorsque trop d'évènements de lectures sont envoyés au F&A, un goulot d'étranglement apparaît entre les lecteurs et le F&A.

Dans ce chapitre, nous proposons une solution au passage à l'échelle du composant F&A [31]. Nous commencerons, dans la section 3.1, par analyser les problèmes des solutions trouvées dans la littérature afin d'identifier les pré-requis de notre solution. La section 3.2 contient les explications des mécanismes de notre machine à évènements distribuée basée sur les DHT et la section 3.3 présente l'implémentation et les résultats de celle-ci.

3.1 Exposé du problème

3.1.1 DHT et principaux besoins

Comme nous l'avons vu précédemment, un goulot d'étranglement peut apparaître entre les lecteurs et le composant de filtrage et d'agrégation (figure 3.1). En effet, comme nous le verrons dans la section 3.3, si une centaine de lecteurs est connectée et que chacun d'eux lit une centaines d'identifiants simultanément, le réseau peut ne pas pouvoir gérer autant d'évènements, et le F&A peut ne pas être capable de traiter tant de données. Notre expérience montre une perte de données au-delà de neuf lecteurs lisant 500 d'identifiants. Un F&A pour gérer tous les lecteurs des hangars d'une entreprise n'est donc pas assez efficace. On se retrouve alors avec des rapports incomplets. Il faut donc trouver une solution au problème de **passage à l'échelle**.

Avec une duplication simple des composants F&A et une répartition intelligente des lecteurs, le problème de passage à l'échelle nécessite un remaniement du code des applications métiers, car celle-ci devront envoyer deux spécifications différentes à tous F&A dont un lecteur serait impliqué dans l'opération, et elles devront aussi fusionner autant de rapports qu'elles ont envoyé de spécifications. En effet, le standard permet de grouper les identifiants remontés par les lecteurs lors de la phase d'agrégation des données. Ces groupes devront être retravaillés au niveau de l'application métier. Si par exemple l'on souhaite regrouper dans le rapport les identifiants des produits par marque du constructeur, il faut vérifier les deux rapports. De plus, si une étiquette d'un produit se trouve proche de deux lecteurs n'étant pas connectés au même F&A, il apparaîtra dans les deux rapports. Il faut alors fusionner les informations de cet identifiant contenus dans les deux rapports pour l'intégrer dans le rapport final. Ces deux phases de division des spécifications et

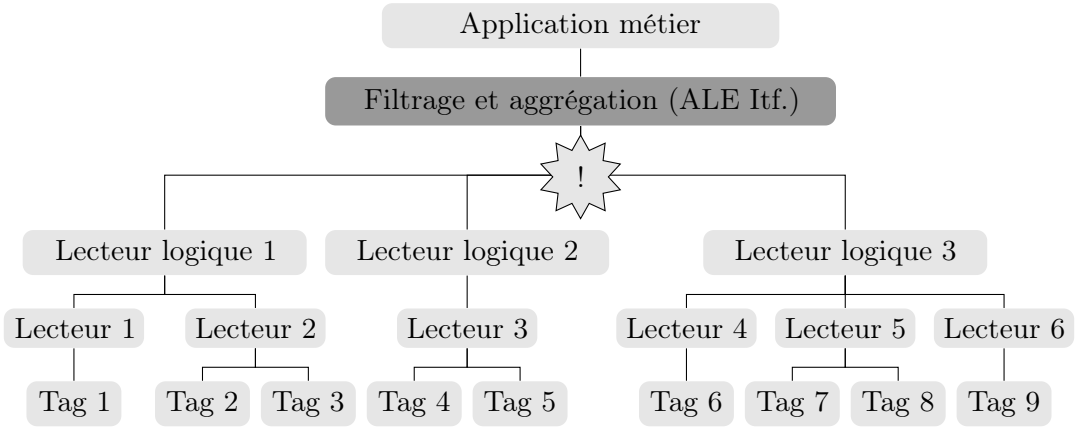


FIGURE 3.1 – *Goulot d'étranglement dans l'architecture typique*

de fusion des rapports ne sont pas dans les attributions des applications métiers. La **transparence** pour celles-ci est donc le second besoin de notre solution. De plus, cette solution ne fait que repousser le goulot d'étranglement lors d'un passage à l'échelle important. En effet, lorsque le nombre de F&A augmentera de façon conséquente, le goulot réapparaîtra entre les F&A et les applications métiers.

Les travaux de Park *et al.* [32] proposent une solution de répartition de charge au sein de plusieurs composants F&Abasée sur une duplication. La figure 3.2 reprend les lecteurs et étiquettes de l'architecture typique 3.1. Leur solution présente des mécanismes permettant de connaitre la charge des F&A, et ainsi de migrer les ECSpecs d'un F&A surchargé vers un autre moins chargé. Elle fournit aussi un moyen de migrer les lecteurs concernés par cette ECSpecs migrée vers le nouveau F&A responsable, afin de lui permettre de réaliser l'opération décrite par cette ECSpecs. Si cette solution résout le problème de surcharge de travail pour les F&A et résout en partie la transparence, elle pose néanmoins de nouveaux problèmes : *(i)* conflits : si un F&A est surchargé par cinq ECSpecs utilisant toutes au moins un lecteur identique, ces ECSpecs ne peuvent être migrées car on ne peut pas migrer les lecteurs correspondants ; *(ii)* surcharge du réseau : dans le cas d'un F&A surchargé avec des lecteurs géographiquement proches, et le F&A libre se trouvant à l'autre bout du monde, lorsque l'ECSpecs sera migrée sur ce deuxième F&A, les évènements des lecteurs vont devoir traverser le monde, ce qui n'est pas une solution économiquement viable en terme d'occupation de la bande passante, de consommation énergétique et de temps latence. Ce dernier problème apparaît aussi avec un seul composant F&Asi l'on connecte directement les lecteurs géographiquement éloignés du F&A. Mais cette connexion étant dans ce cas manuelle, cette latence est prévisible lors du déploiement. Quant à la transparence, celle-ci est vérifiée car les F&A vont déplacer eux-mêmes les lecteurs afin de répondre à la spécification. Néanmoins, lorsqu'une application veut procéder à un inventaire complet, les lecteurs seront tous déplacés vers le F&A qui exécute cette opération. C'est justement la situation que l'on veut éviter pour le passage à l'échelle.

Afin de solutionner le problème de transparence tout en offrant de la répartition de charge, Liu *et al.* [33] proposent l'insertion d'un composant entre les F&A et les applications métiers, appelé F&A global. Il va se charger de séparer les spécifications entre les F&A et de fusionner les rapports reçus des F&A contactés. Un ensemble de connections gère les données reçues des lecteurs, ainsi que les charges et IP des sous-F&A. La figure 3.3 reprend l'architecture typique (figure 3.1) dans le contexte d'un composant de filtrage et d'agrégation global.

Grâce à leur composant global, les problèmes de surcharge de travail des F&A et de transparence pour les couches supérieures sont résolus. Mais le système reste centralisé comme dans

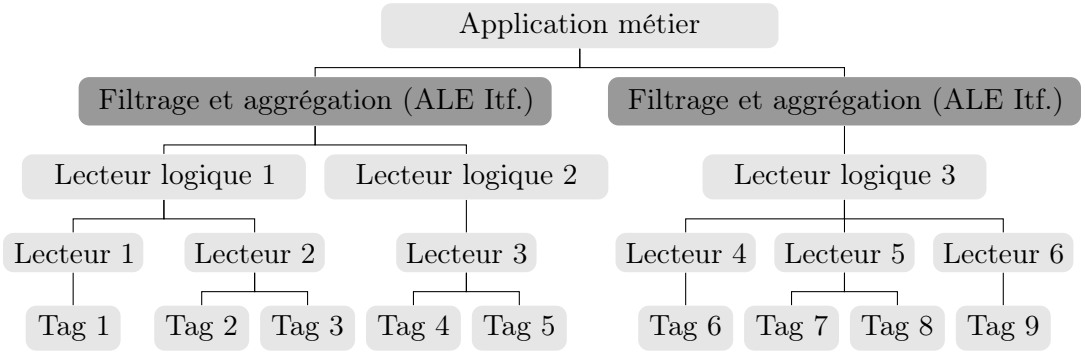


FIGURE 3.2 – *Duplication des F&A*

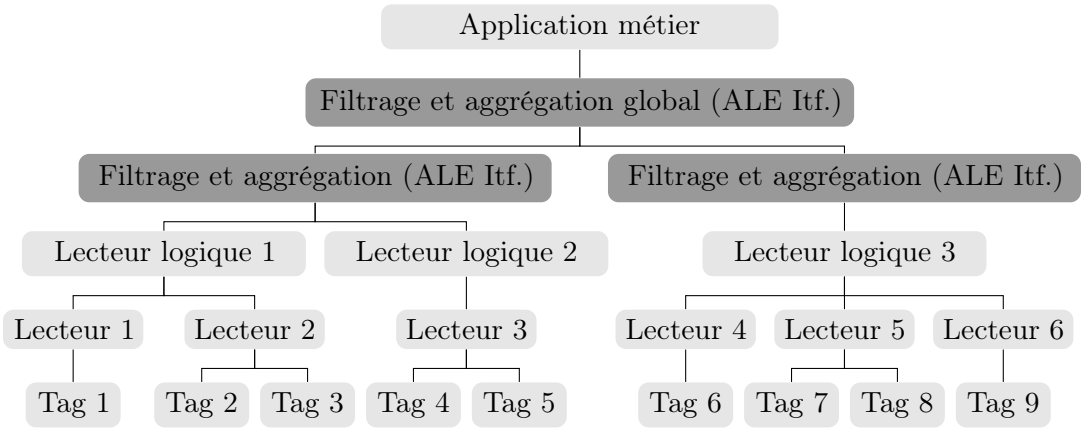


FIGURE 3.3 – *Global ALE Architecture*

l'architecture typique. En effet, lorsque le nombre de composant F&A augmente, le nombre d'évènement reçus par le F&A global augmente aussi. Ainsi un goulot d'étranglement peut à nouveau apparaître entre les F&A et le F&A global. Ces solutions ne sont pas efficaces pour un passage à large échelle. De plus, le composant global nécessite une maintenance particulière afin d'y avoir toutes les données requises pour le bon fonctionnement de l'intergiciel.

Nous avons identifié un problème sur le composant F&A tel qu'il est défini dans le standard : le passage à l'échelle. En voulant solutionner ce problème, deux autres problèmes surviennent : la conformité au standard EPCGlobal et la transparence pour les couches supérieures de l'intergiciel.

Afin de résoudre ces problèmes, nous proposons une solution basée sur les DHT. En effet, les recherches sur les systèmes pair-à-pair et les DHT en font d'excellentes techniques de passage à l'échelle, déjà très largement répandus dans les systèmes de partage de fichiers. De plus, en fournissant sur chacun des nœuds du système l'interface standard d'EPCGlobal et un système de division des spécifications et de fusion des rapports, on fournit à l'intergiciel les trois besoins requis : être en conformité avec les standards, la transparence pour les couches supérieures de l'intergiciel, et enfin le passage à l'échelle.

3.1.2 Pourquoi Chord ?

Dans la section 2.6, différents protocoles de pair-à-pair sont présentés. Nous avons décidé d'utiliser Chord car il est efficace et simple d'utilisation. Mais le protocole p2p utilisé dans notre solution n'est pas fixe, et nous pouvons tout à fait le remplacer par n'importe quel autre protocole p2p tel que CAN, Pastry ou Tapestry.

3.2 Passage à l'échelle

3.2.1 De l'ALE au pair-à-pair

Nous avons maintenant tous les éléments pour construire notre composant de filtrage et d'agrégation distribué (F&A). Nous savons comment fonctionne le standard d'EPCGlobal[8], avec les spécifications, les rapports, les lecteurs logiques, *etc.* et nous avons choisi un système de pair-à-pair (p2p)[27]. La première étape va être de lier les concepts du p2p à ceux du F&A afin de distribuer notre machine à évènements.

Dans notre solution, les nœuds du système p2p sont les machines à évènements de F&A. Ce sont elles qui partagent des informations. Les lecteurs sont les objets partagés. Comme nous l'avons vu dans la section 2.3, les lecteurs sont configurés en lecteurs logiques via des fichiers de spécifications, appelés LRSpecs. Ils peuvent être déclarés composites (exemple du fichier 2.3) ou non (exemple du fichier 2.4). Lors de cette déclaration, le nom du nouveau lecteur est utilisé comme identifiant au sein du F&A. En assignant de façon unique au travers tout le réseau cette propriété, nous pouvons alors l'utiliser comme clé pour la table de hachage. En effet, ce sont les lecteurs que les composants F&A partagent, sources de lecture d'identifiants. Si nous faisons une analogie avec les systèmes pair-à-pair pour le partage de fichiers, les données à partager sont les contenus des fichiers, et les clés, devant être uniques, sont basées sur leur nom ou leur contenu. Ici, les données à partager sont les lectures d'identifiants. Ces lectures sont générées et remontées par les lecteurs physiques, et traitées par le F&A comme si elles étaient lues par les lecteurs logiques configurés et impliqués dans l'opération. De plus, la liste des lecteurs logiques configurés sur un F&A est accessible aux couches supérieures via son interface. Du point de vue de ces couches supérieures, ce sont donc les lecteurs logiques qui émettent les lectures, comme les fichiers possèdent du contenu.

Conservons encore un peu l'analogie avec les systèmes de partage de fichiers. Un fois que l'on connait la clé du fichier qui nous intéresse, il faut alors retrouver les adresses des pairs partageant ces fichiers, et donc possédant le contenu partagé, afin de récupérer ces données. Dans notre cas, lorsque l'on désire interroger un F&A, il faut connaître les informations de connexions à celui-ci. Ces informations dépendent de l'implémentation, dans notre cas, il s'agit de l'URL du service web qu'il fournit. En effet, l'interface ALE du composant F&A est accessible par les web services, c'est donc cette donnée qui est à partager. Ainsi, les enregistrements des systèmes de partage de fichiers qui sont définis par l'identifiant du fichier auquel sont liées les adresses contenant ces fichiers, deviennent pour notre cas des paires *<identifiant du lecteur (ou ReaderName), URL du composant F&A auquel il est connecté (ou nodeURL)>*. Dans le reste de cette section, les termes *F&A*, *pair* ou *nœud* désignent tous les trois les pairs du réseaux. Les termes *lecteur* ou *objet* sont utilisés pour représenter les lecteurs logiques partagés du système et le terme *enregistrement* correspond à une paire *<ReaderName, nodeUrl>*.

Les termes sont posés et les liens entre le réseau pair-à-pair et les éléments du F&A sont définis. La figure 3.4 présente un exemple de réseau de F&A et leurs lecteurs connectés lors d'un enregistrement d'un nouveau lecteur à partager. Le réseau est constitué de cinq composants F&A sur lesquels sont connectés trois lecteurs. Lors de la connexion d'un nouveau lecteur, ici le lecteur 1, ce dernier va d'abord être configuré localement sur le F&A 4 (flèche ❶). Ensuite, ce nœud 4 va utiliser la fonction de hachage afin de savoir où déclarer ce nouveau lecteur. Finalement, le nœud 4 va envoyer l'enregistrement *<Nom du lecteur* 1*, URL du nœud* 4*>* au nœud de contact 5 (flèche ❷). Une fois cela fait, chaque nœud va pouvoir retrouver l'URL du nœud 4 gérant le lecteur 1 en interrogeant le nœud 5.

C'est le mécanisme présenté dans la figure 2.9(a), où le nœud k serait le F&A 4, le nœud i serait le F&A 5 et l'information sur C serait l'URL du F&A 4 qui gère le lecteur 1.

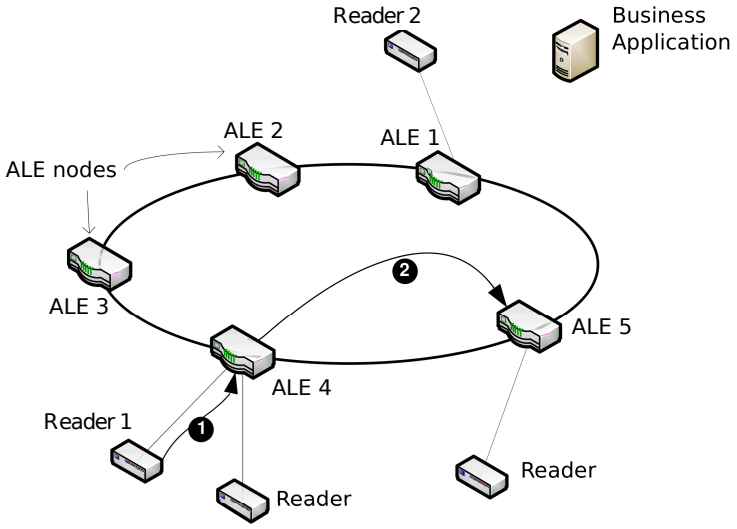


FIGURE 3.4 – *Enregistrement d'un nouveau lecteur dans le système*

3.2.2 Mécanismes d'utilisation

Le réseau est maintenant configuré, les lecteurs sont enregistrés, nous pouvons désormais utiliser notre machine à évènements distribuée.

Lorsqu'une application métier veut effectuer une opération, elle doit d'abord définir une spécification (*i.e.* l'ECSpecs) compatible avec le standard d'EPCGlobal. Reprenons pour cela la spécification 2.1 (section 2.3) et appliquons-la au réseau défini par la figure 3.5. Elle déclare deux lecteurs logiques impliqués dans l'opération (LogicalReader1 et LogicalReader2, lignes 5 et 6) et demande, toutes les 10 secondes, la liste de toutes les étiquettes RFID lus par ces deux lecteurs pendant 9,5 secondes (lignes 8 à 10 et ligne 14).

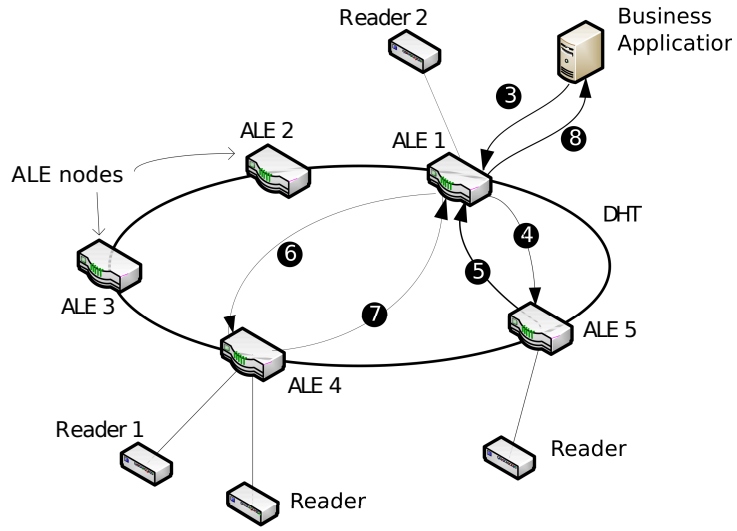


FIGURE 3.5 – *Etapes de génération de rapport dans le système distribué*

Dans le réseau de la figure 3.5, où l'on retrouve les cinq nœuds et les quatre lecteurs logiques, les lecteurs 1 et 2 sont configurés comme non-composites avec respectivement les noms *LogicalReader1* (connecté et configuré sur le nœud 4) et *LogicalReader2* (connecté et configuré sur le nœud 1). Quand l'application métier souhaite effectuer l'opération définie par l'ECSpecs 2.1, elle doit

envoyer cette spécification à un nœud du réseau. Ça peut être le plus proche, le moins chargé ou un autre, cela dépend des choix d'implémentation et/ou de configuration. Il peut être intéressant de contacter le nœud le plus proche pour éviter la latence, mais il peut aussi être mieux dans certains cas de contacter le nœud connecté au plus grand nombre de lecteurs impliqués dans la spécification. Ici, l'application métier envoie sa spécification au nœud 1 (flèche ❸). Ce nœud connait évidemment les lecteurs qui lui sont connectés, et en analysant le fichier de spécification 2.1, il sait que le lecteur 1 impliqué n'est pas connecté directement, mais le lecteur 2 l'est. L'opération est donc distribuée sur deux nœuds. Il va falloir diviser la spécification et en envoyer une vers l'autre nœud impliqué. Le nœud 1 va donc faire appel à la fonction de hachage sur l'identifiant du lecteur 1 *LogicalReader1* et va ainsi connaître le nom du nœud responsable de l'enregistrement concernant ce lecteur. Ici, le nœud responsable est le nœud 5. Le nœud 1 va donc questionner le nœud 5 à la recherche de l'enregistrement (flèche ❹), et ce dernier va lui répondre avec l'URL du nœud 4 sur lequel est connecté le lecteur 1 (flèche ❺).

L'opération de division de la spécification consiste à créer deux fichiers de spécifications : (i) l'un avec uniquement le lecteur 1 (Figure 3.1) et qui va être envoyé au nœud 4 grâce à l'URL reçue du nœud 5 (arrow ❻) ; (ii) l'autre avec uniquement le lecteur 2 et qui va rester en local sur le nœud 1 (Figure 3.2). Cette opération est effectuée par le nœud qui a été contacté par l'application, ici le nœud 1.

Listing 3.1 – *ECSpecs pour le F&A*1

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ECSpec xmlns:ns2="urn:epcglobal:ale:xsd:1">
   <logicalReaders>
      <logicalReader>LogicalReader2</logicalReader>
   </logicalReaders>
   <boundarySpec>
      <repeatPeriod unit="MS">10000</repeatPeriod>
      <duration unit="MS">9500</duration>
      <stableSetInterval unit="MS">0</stableSetInterval>
   </boundarySpec>
   <reportSpecs>
      <reportSpec>
         <reportSet set="CURRENT"/>
         <output includeTag="true"/>
      </reportSpec>
   </reportSpecs>
</ns2:ECSpec>
```

Listing 3.2 – *ECSpecs pour le F&A*4

```
 1. <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:ECSpec xmlns:ns2="urn:epcglobal:ale:xsd:1">
   <logicalReaders>
      <logicalReader>LogicalReader1</logicalReader>
   </logicalReaders>
   <boundarySpec>
      <repeatPeriod unit="MS">10000</repeatPeriod>
      <duration unit="MS">9500</duration>
      <stableSetInterval unit="MS">0</stableSetInterval>
   </boundarySpec>
   <reportSpecs>
      <reportSpec>
         <reportSet set="CURRENT"/>
         <output includeTag="true"/>
      </reportSpec>
   </reportSpecs>
</ns2:ECSpec>
```

Les deux F&A impliqués vont pouvoir maintenant effectuer l'opération de la spécification localement. Typiquement, il s'agit de configurer le lecteur, de lancer l'inventaire, de filtrer et d'agréger les évènements du lecteur et de construire le rapport. Une fois qu'il a construit son

rapport local (ou reçu celui du nœud 4), le nœud 1 sait qu'il a besoin d'un deuxième rapport, et va donc attendre d'avoir les deux pour poursuivre. Lorsque les deux rapports sont présents dans le nœud 1 (flèche ❼), ce dernier va pouvoir les fusionner en un seul, filtrant et agrégeant éventuellement à nouveau les données. Enfin, le nœud 1 qui avait reçu la spécification complète va alors pouvoir envoyer le rapport bien construit à l'application métier demandeuse (flèche ❽).

Ceci est un exemple simple montrant le fonctionnement lors de la déclaration d'une spécification avec des lecteurs distants, et donc avec plusieurs F&A. Nous l'illustrons en n'utilisant que deux lecteurs, mais les mécanismes sont les mêmes dans le cas de spécifications impliquant plusieurs lecteurs. On aura une ECSpecs par composant F&A impliqués, et le F&A contacté devra attendre autant de rapports avant de procéder à l'étape de fusion des rapports. Cette étape de fusion est une étape importante permettant d'enlever les éventuels doublons ou pour la reconstruction des groupes afin de fournir la transparence aux applications métiers. En effet, une ECSpecs peut spécifier des patrons de groupement, afin de grouper les ID des étiquettes électroniques lues par les lecteurs (*e.g.* grouper par numéro de fabriquant, par type d'objet, *etc.*). Avant d'envoyer le rapport final à l'application métier, le nœud 1 de notre exemple doit vérifier et reconstruire les groupes. Ce sont ces étapes de division des spécifications et de fusion des rapports qui permettent une complète transparence. Une autre caractéristique intéressante est l'utilisation des interfaces standards ALE Reading API et ALE Logical Reader API définies par EPCGlobal entre les nœuds et pour les couches supérieures de l'intergiciel. Cette compatibilité aux standards EPCGobal permet de remplacer n'importe quel F&A déjà déployé par un nœud de notre système sans pour autant avoir à réécrire les applications métiers. Enfin, construire le composant F&A de l'intergiciel sur un système pair-à-pair offre les autres caractéristiques recherchées, à savoir le passage à l'échelle, la dynamicité et la robustesse.

3.2.3 Lecteurs logiques distribués

Notre principal avantage quant au passage à l'échelle sur la méthode simple de duplication des composants F&A est la transparence pour les couches supérieures. En effet, en utilisant l'interface standard d'EPCGlobal, une application métier qui serait configurée pour utiliser des lecteurs connectés au même nœud n'aurait rien à changer dans sa configuration. En revanche, en prenant l'exemple d'une application d'inventaire de tous les hangars d'une entreprise, celle-ci doit changer si l'on ajoute un nouveau hangar. Alors que les nouveaux lecteurs sont configurés sur le nœud de ce nouveau hangar, et donc dans la DHT, cette application va envoyer la même spécification qui n'inclut pas ces nouveaux lecteurs. Il va donc falloir la retoucher afin d'y ajouter ce hangar dans le processus d'inventaire. Entre ajouter un lecteur dans un XML généré, ou mettre en place la division/fusion des ECSpecs/ECReports dans les applications métiers, on devine quelle tâche est la plus simple. Il n'empêche qu'on a perdu la transparence. Le moyen de contourner cela est de pouvoir définir des lecteurs logiques distribués. Ainsi, en définissant un lecteur logique global pour cette application d'inventaire, il suffirait d'ajouter le nouveau hangar dans la définition de ce lecteur global. L'application métier n'a plus rien à changer, tout se fait à la configuration du nouveau nœud. Voyons ce processus plus en détails.

Dans notre DHT, les enregistrements sont des paires *<ReaderName, nodeUrl>*. On peut ainsi récupérer l'URL du service web du composant F&A connecté au lecteur. En ajoutant un autre type d'enregistrement, toujours avec un nom de lecteur en clé, mais avec une liste de noms de lecteurs en valeur, nous pouvons définir des lecteurs logiques distribués. Reprenons notre réseau de la figure 3.4. Après l'enregistrement du lecteur 1, notre base de données distribuée contient quatre enregistrements :

- Lecteur 1, URL du nœud 4
- Lecteur 2, URL du nœud 1
- Lecteur 3, URL du nœud 4

– Lecteur 4, URL du nœud 5

En ajoutant un enregistrement du type *<ReaderName, ReaderNamesList>*, on peut définir un lecteur logique appelé "Lecteur distribué 1", regroupant les lecteurs 1 et 2 par exemple. Notre base de données serait remplie comme ceci :

– Lecteur 1, URL du nœud 4
– Lecteur 2, URL du nœud 1
– Lecteur 3, URL du nœud 4
– Lecteur 4, URL du nœud 5
– Lecteur distribué 1, (Lecteur 1, Lecteur 4)

La figure 3.6 présente l'exécution d'une ECSpecs dont le listing 3.3 ne diffère de l'ECSpecs 2.1 que dans les lecteurs logiques impliqués. Cette fois, nous utilisons notre lecteur distribué. Lorsque le nœud 1 reçoit cette spécification de l'application (flèche ❶), il commence par analyser les lecteurs logiques impliqués, comme avant. Il ne connait pas localement le lecteur distribué 1 et va donc exécuter une recherche dans la DHT. En hachant le nom de ce lecteur, il va connaitre le nœud responsable de l'enregistrement, le nœud 2, qu'il va interroger (flèche ❷). La réponse (flèche ❸) indique que ce lecteur est constitué du lecteur 1 et du lecteur 2. Nous nous retrouvons maintenant dans le même cas que dans la figure 3.5. Le nœud 1 connait le lecteur 2, hache le nom du lecteur 1, trouve le nœud 5, l'interroge (flèche ❹) et récupère l'URL du nœud 4 (flèche ❺). Il divise la spécification, en envoie un partie au nœud 4 (fèche ❻), attend les rapports (flèche ❼), les fusionne, et envoie le rapport final à l'application (flèche ❽).

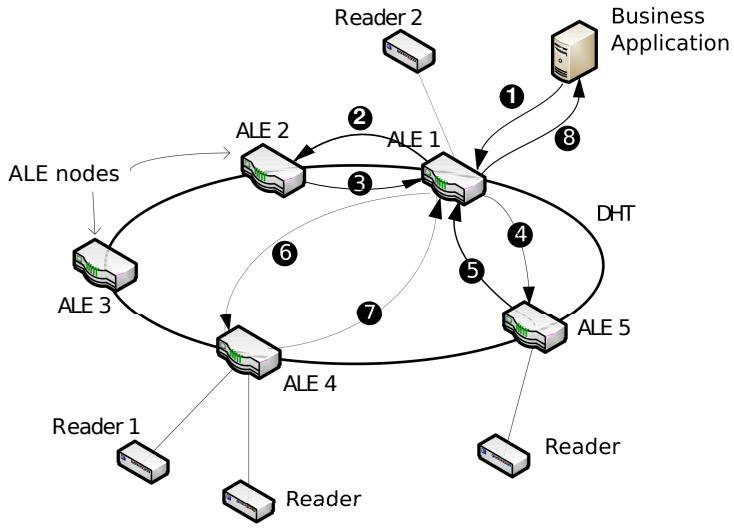


FIGURE 3.6 – *Etapes de génération de rapport avec un lecteur distribué*

Listing 3.3 – *Exemple d'ECSpecs*

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <ns2:ECSpec xmlns:ns2="urn:epcglobal:ale:xsd:1">
3     <logicalReaders>
4        <logicalReader>Lecteur distribué 1</logicalReader>
5     </logicalReaders>
6     <boundarySpec>
7        <repeatPeriod unit="MS">10000</repeatPeriod>
8        <duration unit="MS">9500</duration>
9        <stableSetInterval unit="MS">0</stableSetInterval>
10    </boundarySpec>
11    <reportSpecs>
12       <reportSpec>
13          <reportSet set="CURRENT"/>
```

```
14            <output includeTag="true"/>
15          </reportSpec>
16        </reportSpecs>
17    </ns2:ECSpec>
```

Ainsi, avec une application métier développée pour utiliser un lecteur distribué, il suffit de modifier l'enregistrement dans la DHT pour intégrer de nouveaux lecteurs à ce lecteur distribué. L'application métier n'a plus besoin de rafraichissement. La transparence est obtenue.

La section suivante présente des évaluations et des résultats de notre solution.

3.3 Implémentation et résultats

3.3.1 Implémentation

La figure 3.7 présente l'architecture d'un nœud de notre réseau de machine à évènements distribuée. Les applications métiers utilisent l'interface ALE du standard EPCGlobal pour communiquer les spécifications à l'intergiciel et recevoir les rapports. Un fois la spécification reçue, le composant F&A distribué va analyser les lecteurs logiques impliqués dans cette spécification. Grâce à la fonction *getLogicalReaderNames()* du standard, il peut connaître les lecteurs logiques configurés localement. Ainsi, et avec l'aide de la couche OpenChord [34], notre implémentation distribuée du composant F&A [35] va pouvoir déterminer les lecteurs logiques distants et diviser les spécifications pour en créer une par composant F&A impliqué. Ce composant F&A distribué possède une deuxième interface compatible EPCGlobal, mais utilisant un format binaire plutôt que des services web comme le ferait une application métier. Cette interface permet aux composants F&A distribués d'effectuer les mêmes opérations qu'une application métier. C'est ainsi qu'il va pouvoir envoyer des spécifications et attendre des rapports. En effet, lorsque le composant F&A distribué envoie une spécification et attend le rapport d'un autre nœud, ce premier se comporte alors comme une application métier cliente et va utiliser les méthodes *define() - subscribe()* ou *immediate()* du standard pour interroger les F&A distants possédant des lecteurs impliqués.

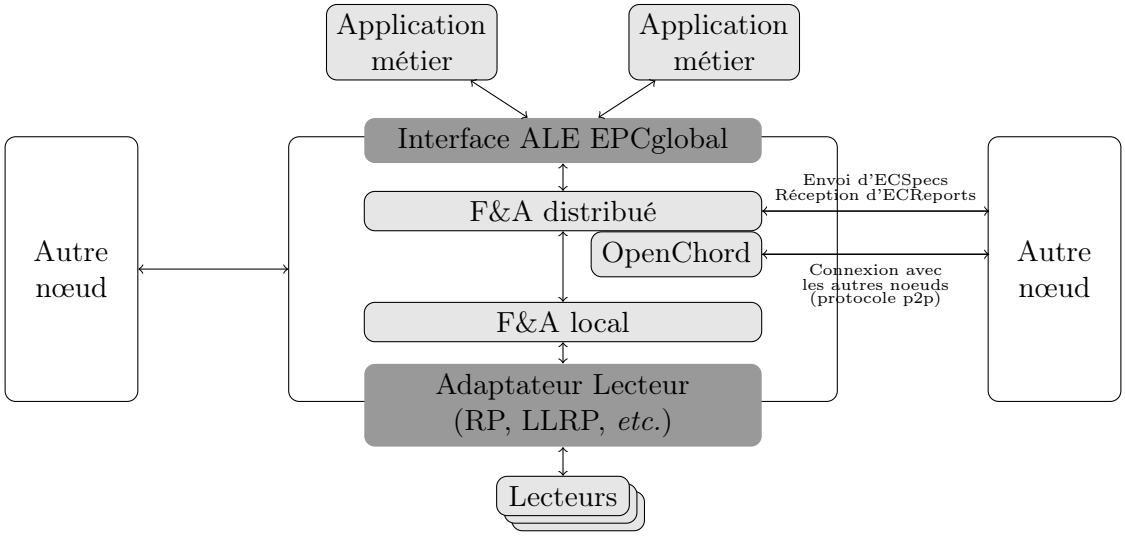


Figure 3.7 – *Architecture des composants d'un F&A distribué.*

Lorsque le composant F&A distribué a divisé la spécification, il envoie les spécifications distantes via l'interface binaire de connexion inter-nœud, et va envoyer la partie à effectuer localement (si il y a des lecteurs logiques locaux impliqués) vers le composant F&A local afin qu'il effectue l'opération telle que décrite dans le standard EPCGlobal. Ainsi, notre machine distribuée est pleinement compatible avec le standard actuel, elle peut même être utilisée seule : si tous les

lecteurs logiques impliqués dans la spécification sont définis localement, alors le comportement de notre machine est identique à celui d'une machine non distribuée.

Afin de pouvoir avoir de nombreux lecteurs et de nombreuses étiquettes électroniques, nous avons écrit un simulateur de lecteur LLRP [36]. Celui-ci permet au F&A de se connecter au lecteur et simule la lecture d'étiquettes RFID. Ainsi, notre simulateur notifie la lecture de 500 étiquettes toutes les 5000 ms et de *Keep-Alive* sont envoyés toutes les 10000 ms afin de conserver la connexion entre le simulateur et le F&A.

La spécification utilisée demande de rapporter toutes les 12000 ms les évènements de lectures reçus pendant 6000ms.

3.3.2 Résultats

En introduisant un mécanisme de division et de fusion des rapports, ainsi que des processus attendant le retour de rapport, nous avons ajouté du travail au composant F&A. Afin d'évaluer notre solution, nous devons rappeler les objectifs : une machine à évènements distribuée, conforme aux standards, passant à l'échelle et dont l'utilisation est transparente pour les couches supérieures du l'intergiciel. La conformité au standard est vérifiée par l'utilisation des interfaces définies dans ces mêmes standards. La transparence est assurée par les mécanismes de division et de fusion des spécifications et des rapports. Pour mesurer le passage à l'échelle, nous devons savoir si ces deux mécanismes ainsi que l'interrogation de la DHT ne coutent pas déraisonnablement plus cher que la solution de duplication simple du F&A, supprimant la transparence.

Pour cela, nous avons fixé le nombre d'étiquettes lues par un lecteur logique à 500, et nous avons fait varier le nombre de lecteurs logiques connectés afin de connaître le seuil à partir duquel le rapport reçu par l'application métier est incomplet.

Ainsi, comme le montre la courbe 3.8, nous observons pour un seul composant F&A en utilisation classique non distribuée des pertes et des rapports incomplets à partir de 10 lecteurs logiques envoyant chacun les 500 identifiants. Le F&A utilisé dans ce test est le même que celui du test distribué, à ceci près qu'il ne se connecte jamais à d'autre nœud et n'interroge jamais la DHT.

Nous avons ensuite ajouté un nœud dans le réseau Chord des F&A, et nous avons recommencé à faire varier le nombre de lecteurs afin de retrouver ce même seuil (figure 3.8). Ce seuil apparaît à 14 lecteurs pour nos deux F&A.

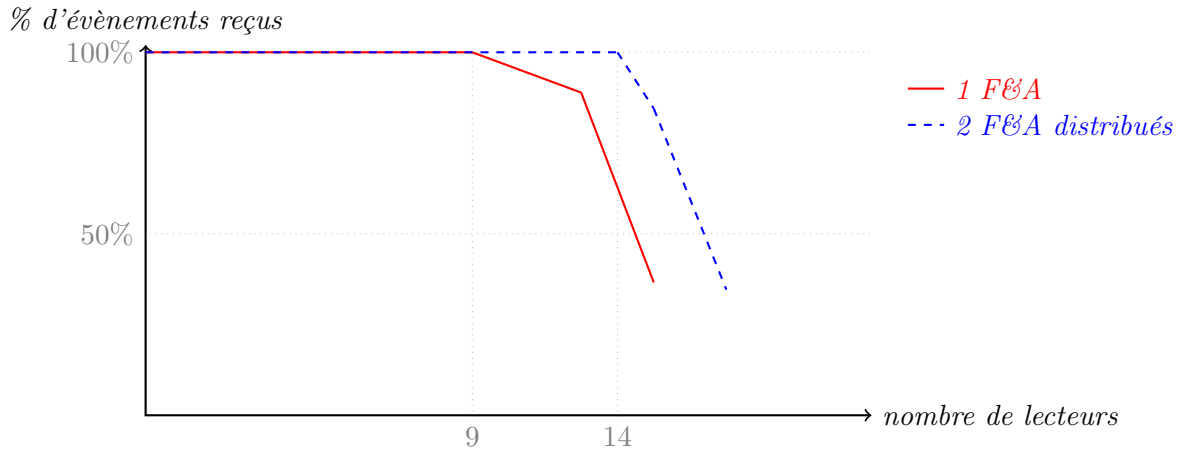


FIGURE 3.8 – *Apparition de pertes d'évènements dans les composants F&A*

Ces résultats montrent que pour gagner la transparence au niveau des applications métiers, nous avons une perte de lecteurs dans notre contexte. Mais non seulement la transparence

aux niveaux des applications métiers est acquise, mais leur configuration est facilitée. En effet, avec une duplication simple des composants F&A, lors de la configuration des applications métiers, il faut connaître *(i)* le nom des lecteurs logiques impliqués dans les applications métiers, informations nécessaires aussi avec notre solution distribuée ; mais surtout *(ii)* l'URL des services des composants F&A connectés aux lecteurs impliqués, ce dont on a pas besoin dans notre solution.

Par manque de temps, nous avons pu mettre en évidence uniquement le surcoût apporté par la transparence. Des tests plus poussés sur le passage à l'échelle, avec plus de lecteurs, plus de nœuds, plus de requêtes auraient été intéressants afin de vérifier que ce surcout ne croît pas exponentiellement, même si notre sentiment à l'égard de ce surcout est qu'il va devenir négligeable face à la complexité de développement et de paramétrage des applications métiers branchées sur un intergiciel n'offrant pas notre transparence. Il aurait été aussi intéressant de chercher d'autres causes à ce surcout, afin de le faire baisser. En effet, même si nous avons optimisé les communications entre les nœuds au niveau des interfaces ALE, les mécanismes en jeu ne sont plus tout à fait identiques, des optimisations techniques peuvent être éventuellement apportées après de plus amples investigations.

3.4 Conclusion

Nous avons analysé les besoins d'un intergiciel distribué pour la RFID. Nous avons ainsi identifié le besoin de conformité avec les standards développés. En effet, un des objectifs clés de cet intergiciel est son inter-opérabilité. Être conforme aux standards permet non seulement d'intégrer les systèmes existant sans trop d'effort de déploiement supplémentaire, de faciliter le développement d'applications métiers et les interactions avec les couches supérieures de l'intergiciel. Le passage à l'échelle est aussi un des besoins identifiés pour cet intergiciel. En effet, dans la vision de l'Internet des Objets où tous les objets à travers le monde seraient équipés de puces RFID, il faudra un intergiciel distribué afin de faciliter l'interrogation des lecteurs connectés. Dans un effort pour faciliter encore le déploiement et le développement des applications métiers utilisant cet intergiciel, la transparence pour les couches supérieures est un besoin très important. Aidé par l'utilisation des standards, nous proposons une solution basée sur les tables de hachage distribuées pour assurer le passage à l'échelle, une concordance aux standards d'EPCGlobal et une transparence pour faciliter le développement d'applications. Cette transparence est gagnée grâce effectivement à la conformité avec le standard, mais aussi grâce à deux mécanismes de division des spécifications et de fusion des résultats.

Ainsi, nous avons développé une machine à évènements distribuée passant à l'échelle, transparente pour les couches supérieures et conforme aux standards. Mais , comme précisé à la fin de la section précédente (section 3.3), il serait intéressant de chercher les causes du surcout de la solution, surcout qui selon mon sentiment est trop élevé. De plus, des tests à large échelle permettrait de valider les avantages de la solution.

Publications

1. Rapport de recherche :
   - *DHT-based distributed ALE engine in RFID Middleware.* L. Schmidt, R. Dagher, R. Quilez, N. Mitton et D. Simplot-Ryl. RR-7316, INRIA, 2010
2. Développements logiciels :
   - ALE distribuée. *http ://chercheurs.lille.inria.fr/schmidt/Research.html*
   - Simulateur LLRP. *http ://chercheurs.lille.inria.fr/schmidt/Research.html*

4 Infrastructure auto-configurable pour les réseaux orientés gestion des stocks

La gestion des biens dans un monde où le commerce est grandissant nécessite de plus en plus de support technologique pour la logistique et la traçabilité. L'exemple de l'inventaire est révélateur : les entreprises étant de taille de plus en plus importante, leurs stocks (eux aussi de plus en plus importants) peuvent être répartis sur plusieurs sites, rendant les opérations d'inventaire fastidieuses.

La solution généralement adoptée est l'utilisation de serveurs et de bases de données (composant EPCIS d'EPCGlobal [9]) connectés de manière à former un réseau et à être accessibles depuis n'importe où. Ces bases de données sont alimentées par les données provenant des lecteurs de codes barre ou d'étiquette RFID présents sur les objets. Certains lecteurs RFID peuvent également jouer le rôle de base de données et être directement reliés au réseau.

Bien que cette architecture de réseau puisse paraître générale, il est préférable que l'infrastructure de communication prenne en compte les besoins et les spécificités de l'application de gestion de stocks. Dans ce chapitre, nous nous concentrons sur une application de gestion de stocks comprenant une grande quantité de produits répartis sur plusieurs sites géographiques et nécessitant un coût en communications faible et un stockage des données fiable. Le réseau formé par les différentes bases de données doit se conformer aux standards EPCGlobal. L'identification des produits se fait par la lecture de leurs identifiants (sous forme de RFID ou de code barre par exemple). Pour assurer les opérations d'identification, de traçabilité et d'inventaire, le réseau est composé de serveurs de base de données, de routeurs ou de lecteurs fixes ou mobiles.

Notre première motivation est d'assurer le passage à l'échelle et la fiabilité d'un tel réseau. En effet, la possible augmentation du nombre d'entités dans le réseau ne doit pas surcharger le réseau. De même, les données doivent être dupliquées pour assurer la fiabilité. Cependant, cette duplication doit se faire de manière intelligente pour minimiser le surcout qui lui est associé.

Notre seconde motivation est l'auto-organisation du réseau et son adaptation à chaque modification de manière transparente et localisée. De plus, plusieurs niveaux de granularité sont nécessaires pour les différentes primitives de diffusion utilisées. Ces primitives sont la diffusion (*Compter le nombre de produits d'un type spécifique*), le k-cast (*Y a t'il une quantité k de ce type de produits ?*) ou l'anycast (*Quel est le prix de ce produit ?*).

Ce chapitre présente SENSATION [37], une structure auto-configurable pour la gestion de stocks. Cette structure offre un stockage fiable (avec un niveau de réplication des données k personnalisable) ainsi qu'un traitement rapide des requêtes tout en présentant un surcoût en terme de bande passante faible. Elle gère plusieurs types d'opérations comprenant le stockage distribué (pour des données accessibles rapidement), la réplication de données (pour des données

fiables et la tolérance aux pannes) et la gestion efficace de requêtes (pour des réponses limitant le trafic et le surcoût lié au flooding).

La solution proposée utilise deux tables de hachage distribuées (Distributed Hash Table - DHT). La première permet d'atteindre le groupe correct correspondant au type de l'objet pointé par la requête. La seconde permet d'atteindre le nœud possédant effectivement cette donnée. Par exemple, la requête *Quel est le prix de tel pantalon ?* sera envoyée par la première DHT à la couche responsable des pantalons, et la seconde DHT l'enverra à un nœud connaissant le prix des pantalons du type demandé.

Pour présenter SENSATION, nous présentons dans une première section un cas d'étude qui met en évidence les besoins d'une application de gestion de larges stocks. La section 4.2 présente les protocoles utilisés dans SENSATION. La section 4.3 est consacrée à la description de notre solution. Une évaluation des performances de SENSATION est proposée dans la section 4.4.

4.1 Étude de cas

Notre cas d'étude porte sur une application de gestion de larges stocks répartis sur une zone géographique importante. Il peut s'agir par exemple de l'application d'un distributeur dont les entrepôts sont répartis sur un ou même plusieurs pays ; dans chacun de ces entrepôts sont disposés des serveurs et des bases de données interconnectés formant ainsi un réseau. La gestion d'une telle organisation suppose de disposer de plusieurs outils à différents niveaux.

Pour donner un aperçu de ces différents outils, considérons le cas d'un utilisateur situé dans un entrepôt donné. Il peut vouloir faire un inventaire :

- de l'ensemble des entrepôts ;
- d'un entrepôt particulier situé dans une zone géographique particulière ;
- d'une zone particulière d'un entrepôt particulier.

Outre sa dimension géographique, ce même inventaire peut porter sur la nature des produits à inventorier et donc concerner :

- un type de produits (*"inventaire de tous les articles de sport"*) ;
- un type de produits encore plus précis (*"inventaire de tous les pantalons de sport"*) ;
- un type de produits encore plus précis situés dans un lieu donné (*"inventaire des pantalons de sport de taille XL situés à Paris"*) .

Ainsi, l'application devra donc être capable de gérer des requêtes de granularités différentes aussi bien sur les types de produits que sur les lieux, *etc.*. Enfin, ces requêtes peuvent également être de natures différentes, et peuvent par conséquent nécessiter des mécanismes différents concernant leur diffusion. On peut par exemple considérer les mécanismes suivants :

- les requêtes de broadcast (*Combien de produits de tel type ?*) : pour répondre à ce type de requêtes, il faut interroger tous les détenteurs d'informations sur des produits du type demandé ;
- les requêtes nécessitant du k-cast (*Y a t-il au moins k produits de tel type ?*) : il n'est pas indispensable d'interroger tous les détenteurs d'une information sur un produit du type demandé, on peut s'arrêter lorsque k produits ont été trouvés ;
- des requêtes anycast (*prix de ce produit ?*) : n'importe quel détenteur de cette information peut répondre, il n'est en aucun cas nécessaire de tous les interroger.

Si on utilise une structure de base de données »à plat« , dans les situations décrites précédemment, il est nécessaire d'interroger chaque lecteur ou base de données. A la réception des réponses, des filtres peuvent être appliqués ou les données peuvent être agrégées. Cette technique implique généralement une latence non négligeable et peut causer une surcharge du réseau et des lecteurs non nécessaire. En effet, on peut observer que :

- pour une requête de broadcast, seules les bases de données possédant des informations

concernant l'article recherché doivent être interrogées (les serveurs de *Paris* n'ont pas besoin de recevoir une requête de type *Lille*) ;
- une requête k-cast devrait s'arrêter une fois que k produits ont été trouvés ;
- une requête unicast peut être effectuée en n'interrogeant qu'un seul serveur.

Ces fonctionnalités sont celles offertes par SENSATION avec pour objectif d'assurer le passage à l'échelle et une haute qualité de service.

Une question de la même importance apportée par notre cas d'étude est celle de permettre un niveau personnalisable (ajustable) de tolérance aux pannes. Pour apporter plus de confiance aux données, il est nécessaire de prendre en compte un degré de redondance des informations, c'est-à-dire fournir un ou plusieurs serveurs alternatifs dans lesquels enregistrer les même données. Il faut alors décider où, quand, comment et combien de fois répliquer ces données. En effet, dans la majorité des cas, les données sont simplement répliquées dans un ou plusieurs serveurs de secours, qui sont utilisés lorsque le serveur initial est en échec. SENSATION n'adopte pas cette approche et réplique les données de manière efficace et intègre cette réplication dans son mécanisme de routage en routant les requêtes vers le serveur le plus proche possédant l'information recherchée et réduit ainsi la charge supplémentaire induite par le routage.

4.2 Présentation des systèmes à DHT utilisés

Afin de distribuer notre réseau, nous utilisons les DHT (présenté en section 2.6). Nous utiliserons ici SOLIST [38], un framework pour les réseaux de capteurs sans fil (Wireless Sensors Networks - WSN), ainsi que CAN [28] et Tribe [39], deux protocoles pair-à-pair (peer-to-peer - p2p) basés sur les DHT. Cette section présente ces trois outils nécessaires à SENSATION.

4.2.1 SOLIST

SOLIST [38] présente un réseau structuré en couches et fournit une série efficace d'opérations de type $*$-cast pour les réseaux de capteurs sans fil. SOLIST identifie une série de fonctionnalités basiques très largement répandues dans les applications distribuée et propose donc une implémentation efficace de cette série dans un réseau structuré en multi-couches. SOLIST n'est pas à proprement parler un système de pair-à-pair comme les systèmes présentés en section 2.6. En fait, SOLIST utilise les systèmes pair-à-pair afin de structurer son réseau multi-couches. En effet, le but ici est de donner un type à chaque capteur et de les grouper en fonction de ce type, qu'il soit statique (capteurs hétérogènes) ou dynamique (état de la batterie, données captées, *etc.*). Une couche est alors composée par tous les capteurs du même type. La figure 4.1 montre une projection des différentes couches d'un réseau SOLIST.

Afin d'entrer dans la couche du bon type t, ou d'effectuer une requête sur ce type t, un capteur devra contacter au moins un nœud de la couche correspondante. Pour trouver ce nœud le plus proche, SOLIST procède en deux étapes. La première consiste à contacter un *point d'entré* pour cette couche. Ce point d'entré n'appartient pas forcément au type recherché, mais il sera au courant du capteur le plus proche qui lui appartient à cette couche. La deuxième étape consiste alors à contacter ce nœud, appelé *nœud de contact*. Procéder en deux étapes permet de toujours atteindre le nœud le plus proche, limitant ainsi la consommation d'énergie, la bande passante et la latence. Pour identifier les *points d'entrés* ainsi que les *nœuds de contact*, SOLIST utilise des fonctions de hachages. En effet, les capteurs d'un même type appartiennent tous au même réseau pair-à-pair. Ainsi, on a un système p2p sur chacune des couches, ainsi que sur la couche principale.

L'efficacité des opérations $*$-cast dans un réseau SOLIST réside dans cette structure en couches. En effet, pour trouver le *nœud de contact*, étant dans une structure p2p, c'est une requête de

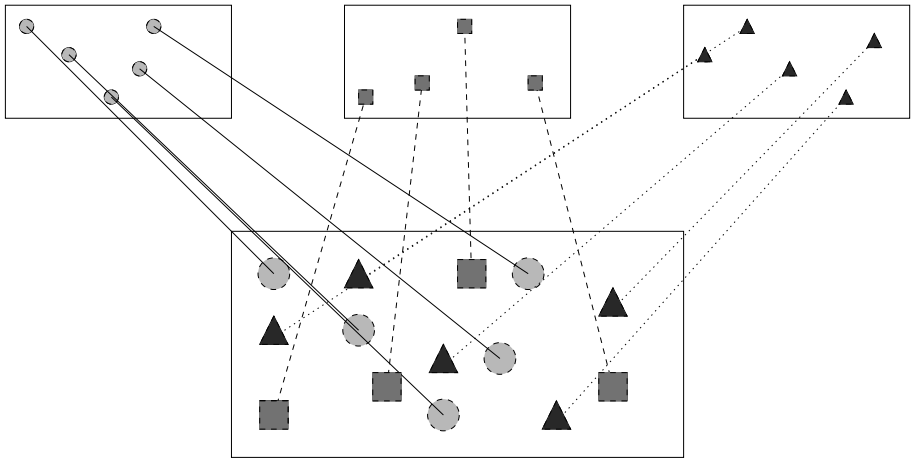


FIGURE 4.1 – *Projection de* 3 *groupes de nœuds (▲, ■ et ●) vers trois couches.*

recherche dans une DHT comme présentée avant. Une fois ce *nœud de contact* trouvé, la couche regroupant les capteurs du même type est aussi organisée aussi en un réseau p2p. Ainsi l'opération *unicast*, qui consiste à trouver un capteur du type t recherché est résolue lorsque l'on a trouvé le nœud de contact, celui-ci étant du type t. L'opération de *k-cast*, consistant à trouver k capteur du type t est alors effectuée dans la couche du nœud de contact. Ainsi, et c'est pareil pour les opérations de *broadcast* (diffusion vers tous les capteurs de type t), la requête n'est pas envoyée aux nœuds n'appartenant pas à ce réseau p2p, donc à cette couche, donc à ce type t. Elle n'est pas non plus traitée ou transférée par ces même nœuds. C'est ainsi que SOLIST offre cette gamme d'opérations $*$-cast efficace en terme de bande passante, de consommation d'énergie et de latence.

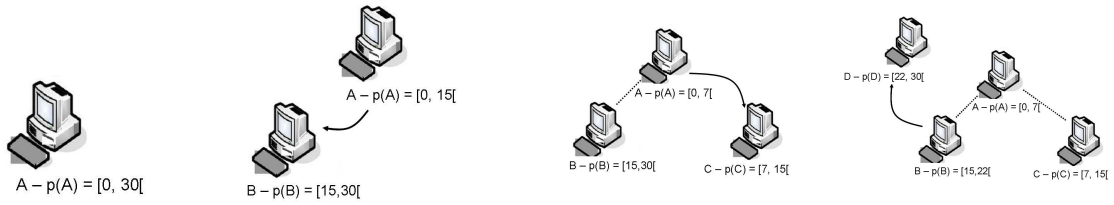
4.2.2 Tribe

Tribe [39] crée une topologie représentant un réseau logique, et décrit la localisation relative des nœuds en fonction de leurs voisinages dans le réseau physique.

Affectation des partitions : Quand un nœud arrive dans le réseau, il reçoit le contrôle d'une région qui va servir pour deux objectifs : l'identification des nœuds et le routage. Un nouveau nœud u dans le réseau reçoit le contrôle d'une région de la part de son voisin contrôlant la plus grande région. Ce dernier donne la moitié de sa région au nouveau nœud u et devient son nœud père. Tribe construit ainsi un arbre comme illustré par les liens entre les nœuds sur la figure **??**. Au début (Fig. 4.2(a)), le nœud A est tout seul et est donc responsable de tout l'espace virtuel $\mathcal{V} = [0, 30[$. Ensuite (Fig. 4.2(b)), le nœud B arrive et reçoit de A la moitié de la partition de A, donc de l'espace entier. Lorsque le nœud C apparaît, le nœud A donne encore une fois la moitié de sa région de contrôle (Fig. 4.2(c)). Enfin, quand le nœud D arrive, le nœud B partage sa partition car c'est lui qui possède la plus grande partition (Fig. 4.2(d)). Cela mène à une structure en arbre représentée par les liens entre les nœuds sur les figures.

Router au sein de la structure Tribe : Que ce soit dans le cas d'un enregistrement d'une nouvelle information ou dans celui de la recherche d'une information, une opération de routage doit être effectuée afin d'atteindre le nœud responsable de la clé retournée par la fonction de hachage. Soit $p(u) = [p_u^{\ominus}, p_u^{\oplus}[$ l'espace d'adressage contrôlé par le nœud u. Lorsque ce nœud u est arrivé dans le réseau, il a reçu un espace d'adressage $p_{init}(u) = [p_u^{\ominus}, p_{u_{init}}^{\oplus}[$ où $p_u^{\oplus} \leq p_{u_{init}}^{\oplus}$ étant donné que u peut avoir partagé sa région avec ses fils. Si ce nœud u souhaite contacter le nœud responsable de la clé *clé* $\in \mathcal{V}$, il doit procéder comme suit :

- si *clé* $\in p(u)$, u peut répondre à la requête, étant responsable de *clé* ;
- si *clé* $\notin p_{init}(u)$, alors u transfère la requête à son père ;



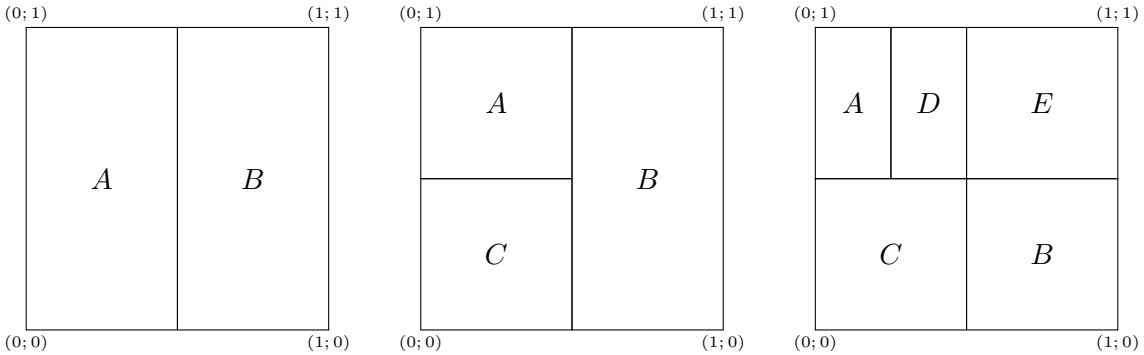
(a) (b) (c) (d)

FIGURE 4.2 – *Partage de l'espace d'adressage dans Tribe. (a) Le nœud A est seul et responsable de tout l'espace virtuel* $\mathcal{V} = [0, 30[$. *(b) Arrivée du nœud B et partage de la partition de A. (c) Apparition du nœud C et à nouveau partage de la partition de A. (d) Arrivée du nœud D et partage de la partition du nœud B.*

– si $p_u^{\oplus} < \textit{clé} \leq p_{u_{init}}^{\oplus}$, alors u transfère cette requête à son fils v tel que $\textit{clé} \in p_{init}(v)$.
Chaque nœud réitère ce processus jusqu'à atteindre le nœud responsable de la clé.

4.2.3 CAN

L'espace d'adressage de CAN [28] est un d-tore à coordonnées cartésiennes à d-dimensions. La figure 4.3(a) représente un réseau avec deux dimensions et deux nœuds (A et B). L'espace d'adressage a été divisé en deux, et chacun des nœuds est responsable de sa partition (*e.g.* le nœud A est responsable de la zone $[(0;0),(0,5;1)]$ et B de $[(0,5;0),(1;1)]$). Lorsqu'un nouveau nœud C entre dans le système, il choisit aléatoirement un point P de l'espace. Le nœud responsable de ce point va alors diviser sa zone de responsabilité et partager la moitié avec le nouveau nœud. Dans la figure 4.3(b), le nœud C choisit le point de coordonnées $(0,2;0,3)$. Ce point appartient à la zone du nœud A qui va alors partager sa zone avec C. Après l'insertion de C, celui-ci est responsable de la zone $[(0;0),(0,5;0,5)]$ tandis que la zone de A devient $[(0;0,5),(0,5;1)]$. Celle du nœud B ne change pas. La division de zone ici, dans un espace à deux dimensions, se fait d'abord selon l'axe X, puis Y et ainsi de suite. La figure 4.3(c) représente ce réseau à deux dimensions avec cinq nœuds dans le système.



(a) Réseau CAN à deux dimensions avec deux nœuds (b) Arrivée du nœud C et partage de la zone de A (c) Partitionnement après les arrivée de D puis E

FIGURE 4.3 – *Réseau p2p CAN avec deux, trois et cinq nœuds*

Afin de router les requêtes dans un tel réseau, les nœuds doivent connaître des informations sur les zones et les voisins. Deux nœuds sont voisins s'ils partagent une frontière commune. Dans la figure 4.3(c), le nœud E est voisin de B, D et A (l'espace virtuel est un tore), mais pas de

C. Chaque nœud va donc maintenir une table de routage contenant l' adresse IP et la zone de responsabilité de tous ces voisins. La table 4.4 présente la table de routage du nœud A de la figure 4.3(c). Lorsque ce nœud A cherche la clé $(0,7;0,2)$, il regarde d'abord si cette clé est dans

IP	**Zone virtuelle**
IP du nœud C	$[(0;0),(0,5;0,5)]$
IP du nœud D	$[(0,25;0,5),(0,5;1)]$
IP du nœud E	$[(0,5;0,5),(1;1)]$

FIGURE 4.4 – *Table de routage du nœud A dans le réseau CAN de la figure 4.3(c)*

sa zone. Si elle n'y est pas (ce qui est le cas dans notre exemple), le nœud va alors interroger sa table de routage. La clé qu'il cherche n'appartient à aucun de ses voisins (en effet, elle est dans la zone du nœud B), il va donc transmettre sa requête à un voisin. Le voisin retenu est celui dont les coordonnées virtuelles sont les plus proches de la clé cible, ici le nœud C. Le nœud A va donc transmettre sa requête au nœud C, qui va alors la transmettre à B, en accord avec la table de routage de C.

4.3 Description de SENSATION

4.3.1 Description générale

SENSATION a pour objectif de proposer une structure auto-organisée pour la gestion de dynamique des données au sein d'un réseau orienté gestion de stocks sur plusieurs sites géographiques. Il a pour objectif de répondre aux requêtes de gestion pouvant concerner une famille de produits ou une zone géographique.

Dans cette optique, dans SENSATION, les données se voient attribuer un *type* comme dans le framework SOLIST [38] pour réseaux de capteurs. Ce type peut être un type de produits (par exemple, *type=pantalons*) ou un lieu (*type=entrepôt A*). Nous ne faisons aucune distinction entre ces types. Pour traiter des requêtes concernant une famille de produits ou une zone géographique plus importante (plusieurs entrepôts par exemple), les requêtes sont envoyées à un ensemble de couches (chaque couche correspondant à un type). Par exemple, si une requête a pour objectif de faire l'inventaire des articles de sport, et que ces articles de sport incluent les vélos et les raquettes, des requêtes indépendantes seront envoyées à la couche « raquette » et à la couche « vélo » et les résultats seront ensuite agrégés.

De la même manière si une requête concerne chaque entrepôt de Lille, et qu'il y a deux entrepôts (A et B) à Lille, des requêtes distinctes sont envoyées à la couche entrepôt A et à la couche entrepôt B.

Chaque couche t est composée d'un réseau *overlay* (réseau abstrait) composé de nœuds et de liens et possédant un espace virtuel. Les nœuds sont les bases de données et les serveurs possédant des informations relatives au type t (ce sera par exemple chaque serveur ou EPCIS possédant des informations sur les pantalons). Au moment de rejoindre le réseau overlay d'une couche t, une base de données A se verra assigner une partie de l'espace virtuel ainsi que des « voisins », c'est-à-dire d'autres bases de données de la couche auxquelles A est connectée. Étant donné qu'il s'agit d'un réseau logique, les liens directs de ce réseau ne correspondent pas forcément à un lien physique direct. SENSATION utilise Tribe [39] pour attribuer une partie de l'espace aux bases de données et pour la création des liens. Tribe a été conçu pour les réseaux auto-organisés à large échelle. Il créé, sans entité centrale ni système de positionnement, une topologie représentant le réseau logique, et décrit les locations relatives des nœuds en fonction de leur voisinage dans le réseau physique. Il prévoit également un mécanisme de routage.

Bien que Tribe puisse être directement utilisé à l'intérieur d'une couche, l'utilisation d'une architecture « à la SOLIST » demande quelques adaptations.

Dans SENSATION l'espace doit être défini et les bases de données doivent connaitre certaines coordonnées leur permettant d'évoluer dans cet espace. Nous utilisons CAN à ces fins [28]. CAN introduit la notion d'espaces d'identification multi-dimensionnels qui améliore l'efficacité du routage par rapport au routage linéaire dans une seule dimension. Si l'espace est de 3 dimensions par exemple, chaque identifiant sera présenté sous forme de triplet.

Cette structure est ensuite utilisée dans le but d'atteindre trois objectifs :

- **stockage distribué :** chaque base de données possédant une nouvelle donnée de type t identifie dans un premier temps le nœud responsable des données de type t. Il peut, grâce à ce nœud, entrer dans la couche Tribe t pour ensuite contacter le nœud responsable de cette donnée en utilisant le mécanisme de routage de Tribe ;
- **réplication :** n'importe quel niveau de réplication peut être imposé à SENSATION. Une fois la couche de Tribe atteinte, appliquer simplement des fonctions de hachage différentes à l'identifiant de l'objet à enregistrer changerait le serveur responsable de cette donnée ;
- **gestion efficace des requêtes :** l'utilisation de couches permet de ne pas envoyer les requêtes à tout le réseau mais seulement à des nœuds concernés par cette requête. Des ressources sont économisées et la latence réduite. La première étape qui consiste à s'adresser à la bonne couche permet cette sélection. L'utilisation de Tribe au sein d'un overlay facilite la gestion des requêtes. En effet, la structure en arbre de Tribe est appropriée pour effectuer de l'agrégation de données d'une requête de dénombrement à chaque niveau de l'arbre et pour arrêter le flooding lorsqu'un nombre suffisant de nœuds est atteint.

À notre connaissance, SENSATION est le premier algorithme répondant a ces objectifs.

4.3.2 Mapping du système de coordonnées virtuelles

La structure virtuelle doit inclure tous les nœuds de tous les sites de stockage gérés par le réseau. Si l'on se place à une échelle nationale, les sites représentent les villes dans lesquelles un magasin est situé. La structure de recherche sera divisée en NC cellules, chacune d'elles correspondant à un site.

Nous attribuons à chaque cellule (ou ville par exemple) une coordonnée (X_C, Y_C) qui représente sa position relative dans le système virtuel selon les dimensions x et y. Plus précisément, la cellule située dans le coin inférieur gauche a les coordonnées $(0, 0)$, celles qui suivent ont les coordonnées $(1, 0)$ et $(0, 1)$. Chaque cellule est ensuite logiquement divisée en T zones carrées, où T représente le nombre maximum de catégories (qui pourraient être des types de produits ou de sites) supporté par l'application. Chaque carré représente une localisation virtuelle dans la cellule, avec la coordonnée relative (X_T, Y_T) à l'intérieur d'une cellule ; chaque location agit comme un point d'entrée pour un type différent dans la cellule logique (le carré de coordonnées (X_T, Y_T) de la cellule i et le carré de coordonnées (X_T, Y_T) dans la cellule j sont des points d'entrée de la même couche). La figure 4.5 illustre ce concept ; dans cet exemple la zone géographique est divisée en cellules virtuelles de tailles 3×3 et chaque cellule est divisée en 16 sites. Chaque site dans la cellule représente un point d'entrée différent pour les 16 types gérés. Cela signifie que l'application peut gérer jusqu'à 16 types, chaque type étant représenté par un entier entre 0 et 15. Pour des raisons de clarté et de simplicité, nous considérons le cas où les grilles virtuelles sont divisées par le même nombre de cellules dans les dimensions x et y. Cette hypothèse ne limite pas la possibilité d'utilisation de cette méthode à ces cas.

Par construction nous avons exactement T points d'entrée dans chaque cellule, une pour chacun des types gérés par l'application. Il peut y avoir moins de T points d'entrée. Si une position est vide, le serveur le plus proche assure le rôle de point d'entrée pour cette position vide. Cela signifie qu'à chaque action à effectuer (insertion, recherche,...) sur un objet de type

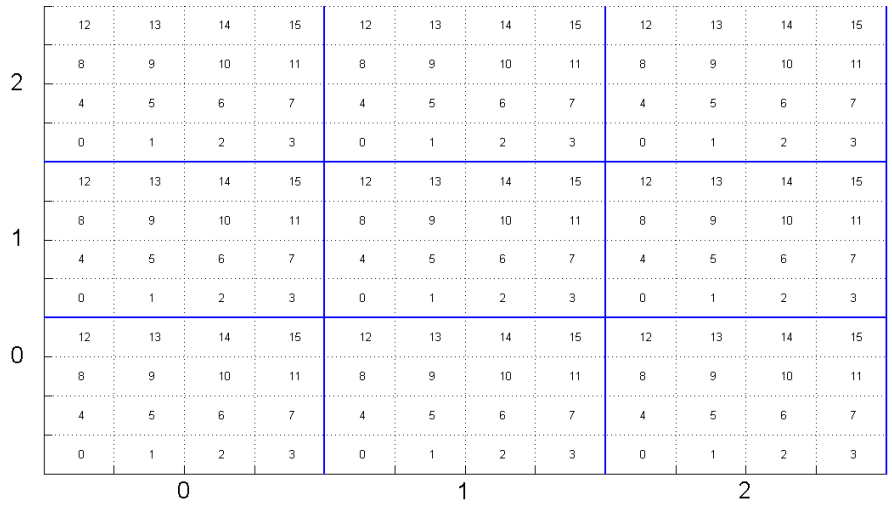


FIGURE 4.5 – *Division des cellules et points d'entré*

p, l'utilisateur u effectuant la requête doit d'abord atteindre le point d'entrée p de la cellule à laquelle il appartient. En contactant le serveur de base de données en charge du type requis p (qui est en fait le serveur en charge du point d'entrée p), u peut atteindre la couche p (nous disons que « u est dans la couche p ») pour ensuite pouvoir atteindre le serveur ayant effectivement enregistré l'objet en utilisant Tribe. Il est à noter que chaque point d'entrée d'un type donné se trouve dans la même position relative dans chaque cellule. De cette manière, selon la position d'où la requête est lancée, elle peut être acheminée au point de contact le plus proche. Le point critique concerne la méthode de partage de l'espace virtuel représenté par l'ensemble des points d'entrée. Nous proposons de partager l'ensemble des points d'entrée entre les serveurs d'une cellule en utilisant CAN. Par conséquent, au point d'équilibre, les serveurs d'une cellule partagent les points d'entrée de sorte que chacun d'eux soit en charge de partitions de la DHT qui ne superposent pas, la DHT étant l'ensemble des points d'entrée (cela constitue l'une des principales différences entre SOLIST et SENSATION).

Pour cela nous utilisons une formule de congruence pour la fonction de mapping afin de déterminer le point d'entrée de type p dans la cellule (X_C, Y_C) notée $ep_p^{(X_C,Y_C)}$ donné par :

$$\begin{aligned} ep_p^{(X_C,Y_C)} &= \left(Xep_p^{(X_C,Y_C)}, Yep_p^{(X_C,Y_C)}\right) \\ &= \left(mod(p,d) + X_C \cdot d, \left\lfloor \frac{p}{d} \right\rfloor + Y_C \cdot d\right) \end{aligned} \tag{4.1}$$

où $d = \sqrt{T}$, $\left\lfloor \frac{p}{d} \right\rfloor$ est la division entière entre p et d et $mod(p,d)$ est le reste de cette division. Une fois $ep_p^{(X_C,Y_C)}$ calculé, la requête peut être routée (en utilisant CAN) jusqu'au point d'entrée demandé p. Une fois la bonne couche atteinte, Tribe est utilisée pour trouver le nœud conservant l'item.

Attribution des coordonnées virtuelles :

Pour être intégrées à la structure, les bases de données ont besoin de coordonnées attribuées par l'utilisateur qui, de manière arbitraire, attribue à chaque nouveau nœud (base de données) une position géographique unique dans la grille. Un identifiant unique dans le système virtuel

est généré comme une séquence de bits contenant de l'information sur les coordonnées virtuelles (X_C, Y_C) et la position relative dans la cellule (X_T, Y_T) :

$$ID = |bin(X_C, nbC)|bin(Y_C, nbC)|bin(X_T, nbT)|bin(Y_T, nbT)|$$

où $bin(x, n)$ est la représentation sous forme d'entier de x sur n bits, $nbC = \frac{\lceil \log_2(NC) \rceil}{2}$, $nbT = \frac{\lceil \log_2(T) \rceil}{2}$ et $|$ représente une concaténation de chaines de caractères. Pour mieux appréhender ce passage d'un site virtuel à une adresse unique, considérons un exemple dans lequel un utilisateur connecte une nouvelle base de données au réseau aux coordonnées géographique $(7, 11)$ d'une grille à 9 cellules (ici, $NC = 9$). Si $T = 16$, l'identifiant virtuel a une taille de 8 bits : 2 bits sont utilisés pour chaque champ. En regardant la figure 4.6, il apparait clairement que les coordonnées $(7, 11)$ appartiennent à la cellule $(1, 2)$ et au carré $(3, 3)$ de cette cellule. Par conséquent, l'identifiant $ID = 01|10|11|11$ (01 étant la représentation binaire de X_C, 10 la représentation binaire de Y_C et 11 la représentation binaire de X_T et Y_T sur 2 bits chacun).

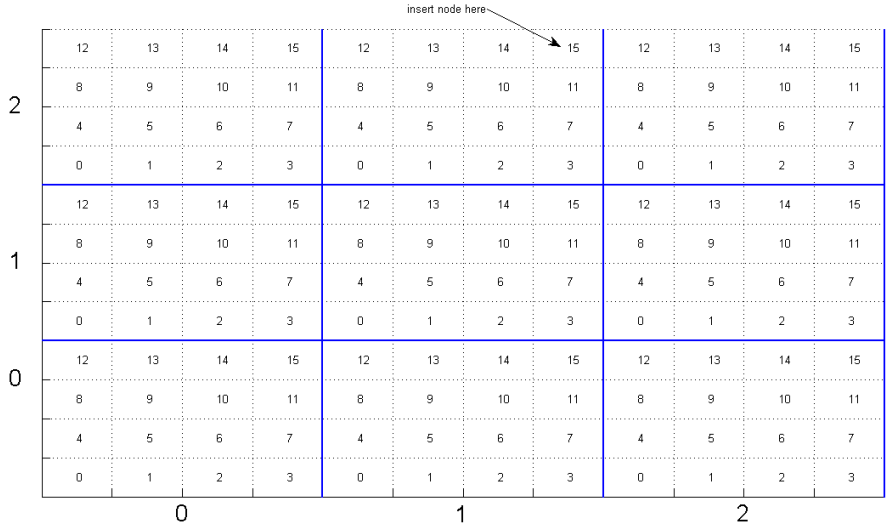


FIGURE 4.6 – *Ajout d'un nœud. Les coordonnées des cellules sont reportées en bas et sur la gauche des dimensions x et y respectivement.*

Enfin, il est intéressant de remarquer que chaque location dans chaque cellule peut être ou ne pas être occupée par un serveur. Par contre, chaque point d'entrée qu'une location représente appartient à un serveur, qui par ailleurs est le serveur CAN gérant le point d'entrée. En effet, les requêtes d'accès à la couche Tribe seront adressées au nœud responsable.

Ajout d'un EPCIS :

Lorsqu'un serveur souhaite rejoindre l'infrastructure il doit :

1. obtenir une portion de l'ensemble des points d'entrée (c'est-à-dire rejoindre CAN) et ;
2. obtenir une partie de l'espace d'adressage de Tribe dans toutes les couches obtenues par CAN et correspondant aux objets qu'il doit enregistrer (c'est-à-dire rejoindre Tribe).

Par exemple, considérons un serveur u souhaitant rejoindre l'infrastructure. Le nœud CAN le plus proche (noté v) déjà dans la structure doit donner à u une part des points d'entrée dont il avait la charge jusqu'à présent. Supposons par exemple que u reçoive l'ensemble des points d'entrée $P = 1, 2, 7, 8$. Cela signifie que u sera le point d'entrée pour toutes les requêtes concernant les

couches $1, 2, 7, 8$ au sein de la cellule, de sorte que u doit aussi rejoindre les couches correspondantes dans Tribe. En d'autres termes, il doit également obtenir une portion de l'espace d'adressage dans Tribe dans chacune de ces couches.

Ajout items :

Lorsqu'un item de type i doit être enregistré dans l'infrastructure, le serveur qui gère actuellement le point d'entrée i est contacté en utilisant CAN (cette action est effectuée localement à la cellule puisque chaque cellule contient un nœud en charge du type i). Étant donnée que ce serveur possède également un accès à la couche i dans Tribe, la requête peut être acheminée en utilisant Tribe. Lorsque la couche Tribe correspondante est atteinte, n'importe quel niveau de réplication de données peut être imposé pour augmenter la tolérance aux pannes. En appliquant des fonctions de hachage différentes à un même identifiant, des serveurs différents géreront les identifiants hachés. La seule contrainte à imposer pour avoir un niveau de réplication de h est que chaque serveur ait connaissance de cet ensemble de fonctions de hachage. Cette technique permet d'obtenir un niveau personnalisable de tolérance aux pannes. De plus, elle laisse le choix à l'application de réduire le coût induit par l'utilisation de Tribe lors des requêtes. Ce mécanisme est mis en évidence dans le paragraphe suivant.

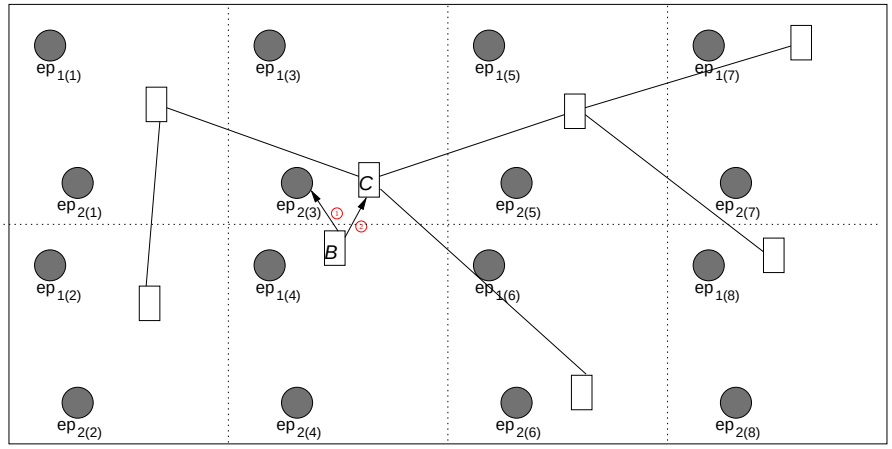
Phase de recherche :

Lorsqu'une requête est lancée, le nœud en charge du type sur lequel s'applique la requête est contacté en utilisant CAN. Cette étape est la même que celle de la phase d'ajout d'un item. Une fois situé dans la couche, le type de la requête détermine les actions effectuées. En particulier, pour l'unicast, la requête est routée en utilisant Tribe jusqu'à ce qu'un serveur gérant l'identifiant haché soit atteint. Ce serveur est en charge de fournir directement à l'utilisateur ayant lancé la requête la réponse correspondante. Lorsqu'un niveau de réplication h est imposé, nous savons que l'item (s'il est présent) a été enregistré h fois dans la couche. Par conséquent, il est moins coûteux (d'un point de vue latence et overhead) de calculer h fois la fonction de hachage et d'atteindre le nœud Tribe le moins éloigné d'un nœud de contact. Cette fonctionnalité de SENSATION est analysée dans les résultats de simulation.

4.3.3 Exemple

Supposons que l'on souhaite faire, à partir de la base de données B, l'inventaire des pantalons d'une entreprise dont les stocks sont répartis sur plusieurs sites. Pour répondre à cette requête en utilisant SENSATION, B contacte dans un premier temps le point d'entrée le plus proche de type « pantalons ». Pour identifier ce dernier, B utilise l'équation 4.1 et CAN (qui lui donne la location relative du point d'entrée correspondant à la couche « pantalons » de chaque cellule). Il calcule le plus proche ep_t. B contacte ensuite ep_t et récupère la localisation du nœud le plus proche C gérant les pantalons, appelé ici *nœud de contact*. Le nœud ep_t connait le nœud C puisque ce dernier s'est enregistré au nœud ep_t en utilisant également l'équation 4.1. La requête de dénombrement est ensuite envoyée à C. Puisque C a préalablement rejoint la couche correspondant aux pantalons, il peut transmettre le message à tous les nœuds gérant des pantalons et s'étant enregistrés dans cette couche. La requête de dénombrement sera envoyée en broadcast dans l'overlay et la réponse contiendra le nombre de pantalons de tout le réseau.

Considérons maintenant une autre requête consistant à savoir si un produit existe (ici, un pantalon). Comme dans la situation précédente, B contacte le point d'entrée le plus proche de type « pantalons ». Si le point d'entrée retourne l'identifiant d'un nœud, l'existence de tels produits est avérée. Dans le cas contraire, le point d'entrée retourne *NOT FOUND* et il n'y a



① Node B computes the coordinates of the nearest entry point ($ep_{2(3)}$) which returns the nearest node (C) identifier and coordinates belonging to the layer
② Node B contacts node C, which is the nearest node in the ligh−t−layer

FIGURE 4.7 – *Nœud B retrouvant le nœud du* LIGH-*trousers*-LAYER *le plus proche.*

aucun pantalon. Si un utilisateur veut savoir s'il y a au moins 10 pantalons pour confirmer une commande, il commence également par contacter le point d'entrée pour le type « pantalons », qui retournera l'identité de contact C. Il envoie ensuite une requête k-cast ($k = 10$) à C, qui transmet la requête dans l'overlay, en décrémentant k du nombre de pantalons qu'il gère, au nœud suivant de la même couche en suivant l'algorithme de diffusion. Le nœud suivant répète la même opération (décrémenter k et transmettre la requête avec la valeur de k mise à jour) jusqu'à ce que la mise à jour de k donne $k = 0$. Lorsque l'utilisateur reçoit une réponse, il sait s'il existe assez de pantalons sans inonder le réseau. Ce mécanisme permet de limiter le nombre de nœuds recevant la requête par rapport à un modèle où la requête serait envoyée à tous les nœuds puis les résultats interprétés. Une requête peut également concerner la localisation géographique. Dans un tel cas, le même mécanisme est appliqué au type « Lille » si la requête vise à effectuer l'inventaire de l'ensemble des entrepôts de Lille.

Tribe constitue donc une structure unique et efficace pouvant être utilisés pour le stockage distribué et la réplication de données ainsi le traitement de requêtes. Dans la section suivante, nous présentons les résultats d'évaluation des performances par simulation de SENSATION.

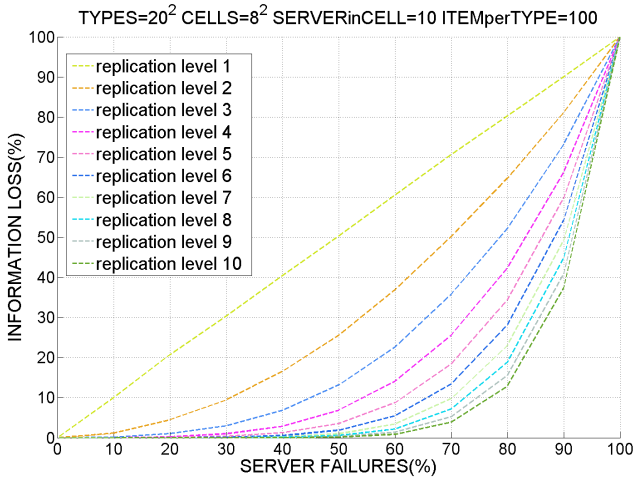
4.4 SIMULATIONS

Une analyse exhaustive des performances est effectuée afin d'évaluer le coût des requêtes de routage, la robustesse à différents niveaux de réplication ainsi que la distribution de l'information dans SENSATION [1].

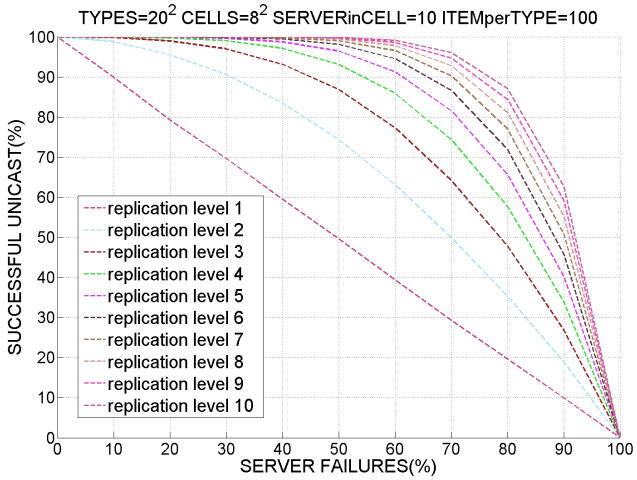
Conditions d'évaluation :

SENSATION est évalué par simulation en utilisant le logiciel MatLab. Dans chacune des cellules, 10 serveurs sont positionnés aléatoirement dans un espace bi-dimensionel divisés en 63 cellules (640 serveurs au total) de tailles égales. Après avoir insérés les serveurs, des objets sont ajoutés de la manière suivante : 100 objets de type i ($i \in [1, 400]$) sont insérés dans des serveurs choisis

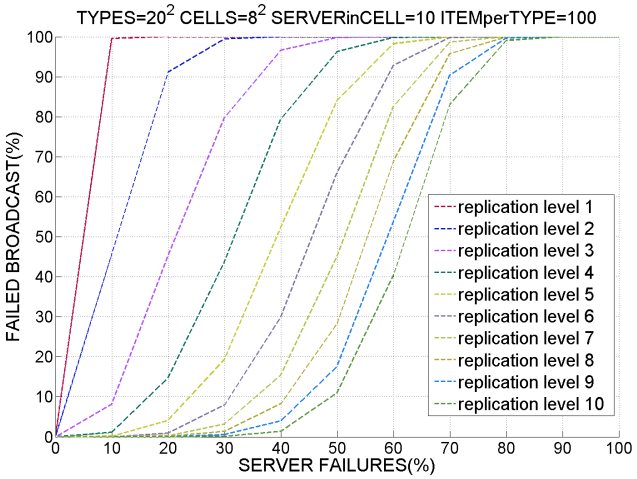
1. SENSATION étant à notre connaissance le seul algorithme de son genre, nous évaluons ces performances dans l'absolu sans pouvoir le comparer à ses concurrents.



(a) Pourcentage d'information perdues.



(b) Reques *unicast* réussies.



(c) Pourcentage de requêtes *broadcast* echouées.

FIGURE 4.8 – *Performances de SENSATION pour différents niveaux de réplication dans Tribe et pourcentages de serveurs défaillants.*

aléatoirement. Chaque expérimentation est effectuée 30 fois et les résultats sont donnés sous forme de moyenne.

Overhead :

Après la phase d'enregistrement, chaque objet enregistré fait l'objet d'une requête lancée par un serveur. Cette action (qui consiste en fait en ce que nous avons appelé une requête unicast) dans un réseau classique de base de données ne devrait avoir de coût, puisque l'information est enregistrée dans le serveur local. A l'inverse l'utilisation de SENSATION oblige à contacter le point d'entrée correspondant (routage avec CAN). Une fois le nœud en charge contacté, Tribe est utilisé pour localiser le serveur possédant effectivement l'objet requis. La figure 4.9 montre la charge supplémentaire introduite par CAN pour atteindre les points d'entrée nécessaires. Pour des raisons de lisibilité, les résultats sont présentés sous forme d'histogramme. On peut aisément constater que 35% des points d'entrée peuvent être contactés en moins de 2 sauts, ce qui correspond à overhead faible. Il est à noter que cet overhead est indépendant du niveau réplication choisi dans les couches de Tribe.

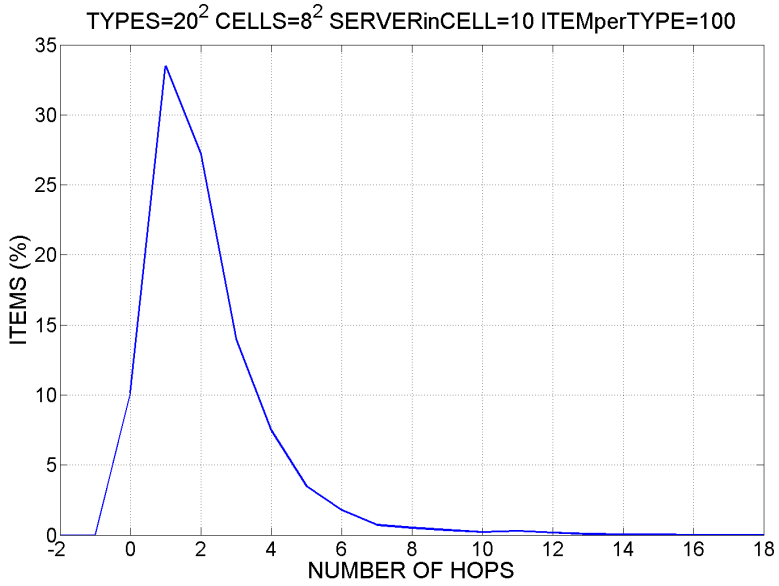


FIGURE 4.9 – *Overhead dans CAN.*

Gestion des défaillances :

Afin d'évaluer la robustesse de SENSATION, un pourcentage de serveurs défaillants allant de 0 à 100% est considéré. Lorsqu'un serveur subit une défaillance, les objets qui y sont enregistrés sont indisponibles (il s'agit d'une défaillance de base de données), mais le routage dans cette couche est toujours possible. Cela signifie que durant une défaillance l'information n'est pas disponible mais CAN et Tribe peuvent réagir à cette défaillance et s'auto-organiser à nouveau. La figure 4.8(a) montre le pourcentage d'informations perdues pour différents niveaux de réplication et différents modes de perte (cela revient à donner le taux d'échec des requêtes unicast). On peut remarquer qu'avec un niveau de réplication de 1, la perte d'information est linéaire. En effet, les informations sont uniformément distribuées tout comme le nombre de défaillances. De plus, comme attendu, l'élévation du niveau de réplication dans les couches de Tribe entraine un accroissement de la fiabilité de SENSATION. Avec un faible taux de réplication de 2, seules 25% des données sont

perdues lorsque 50% des serveurs sont défaillants. Le même taux de perte est obtenu lorsque 70% des serveurs sont défaillants et un niveau de réplication de 3. Il est également à noter que cela reflète le pire scénario puisque dans ces simulations, les défaillances apparaissent de manière aléatoire alors que dans un scénario réaliste, lorsqu'un serveur est défaillant, les serveurs les plus proches sont plus susceptibles d'être également défaillants que d'autres serveurs du réseau. Étant donné que les données sont uniformément réparties, dans un scénario réaliste, la perte serait moins conséquente que dans nos simulations.

Performance des requêtes :

La figure 4.8(b) représente le pourcentage de requêtes unicast réussies, pour différents niveaux de réplication et de pertes. Les résultats montrent que les requêtes unicast sont plutôt bien gérées par SENSATION avec un niveau de réplication suffisamment faible et cela même avec des taux de perte relativement élevés. Fixons un niveau de réplication de k. L'information concernant un objet est perdue si et seulement si les k serveurs conservant cette information sont simultanément défaillants. Dans le cas contraire, suite à une réorganisation, l'objet est enregistré de nouveau. Les pertes sont temporaires. Malheureusement, le même comportement n'est pas attendu avec les requêtes de diffusion. En effet une requête de diffusion ne peut être correctement effectuée que si, et seulement si, tous les objets du type requis sont présents dans un des k serveurs dans lequel il est enregistré. Par conséquent, la simple disparition d'un des objets d'un type donné (l'objet pour lequel la requête unicast a échoué) entraine un échec de la requête de diffusion. Cependant, comme illustré par la figure 4.8(c), SENSATION résiste à ces échecs.

4.5 Conclusion du chapitre

Dans ce chapitre, nous avons présenté SENSATION, une architecture auto-organisée permettant la gestion de larges stocks. Cette architecture présentent trois avantages. Elle facilite le stockage distribué des données en utilisant la notion de couche, chaque couche correspondant à un type de données. Elle permet de personnaliser le niveau de réplication des données par l'utilisation de Tribe et de fonctions de hachage distinctes appliquées à l'identifiant de l'objet à enregistrer. Enfin, cette architecture propose une gestion optimisée des requêtes qui ne sont plus envoyées à tout le réseau mais seulement aux nœuds susceptibles d'y répondre, c'est-à-dire aux nœuds appartenant à la couche visée. L'utilisation de Tribe au sein de chaque overlay permet ensuite de router efficacement ces requêtes. Ces fonctionnalités facilitent la gestion distribuée des bases de données et des EPCIS d'entrepôts de stockage, qui ont tendance à être de taille de plus en plus importante et à être de plus en plus interconnectés. Les résultats de simulation montrent que SENSATION est efficace en terme de fiabilité et de passage à l'échelle. En effet, nous avons montré que l'overhead associé à l'utilisation de SENSATION par rapport à l'utilisation d'un système de base de données local est relativement faible (un tiers des points d'entrée peuvent être contactés en moins de 2 sauts indépendamment du niveau de réplication choisi). La robustesse de notre proposition, évaluée en considérant un taux variable de serveurs défaillants, est, elle, dépendante du niveau de réplication choisi. Ainsi, lorsque la moitié des serveurs sont défaillants, seuls un quart des informations enregistrées sont indisponibles. Cependant, comme nous l'avons expliqué précédemment, ces résultats de simulation ne donnent qu'un aperçu des performances attendues par SENSATION. En effet, dans un environnement réel, les pannes ne sont pas uniformément réparties dans le réseau et ont plutôt tendance à être concentrées dans une zone restreinte de ce réseau. Il serait dont intéressant d'évaluer les performances de SENSATION lors d'un déploiement réel.

Publications

1. Conférences internationales :
   - *A k-layer self-organizing structure for product management in stock-based networks.* A. Carneiro Viana, N. Mitton, L. Schmidt, M. Vecchio. In IEEE ICEBE, Shangaï, China, 2010. IEEE Computer Society. *to appear*
2. Rapport de recherche :
   - *SElf-orgaNizing Structures for mAnagemenT In stock Oriented Networks.* A. Carneiro Viana, N. Mitton, L. Schmidt, M. Vecchio. RR-7192, INRIA, 2009

ONS Multi-Racines

5

Allant vers des environnements où l'intelligence ambiante est de plus en plus présente, où la plupart des objets sont connectés à des réseaux ubiquitaires, l'interopérabilité inter-entreprise devient un besoin essentiel. Les réseaux complexes, composés d'un large nombre de différents types d'objet, formeront le fameux Internet des Objets [2] (Internet of Things - IoT). Parmi ces réseaux, l'Internet des Biens gèrera les échanges d'information dans un monde où les entreprises seraient interconnectées [3]. L'architecture supportant ce passage à l'échelle devrait être pensée en prenant en compte un modèle de gouvernance ouverte.

Les raisons pour le développement d'un modèle de gouvernance ouverte ont été fortement mises en avant à un niveau politique mais cela devrait aussi être une opportunité pour les entreprises de penser aux innovations technologiques et d'usage, mais aussi aux besoins en sécurité, stabilité et performances. Les limitations et les faiblesses de l'architecture actuelle de l'Internet encouragent en effet à étudier prudemment les approches architecturales à partir de zéro pour suivre un modèle de gouvernance ouvert pour le futur Internet des Objets.

Le réseau à large échelle d'EPCGlobal sera une partie de cet IoT. Un de ses composants standardisés important est le service d'annuaire centralisé pour les objets, appelé Service de Nommage des Objets [10] (Object Name Service - ONS) et basé sur les DNS (Domain Name System) [21]. Étant donné l'importance du système d'ONS dans un futur proche, il serait avisé de prendre le temps de développer des solutions alternatives et de les comparer avec l'architecture en titre, étant donné qu'il est encore possible aujourd'hui d'influencer l'évolution des réseaux.

Depuis le sommet mondial sur la société de l'information (Tunis, 2005) et les négociations sur la gouvernance de l'Internet, la conscience Européenne sur ce sujet s'est élevée dramatiquement. La conception architecturale de l'Internet des Objets est en conséquence considérée, à raison ou à tort, comme un problème politique qui doit être négocié d'un point de vue de la gouvernance avant d'être implémentée. En particulier, les applications de logistique qui peuvent impliquer la localisation de produit (*e.g.* drogue, armes, déchets nucléaires), les applications sensibles (rappel de produits, applications d'anti-contrefaçon, contrôle de sécurité alimentaire) sont considérés comme des applications critiques, tout comme les architectures qui les supportent. Dans cette perspective, la Commission Européenne travaille actuellement sur un avant-projet de communications sur les réseaux futurs et l'Internet, les premiers défis concernant l'Internet des Objets en mettant l'accent sur le besoin d'un modèle de gouvernance ouverte de cette architecture.

D'un autre coté, l'architecture actuelle de l'Internet recevait de plus en plus de critiques tandis que le nombre croissant d'utilisateurs et de nouveaux usages était accompagné par de plus en plus de failles de sécurité (envoi massif de publicités, faille de Kaminsky, vers). On parle donc de plus en plus de repartir de zéro pour la nouvelle génération de l'Internet.

Pour toutes ces raisons, une seule approche pour l'architecture de l'Internet des Objet qui serait basée sur les standards actuels de l'Internet doit être considérée soigneusement. Une approche avisée suggérerait de comparer l'architecture actuelle avec des modèles alternatifs. Tandis que l'architecture courante pourrait probablement engendrer une importante économie en utilisant les

technologies déjà répandues, les approches alternatives pourraient être l'opportunité d'adresser aussi bien les problèmes de gouvernance que de sécurité.

Dans ce chapitre, nous commençons par présenter dans une première section le problème et les défis d'un ONS multi-racines. Nous présenterons ensuite, dans la section 5.2, une solution basé sur les DHT [40]. La section 5.3 discute de certains points améliorant la solution.

5.1 Définition du problème

Depuis le lancement des standards EPCglobal en 2001, de plus en plus d'entreprises ont commencé à explorer les possibilités de ces technologies, services et interfaces tels que l'EPCIS, qui représente la première étape vers l'usage du réseau EPCGlobal. Fondations d'un type de connectivité qui augmentera la visibilité tout au long de la chaîne d'approvisonnement et aidera à suivre les cargaisons, combattra l'introduction de produits contrefaits et préviendra les problèmes de stock vide des distributeurs, ces standards font apparaître de nouvelles et nombreuses classes d'applications. Mais pour comprendre le besoin d'évolution du réseau EPCGlobal, nous devons nous souvenir que, au début, il était prédominé par les besoins des compagnies alimentaires et des distributeurs. Par conséquent, l'architecture du réseau EPCGlobal actuel est très focalisé sur les besoins de ces scénarii métiers. Cependant, dans la course au gain de temps, les entreprises commencent à adopter la RFID au delà de la chaîne d'approvisionnement et au travers de petits deploiements sporadiques à petite échelle, impliquant un accroissement des industries de secteurs très variés telles que la santé, l'aérospacial, l'automobile, la défense, *etc.*.

Donc la prochaine étape de développement du réseau EPCGlobal sera de permettre une intégration flexible des informations sur des produits fournis par les organisations horizontalement à travers la chaîne d'approvisionnement, et verticalement à travers plusieurs industries. Ce mouvement d'activités petites et localisées vers de larges réseaux inter-entreprises et inter-nations requièreront des jeux de données plus complets et plus compréhensibles. Cela implique une synchronisation efficace des données, une garantie de disponibilité et une sécurité accrue de ces données. Il y a donc en définitive un besoin d'un équilibrage des données et d'une évolution des standards, incluant celui dû-dit Service de Nommage des Objects.

Comme expliqué dans la section 2.5, le Services de Nommage des Objects (Object Name Service - ONS), définit l'interface de services de recherche en fournissant des liens quasi-permanents ou relativement statiques entre l'identité d'une entreprise responsable d'un objet (souvent le fabriquant) et le services d'information que l'entreprise fourni. Cette entreprise possède des bases de données contenant des données relative aux objets. Ces informations sont gérées par les Services d'Information EPC (EPC Information Services - EPCIS [9]) des partenaires du réseau.

Basé sur le protocole Internet DNS, l'ONS est un modèle client/serveur organisé de façon hiérarchique, offrant la possibilité d'orienter les requêtes venant des applications clientes jusqu'à l'EPCIS de la bonne entreprise.

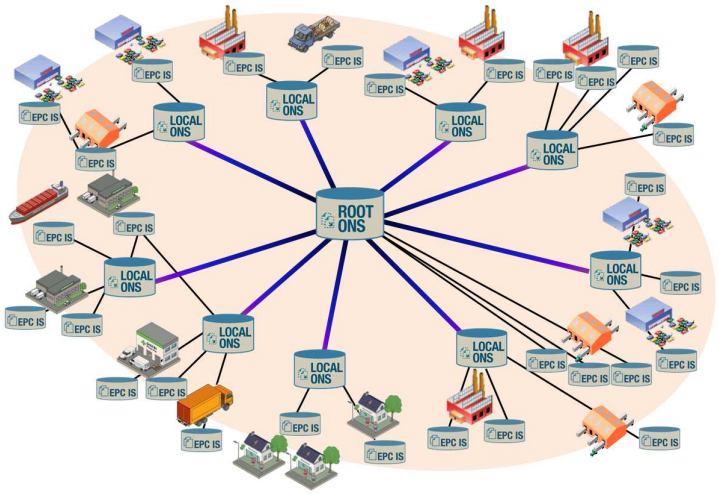
Le standard, dans sa version 1.0.1 [10] conçoit le réseau EPC comme un système d'information global, centralisé, dans lequel de nombreux ONS locaux sont inter-connectés. Pour répondre aux requêtes des applications en les dirigeant vers le bon ONS local, un domaine racine est implémenté (onsepc.com). Ce domaine racine, ou ONS racine, est le coeur du service, structurant le réseau et localisant les services associés. C'est une racine unique et autoritaire, il possède les références de tous les ONS locaux du réseau EPCGlobal.

5.1.1 Défis

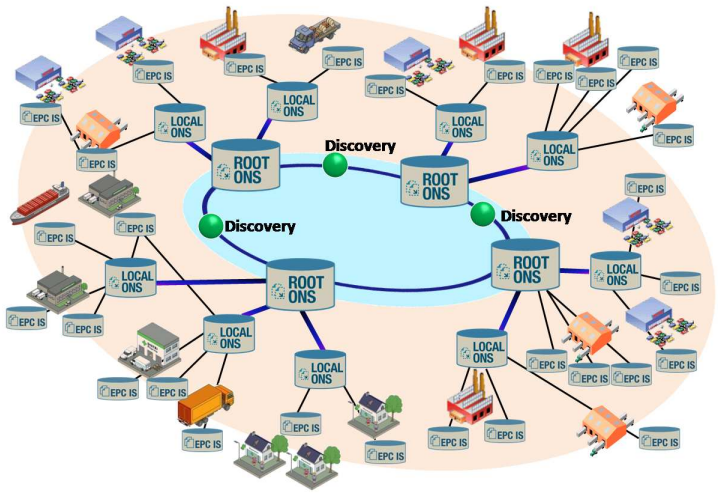
GS1 France a lancé son propre ONS racine afin de répondre au problème géopolitique posé par l'unicité et l'autorité d'un unique ONS racine et pour élaborer les concepts théoriques des

composants du réseau EPCGlobal dans un environnement réel. D'autres régions du monde pensent aussi à avoir leur propre ONS.

L'émergence de plusieurs racines souligne le besoin pour GS1 de faire évoluer son standard vers un modèle architectural symétrique, comme illustré par la figure 5.1.



(a) ONS (existant) centralisé.



(b) ONS multi-racines.

FIGURE 5.1 – *ONS Centralisé Vs Distribué.*

C'est pourquoi, pour permettre aux nombreuses organisations l'adoption et la standardisation à l'échelle mondiale des technologies EPC d'une façon éthique et responsable, cette augmentation en taille pour un réseau demande aussi des efforts de développement d'un modèle de gouvernance ouverte. Par la suite, ce modèle de gouvernance ouverte pourra être étendu pour incorporer plusieurs services ONS d'autre partie du monde, et aussi bien les aspects techniques que métiers seraient administrés sous un série communes de règles. Extraite de l'ONS racine géré par GS1 France, une série de règles pour la gouvernance, incluant par exemple des standards pour les questions de nommage et de découvertes de services associés à des outils de sécurité tels que les certificats d'autorité, les gestions de la vie privée, *etc.* doivent être explorés. En outre, le but de cette plateforme est de donner à la Communauté Européenne un rôle de leader dans le

développement de l'intelligence ambiante dans la chaîne d'approvisionnement et ainsi augmenter la compétitivité à travers la direction d'une implémentation de réseaux d'entreprises ouverts.

C'est dans ce cadre qu'un projet financé par l'association nationale française de la recherche (ANR) a débuté. Avec des partenaires comme l'INRIA, le LIP6 [1], le GREYC [2], l'AFNIC [3] et Orange Lab., le but de GS1 France dans ce projet appelé WINGS [41] est d'explorer différentes pistes pour le passage à l'échelle du service ONS, tout en assurant une gouvernance ouverte.

5.1.2 ONS Multi-racines

Le problème de la distribution de l'ONS ne concerne pas seulement le passage à l'échelle. En effet, on a souligné un réel problème géopolitique sur la responsabilité et l'autorité d'une unique racine ONS. Il faut donc réussir à passer l'ONS à l'échelle tout en supprimant cette unique racine. Ainsi, nous aurons répondu aux deux problèmes majeurs. C'est ce que l'on appelle l'ONS Multi-racines. Les ONS de plus haut niveaux ont tous le même rôle, le même pouvoir, les mêmes dépendances.

Afin de répondre à ces attentes, Evdokimov *et al.* [42] proposent des réplications de l'ONS. Ils décrivent d'abord l'approche simple afin de supprimer cette unique racine l'ONS. Il suffirait en effet à une entité responsable comme EPCGlobal de publier une version du fichier contenant tous les enregistrements de l'ONS. Les différents ONS n'auraient plus qu'à se synchroniser avec ce service fourni par EPCGlobal. Chacun de ces serveurs racines pourraient être configuré afin de répondre aux requêtes provenant d'une certaines plages d'adresses IP afin de réduire la quantité de requêtes à traiter. Afin de déterminer la faisabilité de cette approche, les auteurs ont calculé approximativement la taille des données devant être répliquées et synchronisées, et avec le développement de la technologie RFID, le nombre d'enregistrement va augmenter rapidement. Il faut en effet prévoir que chacune des entreprises enregistrées chez EPCGlobal va probablement avoir son service d'EPCIS. Ainsi la taille de ce fichier de zone racine de l'ONS va rapidement grossir. Evdokimov *et al.* introduisent donc le concept d'ONS Multipolaire Régional (Regional MONS). En utilisant les informations contenues dans l'identifiant EPC, et surtout le préfixe de la compagnie, on peut géographiquement répartir les identifiants EPC. En effet, comme expliqué dans la section 2.2, les trois premiers digits du préfixe servent à identifier le pays d'appartenance du membre GS1 (*e.g.* les codes 300-379 sont réservés pour la France [4]). Le fichier des enregistrements de l'ONS serait alors divisé en plusieurs. Le fichier de zone de chaque MONS régional contiendrait tous les enregistrements des entreprises appartenant à cette région. La figure 5.2 montre les différentes étapes d'une opération de résolution de nom. Lorsqu'une entreprise cherche des informations sur un identifiant EPC, elle va commencer par interroger son MONS régional (flèche ❶). Si l'entreprise fabriquant l'objet appartient à la même région que celle qui fait la requête, la réponse du MONS régional est directement l'adresse de l'ONS local du fabriquant du produit que l'entreprise va alors pouvoir interroger (cas classique de l'ONS). Si en revanche les régions sont différentes, alors la requête est d'abord transmise au MONS régional du fabriquant (flèche ❷) avant de renvoyer l'adresse de l'ONS local qu'on peut interroger (flèche ❸).

Dans la section suivante, nous présentons notre solutions aux problèmes de passage à l'échelle et d'autorité du système d'ONS.

5.2 ONS Multi-racines basé sur les DHT

Dans cette section, nous proposons une solution au problème de l'ONS multi-racines. L'idée de base derrière la solution repose sur le concept de tables de hachage distribuées (Distributed Hash

1. Laboratoire d'Informatique de Paris VI
2. Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen
3. Association Française pour le Nommage Internet en Coopération
4. http ://www.gs1.org/barcodes/support/prefix_list

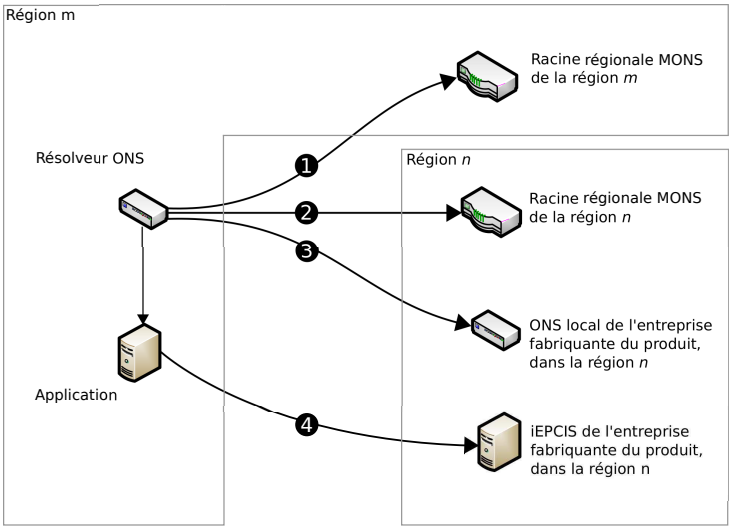


FIGURE 5.2 – *Recherche d'un EPCIS à travers deux MONS régionaux*

Tables - DHT). La raison d'un tel choix est que les DHT rassemblent assez de propriétés pour couvrir l'ensemble des besoins initialement imposés par GS1 et présentés dans la section 5.1.

5.2.1 Problème de l'ONS distribué et lien avec les principes des DHT

Dans la section 2.5, nous avons présenté le service de recherche traditionnel de l'ONS. La figure 2.8 représente le fonctionnement de l'ONS, où une entreprise interroge l'ONS racine afin de retrouver l'EPCIS contenant les informations sur un identifiant EPC donné. Les principes des DHT répondent clairement aux besoins d'un ONS distribué multi-racines. Plus spécifiquement :

- Les ONS racines sont les nœuds de la DHT qui gère l'espace d'adressage.
- Les identifiants EPC sont les clés du service de localisation. Ils sont liés à une position dans l'espace d'adressage. L'ONS racine qui gère cette position de l'espace possède un pointeur vers l'ONS local responsable effectivement de la localisation de cet identifiant.

La figure 5.3 illustre l'utilisation de la DHT dans le service de recherche modifié de l'ONS. Les nœuds racines servent ici de collection virtuelle de pointeurs. Lorsque l'on contacte un nœud ONS-racine afin de retrouver les informations concernant un EPC, ce nœud utilise la DHT afin de connaitre l'ONS racine responsable de l'espace d'adressage contenant l'identifiant. Ce principe présente deux avantages. D'abord, il permet au système d'être complètement distribué et indépendant du système sous-jacent. Ensuite, l'information réelle de localisation est gardée en locale. Basiquement, nous utiliserons deux fonctions de hachage.

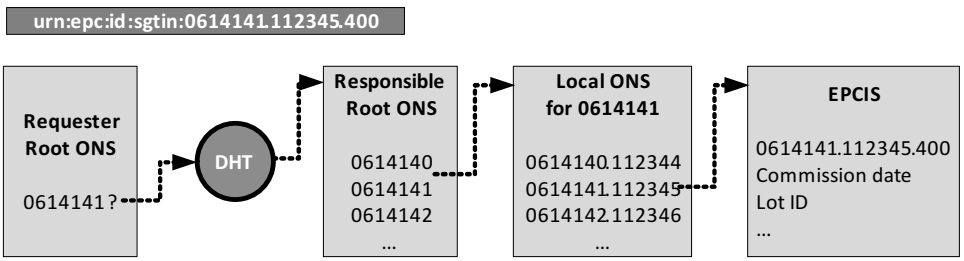


FIGURE 5.3 – *Service de recherche utilisant la DHT. L'ONS racine responsable n'est pas nécessairement l'ONS référent de l'ONS local mais peut être n'importe lequel des ONS racines.*

Avant d'entrer dans les détails du système, il est important de souligner les conditions nécessaires pour qu'il fonctionne.

- Les deux fonctions de hachage doivent être les mêmes pour tous les nœuds participant au système. Cette condition est nécessaire pour la consistance.
- L'espace d'adressage doit être bien défini et devrait être le même pour les deux fonctions de hachage.
- La même fonction de hachage doit être utilisée aussi bien pour enregistrer un contenu que pour en rechercher un.

5.2.2 Structure de l'espace d'adressage

Nous proposons de se baser sur Chord afin d'implémenter la DHT [27]. Chord défini un espace d'adressage à une dimension en cercle de taille 2^K, où K est l'ordre du système. En d'autres termes, une position sur le cercle peut être une valeur appartenant à l'ensemble $[0; 2^K - 1]$. La valeur de K dépend de l'échelle attendu du système. Une valeur traditionnellement utilisée dans les systèmes existants est $K = 128$.

Nous définissons deux fonctions de hachage : $f(\cdot)$ et $g(\cdot)$. Elles seront utilisées respectivement pour la position des nœuds racines et les pointeurs vers les objets sur la courbe. Nous en définissons deux afin de donner la liberté au système de séparer les nœuds des objets.

Premièrement, soit I_{r_i} l'identifiant du nœud r_i. Il est important de souligner que tous les nœuds doivent rejoindre la DHT pour être considéré dans au niveau le plus haut des ONS. Au nœud racine r_i est associé une position sur la courbe donnée par :

$$H_{r_i} = f(I_{r_i}). \tag{5.1}$$

Le nœud racine doit connaître son prédécesseur et son successeur sur la courbe, *i.e.* , respectivement les nœuds r_p et r_s placés sur la courbe tels qu'il n'y a pas de nœud x tel que $I_{r_p} < I_x < I_{r_i}$ et $I_{r_i} < I_x < I_{r_s}$; nous appellerons ces nœuds p_{r_i} et s_{r_i}. Il y a deux raisons à cela. D'abord, une telle information va être utilisée pour acheminer les requêtes dans le système. Ensuite, ça nous permet de définir la notion de « région de responsabilité ». Ainsi, au nœud racine r_i est assigné la région C_{r_i} comprise entre sa propre position et celle de son prédécesseur (plus un) :

$$C_{r_i} = [H_{p_{r_i}} + 1; H_{r_i}]. \tag{5.2}$$

Nous ferons référence à cette intervalle comme la région de responsabilité du nœud racine r_i. Cela veut dire que tous les objets qui seront associés à un point de l'intervalle C_{r_i} seront sous la responsabilité du nœud r_i. Pour cela, nous utilisons la seconde fonction de hachage $g(\cdot)$ pour lier les objets aux points sur la courbe. Soit J_{e_j} l'identifiant de l'objet e_j, son point sur la courbe L_{e_j} est donné par :

$$L_{e_j} = g(J_{e_j}). \tag{5.3}$$

En faisant cela, nous pouvons dire que le nœud racine r_i est responsable de l'objet e_j si et seulement si $L_{r_i} \in C_{r_i}$. Par responsabilité, nous entendons que le nœud r_i connait l'ONS local capable de s'occuper de la requête.

5.2.3 Opérations sur les objets

Avant d'expliquer les opérations d'arrivée ou de départ du réseau, nous allons décrire les opérations sur les objets. Il existe trois opérations sur les objets : `ajouter`, `recueillir`, et `supprimer`.

Ajout d'un objet

Cela consiste à ajouter un nouvel objet dans le système. La figure 5.4 montre les deux étapes de l'ajout d'un objet. Tout d'abord, l'entreprise envoie un message d'enregistrement d'un nouvel objet à l'ONS racine auquel elle est connectée (flèche ❶). Notons que cette étape est naturellement compatible avec le système existant. Cet ONS racine est responsable d'ajouter un pointeur dans la DHT. Ce pointeur est basiquement l'adresse de l'ONS local. Il calcule $g(J_{e_j}) = L_{e_j}$ et envoie un message `ajouter` vers le nœud racine r dont la région de responsabilité contient L_{e_j}, i.e., $L_{e_j} \in C_r$ (flèche ❷).

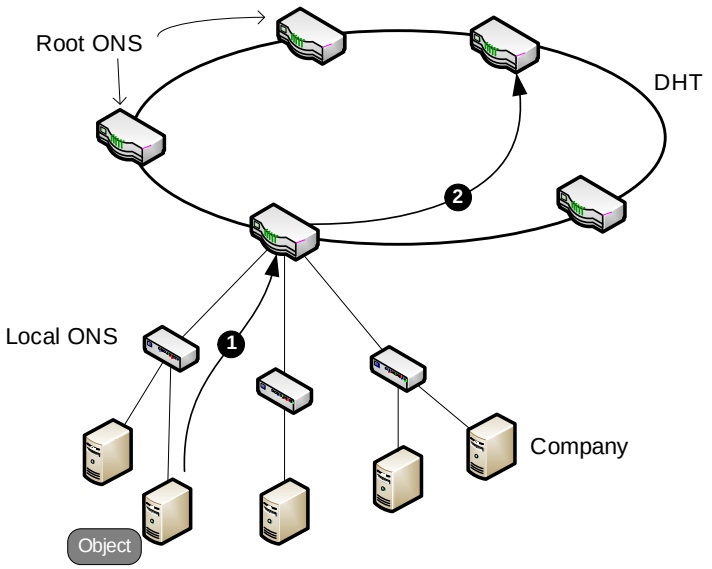


FIGURE 5.4 – *Opération* `ajouter`.

Recueillir des informations sur un objet

L'opération `recueillir` consiste à obtenir le pointeur vers un ONS local qui connait la localisation de l'objet (voir figure 5.5). Cette opération est très similaire à l'opération d'ajout. L'ONS racine qui reçoit une requête d'information sur un objet (flèche ❸) vérifie d'abord s'il ne connait pas lui-même l'ONS local qui gère cet objet. Si il ne le connait pas, il effectue une opération de recherche dans la DHT. Comme dans l'opération d'ajout, il calcule $g(J_{e_j})$ et demande des informations sur l'objet à l'ONS racine responsable (flèche ❹). Si l'objet est correctement enregistré, par l'opération d'ajout, le bon ONS local est renvoyé en réponse (flèche ❺). Si non, un message d'erreur est retourné. Dans le cas d'une réponse positive, la requête peut être transmise au bon ONS local directement (flèche ❻). Notons que la façon dont cette requête est transférée vers l'ONS local dépend du type d'adresse utilisé. En utilisant les adresses IP, nous pouvons utiliser le traditionnel routage DNS et IP, comme dans l'implémentation actuelle de l'ONS.

Supprimer

L'opération de suppression est basiquement la même que l'ajout, avec la différence que le nœud responsable supprime l'entrée (si elle est présente dans le système).

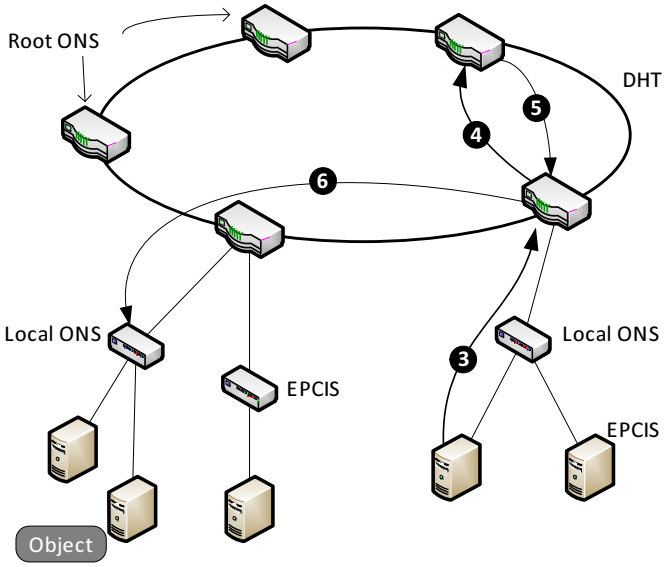


FIGURE 5.5 – *Opération* `recueillir`.

5.2.4 Opérations sur les nœuds racines

Nous adressons maintenant le problème d'arrivée et de départ des nœuds dans le système. Ces opérations sont fondamentales pour assurer la robustesse du système et sont au nombre de deux : `joindre` et `quitter`.

Joindre

Lorqu'un nouveau nœud racine r_i souhaite se joindre au système, il doit d'abord contacter un nœud déjà présent dans le système. Ce dernier va aider le nouveau nœud à obtenir les coordonnées du nœud qui gère la région contenant $f(I_{r_i}) = H_{r_i}$. Appelons ce nœud m_{r_i} ($H_{r_i} \in C_{m_{r_i}}$). Pour être inséré dans la DHT, le nouveau nœud va hériter de m_{r_i} la part de la région comprise entre sa position et la position de m_{r_i}. Plus formellement :

$$C_{r_i} = [p_{m_{r_i}} + 1; H_{r_i}]; \tag{5.4}$$
$$C_{m_{r_i}} = [H_{r_i} + 1; H_{m_{r_i}}]; \tag{5.5}$$
$$p_{r_i} = p_{m_{r_i}}; \tag{5.6}$$
$$s_{r_i} = m_{r_i}; \tag{5.7}$$
$$p_{m_{r_i}} = r_i; \tag{5.8}$$
$$s_{p_{m_{r_i}}} = r_i; \tag{5.9}$$

En héritant d'une région, le nouveau nœud hérite aussi des pointeurs correspondants aux objets déjà présents dans le système. Notons que cette procédure d'arrivée requiert un comportement collaboratif. Cela veut dire, dans sa forme basic, que les nœuds devraient être hiérarchiquement équivalent et suivre les mêmes règles.

Quitter

Lorsqu'un nœud quitte la DHT, il doit donner sa région de responsabilité à son successeur et déclencher une mise à jour de la DHT :

$$C_{s_{r_i}} = [H_{p_{r_i}} + 1; H_{s_{r_i}}]; \tag{5.10}$$
$$p_{s_{r_i}} = p_{r_i}; \tag{5.11}$$
$$s_{p_{r_i}} = s_{r_i}; \tag{5.12}$$
$$\tag{5.13}$$

Ici, en retournant sa région de responsabilité, le nœud quittant le réseau transfère aussi les pointeurs qu'il maintenait à son successeur.

5.2.5 Initialisation

Dans tous les systèmes distribués, il existe un étape critique qui est celle de l'initialisation du système. C'est d'autant plus important lorsqu'un nœud veut joindre le système.

Le but ici est que, lorsqu'un nœud souhaite joindre le réseau et effectuer l'opération `joindre`, il a besoin de connaitre un nœud de contact déjà dans le système ou d'en demander un pour avoir un point d'entrée. Avec l'implication d'EPCGlobal dans un tel système, ce problème de point d'entrée peut être facilement résolu comme EPCGlobal peut servir de nœud de contact. Cela veut dire que lorsqu'un nouveau nœud veut entrer dans le réseau, il demande une adresse d'un nœud de contact à EPCGlobal. EPCGlobal peut répondre avec l'adresse de n'importe quel nœud actif dans le système. Le nouveau ONS racine peut alors contacter ce nœud et être ainsi introduit dans l'anneau.

Si le nœud arrivant est le premier nœud du système à entrer, *i.e.* , il n'y a pas d'autre nœud, cet ONS aura la responsabilité de tout l'espace d'adressage. Tous les autres nœuds racines auront à effectuer l'opération `joindre` comme spécifié dans la section 5.2.4. il est important de souligner le fait que le premier nœud introduit dans le système n'aura pas plus d'importance que tous les autres qui arriveront plus tard. Par exemple, cela pourrait être l'ONS actuel.

Notre solution propose une distribution générale des identifiants. Avec une fonction de hachage particulière, on peut distribuer les informations de façon ciblée. Par exemple, comme expliqué dans la sous-section 2.2, les identifiants EPC contiennent des informations de localisation du fabriquant (les trois premiers chiffres du préfixe de compagnie). Avec une fonction de hachage enregistrant les identifiants de la même région sur le même nœud, et en plaçant un nœud par région, il suffit de configurer les applications métiers pour interroger par défaut l'ONS de leur région, et on se retrouve dans le cas présenté dans la sous-section 5.1.2. Mais on peut aussi avoir une fonction de hachage répartissant les informations de façon intelligente afin de prévenir d'une surcharge de réseau, d'une panne d'un nœud, *etc.*.

5.3 Discussions et améliorations possibles

5.3.1 Duplication et système multi-hachage

Pour assurer la sécurité et la consistence des données, elles doivent être dupliquées dans le réseau. L'opération `ajouter` doit pouvoir retourner plusieurs pointeurs qui peuvent être obtenus de deux façons différentes. D'abord, la fonction de hachage $g(\cdot)$ peut retourner jusqu'à k pointeurs, où k est le niveau de redondance choisi. Ensuite, nous pouvons définir d'autres fonctions de hachages qui devront être utilisée dans chaque opération du système.

L'utilisation de plusieurs fonctions de hachages offre de la robustesse en cas de panne d'un ONS (local ou racine). Si le premier noeud de contacte échoue (ou l'ONS local qui semble contenir l'information), le système peut interroger un deuxième noeud de contact obtenu grâce à une

deuxième fonction de hachage. Les pointeurs des différents lieux de stockage des données sont ainsi dupliqués.

5.3.2 Mécanismes de recherche

Jusqu'à maintenant, nous avons décrit le moyen basique pour chercher et d'acheminer les requêtes dans la structure. Chaque noeud garde une table de routage dans laquelle il enregistre les adresses de son prédécesseur et de son successeur. En plus, il enregistre l'adresse de 2^l ses voisins. Ces noeuds sont choisis de façon exponentielle afin d'optimiser le temps de recherche dans une distribution homogène. Par exemple, en supposant que les noeuds racines sont numérotés de façon continue sur le cercle, le noeud 0 enregistre les noeuds 1 (2^0), 2 (2^1), 4 (2^2), 8 (2^3), ..., 2^{K-1}, ainsi que 2^K. Lorsque le noeud 0 doit manipuler une requête pour un contenu avec un identifiant X, il va le transmettre au noeud dans sa table qui à l'identifiant le plus proche de X.

En prenant en compte le temps de l'application et les besoins en mémoire, ce routage peut être amélioré en permettant aux noeuds d'avoir plusieurs tables de routage. Les noeuds dans les tables peuvent être choisis d'une façon différente ou être plus nombreux. Ce dernier cas permettrait une recherche plus rapide mais impliquerait un plus grand espace mémoire occupé sur les ONS racines et plus de données à mettre à jour en cas d'arrivée/départ d'un noeud. En effet, un compromis doit être trouvé.

5.3.3 Points d'entrée multiples

Pour retrouver des informations sur un EPC, le système va demander à son ONS racine (son point d'entrée dans la structure des ONS) après l'ONS local qui contient l'information recherchée. Lorsque ce point d'entrée tombe en panne, le système dépendant de lui ne sera plus capable de fournir une réponse. Posséder plusieurs points d'entrée permet au système d'interroger un point d'entrée de sauvegarde dans ce cas de panne.

5.4 Conclusion

Il y a plusieurs avantages dans l'utilisation des DHT pour la distribution. Le passage à l'échelle et la dynamicité du réseau sont les premiers d'entre eux. Grâce à la capacité à joindre et à quitter le réseau, un nouvel ONS racine peut être ajouté ou supprimé très facilement. De plus, comme il n'y a pas d'ONS prédominant sur les autres et qu'aucun d'entre eux peut obtenir plus de pouvoir qu'un autre (ils dépendent les uns des autres), l'ensemble du système se restaure automatiquement en cas de panne de l'un d'entre eux. De tels système de DHT sont déjà largement répandus et ont montré leur efficacité dans les systèmes à très large échelle, donc le passage à l'échelle est un avantage bien connu.

Dans ce travail, il y a des avantages spécifiques pour la distribution de l'ONS d'EPCGlobal, tel que la conservation du système existant de l'actuel ONS. Ce noeud deviendra un noeud comme les autres. Le résultat d'une requête d'information sur un EPC est une adresse d'un ONS local contenant cette information. Donc même si l'ONS racine de cet ONS local est en panne, toutes les informations sur les EPC gérés par cet ONS local sont toujours disponibles.

Par manque de temps, l'implémentation de notre solution n'est pas finalisée et n'a pas permis d'effectuer des tests de performance. Mais au vu des performances offertes par les solutions déjà implémentées utilisant les DHT, comme les systèmes de partage de fichiers, on peut espérer montrer que, répondant aux besoins du service de l'ONS, cette solution offre de meilleures performances et une meilleure robustesse qu'une solution basée sur les DNS. De plus, au vu des différents problèmes des DNS, aussi bien géopolitiques que de passage à l'échelle ou encore d'utilisateurs malveillants, concevoir une nouvelle approche pour ce réseau qu'est l'Internet des

Objets et la résolution des noms de ces objets permet d'envisager un déploiement plus efficace et plus facile tout en prenant en compte l'expérience du déploiement des DNS afin de ne pas faire les mêmes erreurs.

Publications

1. Rapport de recherche :
   - *Distributed Planetary Object Name Service : Issues and Design Principles*. M. Dias De Amorim, S. Fdida, N. Mitton, L. Schmidt et D. Simplot-Ryl. RR-7042, INRIA, 2009

Autres contributions 6

Tous les travaux présentés précédemment se basent sur les standards EPCGlobal afin de les distribuer pour lui offrir le passage à l'échelle. Mais il y a deux points importants à ne pas négliger : (i) tous les RFID ou systèmes d'identification n'étant compatibles EPCGlobal et (ii) l'augmentation des lecteurs mobiles et les besoins en applications mobiles pour la RFID.

En effet, il existe une quantité d'autres systèmes d'identification (ID) déjà très utilisés. Les codes à barres de GS1 [20], les cartes à puce sans contact ISO 14443 [43] ou 15693 [44], *etc.*, peuvent aussi bien être utilisés dans une structure utilisant aussi des étiquettes EPCGlobal. Ou en remplacement, on peut préférer des étiquettes non-EPC pour des raisons politiques, ou parce qu'on a déjà l'infrastructure utilisant une autre norme qu'EPCGlobal. De plus, il est intéressant d'offrir une intégration de ces systèmes d'identification dans notre intergiciel RFID distribué. En effet, intégrer ces ID dans un intergiciel à très large échelle permet d'avoir une solution générique, compatible avec tous les ID et passant à l'échelle (voir sections 3 et 5).

Les lecteurs mobiles sont de plus en plus répandus et performants. Certains intègrent même un écran, ou se connectent sur des PDA ou des smartphones. Dans l'optique d'une utilisation régulière de ce genre de matériel, contraint par la batterie, il convient d'étudier les couts des communications entre ces lecteurs et l'intergiciel sur lesquels ils sont connectés. En effet, un des composants clés de l'intergiciel permet de filtrer et d'agréger les données reçues des lecteurs. Mettre ce composant dans un PDA ou un smartphone, ou encore l'embarquer dans le lecteur afin de profiter de leurs ressources pourrait permettre d'économiser la batterie en envoyant des évènements moins gros, ou surtout moins souvent.

Ce chapitre est organisé comme suit. La section 6.1 présente un moteur de traduction des ID, appelée Tag Data Translation (TDT) [7], englobant plusieurs standards d'identification. Dans la section 6.2, nous présenterons un composant de filtrage et d'agrégation (F&A) destiné à l'embarquer dans un PDA muni d'un lecteur RFID.

6.1 Tag Data Translation générale

Dans cette section, nous proposons une TDT générale étendant le standard d'EPCGlobal, ne traitant lui que des identifiants EPCGlobal. Nous présenterons dans un premier temps d'autres standards d'identifications dans la section 6.1.1 (comme les standards de carte à puces ISO 14443 et 15693, les codes à barres GS1, *etc.*). La section 6.1.2 explique comment nous avons inséré ces standards dans la TDT.

6.1.1 Autres types d'identifiants standards

Le système d'identification de GS1 est la base de celui d'EPCGlobal. En effet, comme expliqué dans la section 2.2 sur les standards TDT et TDS, un SGTIN d'EPCGlobal est composé d'un numéro de série et d'un GTIN, ce dernier que l'on retrouve dans les codes à barres EAN/UPC

très largement répandus dans le domaine de l'identification des classes d'objets. Le tableau 6.1 présente les relations entre les identifiants de GS1 et ceux d'EPCGlobal.

Type d'EPC	**Type GS1 original (Application Identifier)**
GID	
SGTIN	GTIN avec un numéro de série (01 + 21)
SSCC	SSCC (00)
SGLN	GLN avec un numéro de série (254 + 21)
GRAI	GRAI (8003)
GIAI	GIAI (8004)
DoD	

FIGURE 6.1 – *Relation entre les types de code GS1 et ceux d'EPCGlobal*

Les identifiants d'application (Application Identifiers - AI) dans le tableau 6.1 sont utilisés pour identifier le rôle du code qui va le suivre, comme l'entête des codes EPC. Les GTIN sont utilisés pour identifier les classes d'objets, *etc.*. Il y a approximativement une centaine d'AI dans la spécification générale GS1 [20]. Le tableau 6.1 montre qu'avec la bonne combinaison d'AI, on peut avoir un code EPC complet. Néanmoins, tous les types de codes à barres ne peuvent pas porter tous les AI GS1. L'EAN/UPC ou l'ITF-14 ne supportent que les GTIN. Le GS1-128, le G1 DataBar et le GS1 DataMatrix peuvent supporter tous les AI. Ainsi, un code EAN que l'on trouve sur nos produits n'identifiera pas ce produit en particulier, mais uniquement sa classe. En revanche, on peut mettre un GTIN et un numéro de série dans un GS1-128, et ainsi identifier un produit en particulier.

Il existe néanmoins d'autres standards que EPCGlobal utilisés dans la RFID et les cartes à puces tels que ISO 14443 ou encore ISO 15693. Dans la norme ISO 14443, l'*ID*entifiant *U*nique (IDU) fait une taille de 4, 7 ou 10 octets pour respectivement les IDU appelés simples (Figure 6.2), doubles ou triples (Figure 6.3).

idu0	**Description**
'08'	idu1 à idu3 est un numéro aléatoire généré dynamiquement
'x0' - 'x7' 'x9' - 'xE'	nombre fixe propriétaire
'18' - 'F8' 'xF'	réservé pour un usage future

idu$n$: n^e octet de l'IDU

FIGURE 6.2 – *IDU de taille simple ISO 14443*

idu0	**Description**
ID du fabricant d'après ISO/IEC 7816-6/AM1	Chaque fabricant est responsable de l'unicité des valeurs des autres octets du numéro unique.

idu$n$: n^e octet de l'IDU

FIGURE 6.3 – *IDU de taille double ou triple ISO 14443*

L'IDU dans la norme ISO 15693 est long de 64 bits (voir figure 6.4).

PF		pf
64 57	56 49	48 1
'E0'	ID du fabricant d'après ISO/IEC 7816-6/AM1	Numéro de série unique assigné par le fabricant de la puce

PF : bit de poids fort
pf : bit de poids faible

FIGURE 6.4 – *Format de l'IDU ISO 15693*

Ces autres standards d'identification sont à prendre en compte afin d'offrir un intergiciel

complet. La section suivante présente ces codes intégrés dans le processus de translation de la TDT.

6.1.2 TDT générale

Dans cette section, nous proposons un moteur général de traduction des identifiants [45]. Le but est d'ajouter les standards présentés dans la section précédente dans le processus de traduction de la TDT afin d'offrir la possibilité de les utiliser au sein du même intergiciel. Nous avons choisi de montrer ici la traduction des identifiants de GS1 et des ISO 14443 et 15693. Notre TDT a besoin immédiatement d'un paramètre afin d'identifier le type de code que l'on manipule, GS1/EPC ou ISO.

Standard de GS1

La première chose à faire dans le cas des codes à barres de GS1 est de savoir à quel code on a affaire. Dans la spécification du système de GS1, il est défini des identifiants symboles (Symbology Identifiers - SI). Ces SI offrent justement le moyen de connaître le type du code à barres, comme le montre le tableau 6.5. Chaque lecteur est capable de savoir le SI d'un code lu. Et même si le lecteur ne peut pas envoyer cette information avec les données de l'étiquette, nous supposons que l'application qui utilise cette TDT connait le type de code à barres.

SI	**code à barres correspondant**
]E0	EAN-13, UPC-A ou UPC-E
]E1	Symbole d'ajout de deux digits
]E2	Symbole d'ajout de cinq digits
]E3	EAN-13, UPC-A ou UPC-E avec un symbole d'ajout
]E4	EAN-8
]I1	ITF-14
]C1	GS1-128
]e0	GS1 DataBar
]d2	DataMatrix

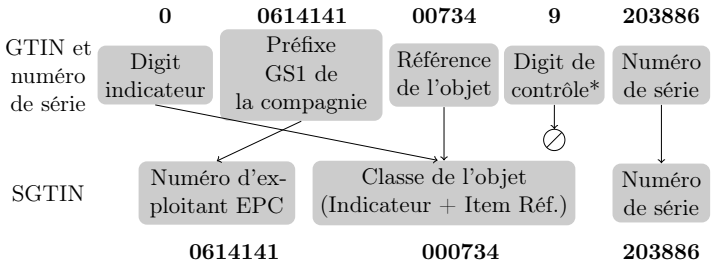
FIGURE 6.5 – *Symbology Identifiers du système de GS*1

Concernant les codes à barres GS1, nous avons ajouté un nouveau format appelé *gs1-ai-identifier*. Ce format commence avec les trois caractères du SI, et est suivi par le code. Il y a deux types de code : (i) sans les AI, donc le code est un GTIN ; (ii) avec des AI entre parenthèses (e.g. *]C1(01)00012345678905(21)12345678* représente un GTIN et un numéro de série dans un code à barres GSA-128). Tous les type de code à barres GS1 peuvent être traduits sous cette forme et dans le format *legacy*.

Dans la section précédente, nous avons vu que certains codes à barres GS1 (GS1-128, GS1 DataBar and GS1 DataMatrix) pouvait supporter plusieurs AI, et donc peuvent supporter un EPC. La figure 6.6 montre le lien entre un GTIN plus un numéro de série (AI 01 et 21) et un SGTIN. Nous appellerons ces codes « conformes EPC » comme ils peuvent être traduit dans toute les représentations de la TDS. Notre TDT peut aussi convertir tous les EPC dans ce nouveau format *gs1-ai-identifier* (mais seulement la version avec les AI).

Normes ISO

Notre TDT peut aussi traduire les identifiants ISO 14443 et 15693 dans plusieurs représentations. Ces standards sont utilisés dans la RFID, les cartes à puce, et la NFC. Les formats doivent suivre les mêmes structures que ceux du standard TDT afin d'être intégrés dans un intergiciel global. De cette façon, les ID peuvent être traduits en *binaire*, *legacy*, *pure-identity*, *tag-encoding*



*le digit de contrôle du code à barres n'apparaît plus dans l'EPC

FIGURE 6.6 – *Du GTIN muni d'un numéro de série au SGTIN*

et *ons-hostname*. Ces formats sont présentés dans la figure 6.7 pour la norme ISO 15693. Le mécanisme est le même pour la norme ISO 14443.

Représentation	**Valeur**
tag-encoding	urn :iso :tag :15693-64 :98.104197
pure-identity	urn :iso :id :15693 :98.104197
ons-hostname	104197.98.15693.onsiso.com
legacy	iso15693 ;mfgcode=98 ;serial=104197
binaire	11110000011000100000000000000000 00000000000000000011001011100000101

FIGURE 6.7 – *Format pour les ID ISO 15693*

Moteur de la TDT

La TDT prend en paramètre l'identifiant, dans n'importe quel format, et un nombre variable de paramètres dépendant les uns des autres : (i) le format voulu en sortie ; (ii) le type de code (GS1 ou ISO) ; (iii) le SI GS1 et la taille du code (pour le format de sortie *gs1-ai-encoding*) ; (iv) taille du code, taille du préfixe de compagnie et filtre (pour certain format, voir section 2.2). La figure 6.8 montre le comportement de la TDT. Nous utilisons l'implémentation de Fosstrak [15] de la TDT d'EPCGlobal comme moteur de traduction pour les EPC.

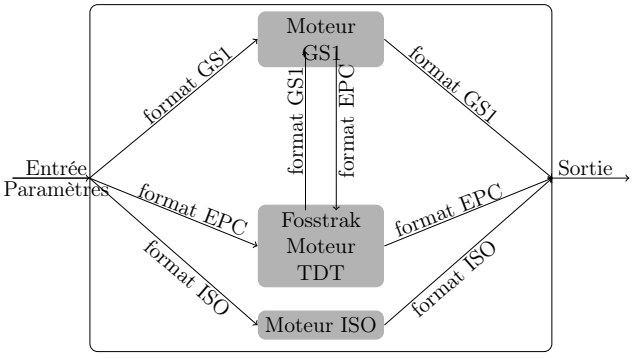


FIGURE 6.8 – *Moteur de la TDT générale*

Si la donnée d'entrée est de type ISO, alors le moteur ISO est choisi pour faire la traduction. C'est un peu plus compliqué quand la donnée d'entrée est de type GS1 à cause des codes à barres « conforme EPC ». Si la TDT doit traduire un EPC dans la forme *gs1-ai-encoding*, la TDT de Fosstrak soit d'abord la traduire au format *legacy* avant de la relayer au moteur GS1. Le processus est identique dans le cas d'une entrée au format *gs1-ai-encoding*, sauf que le moteur utilisé pour le traduire en *legacy* est le GS1. L'avantage d'une telle architecture est l'indépendance du moteur de la TDT de Fosstrak. Il est très facile de mettre à jour ce moteur lorsque le standard d'EPCGlobal est lui aussi mis à jour.

Nous avons rappelé qu'il existe d'autres identifiants que le codes à barres GS1 et les RFID EPC, et qu'il est assez aisé de les intégrer dans la TDT. Nous avons présenté les codes à barres GS1 et les normes ISO 14443 et 15693, mais il existe une quantité d'autres système d'identification. Ainsi nous avons rajouté dans notre implémentation [46] les numéros de téléphone avec les préfixes internationaux, les adresses MAC des cartes réseaux, les IMEI [1] et IMEISV [2] des GSM [47], ainsi que les identifiants des « iButton » [48]. Notons que dans la version 1.4 du standard de la TDT, le format *legacy-ai* apparaît et propose une représentation des identifiants EPC avec les AI de GS1, similaire à notre solution.

6.2 Composant de filtrage et d'agrégation pour l'embarqué

Les lecteurs mobiles sont de plus en plus répandus et pour cause, ils facilitent énormément certaines opérations. Prenons le cas de l'opération d'inventaire. L'utilisateur parcours le rayon avec son lecteur mobile. Ce dernier va donc envoyer un rapport par lecture d'identifiant. Le problème ici est la consommation d'énergie. Si le lecteur mobile doit recharger sa batterie après avoir envoyé quelques centaines d'évènements, il n'est pas efficace parce qu'une telle opération peut traiter plusieurs milliers d'identifiants. La section suivante présente une étude sur le cout des messages à envoyer, et permet de savoir ainsi dans quelles situations il est préférable de ne pas envoyer tous ces évènements avec le lecteur. Dans la section 6.2.2, nous présenterons notre composant de filtrage et d'agrégation embarqué et réparti.

6.2.1 Définition du problème

Dans une opération telle que l'inventaire, le lecteur envoi un évènement à chaque lecture d'identifiant au composant F&A, qui va se charger de construire le rapport *ECReports* pour l'application au dessus. Donc avec un inventaire de deux identifiants différents, le lecteur a envoyé au moins deux messages (une étiquette RFID peut être lues plusieurs fois tant qu'elle est encore dans le champ du lecteur), tandis que le F&A n'en a envoyé qu'un seul. Comparons la taille de ces paquets afin d'en tirer des conclusions. Dans ces expérimentations, nous utiliserons le protocole LLRP d'EPCGlobal [11] entre le lecteur et le F&A. Dans ce standard, un rapport de lecture provenant d'un lecteur s'appelle un *RO_ACCESS_REPORT*. Nous analyserons aussi les évènements sortant du composant F&A, sous forme de fichier XML.

Le tableau 6.9 montre que la lecture d'une étiquette RFID engendre un évènement *RO_ACCESS_REPORT* d'une taille de 73 octets, alors qu'un ECReports avec un seul identifiant fait 1283 octets. La première colonne du tableau indique la taille du rapport ECReports au format XML bien formaté (avec les retours à la ligne et les indentations), la seconde ne prend pas en compte les retours à la ligne ni les indentations du XML. La troisième colonne correspond à la taille du rapport, mais en utilisant cette fois une sérialisation des objets Java, ce qui a tendance à fortement réduire la taille des messages. Ce mécanisme d'envoi de rapport est celui que l'on utilise pour les échanges de ECSpecs et ECReports entre les composants F&A dans notre solution distribuée (voir section 3). Enfin la dernière colonne est pour les rapports du lecteur. Chacune de ces colonnes est divisée en deux, l'une pour la taille en octet du message (en gris clair dans le tableau), et l'autre pour le nombre minimum de paquet TCP nécessaire à l'envoie du (des) message(s) (en blanc dans le tableau).

Pour calculer le nombre de segments TCP [49], deux paramètres sont à prendre en compte :

- MTU : unité de transmission maximum (Maximum Transmission Unit), taille du plus grand paquet IP pouvant être envoyé sur le réseau ;

1. International Mobile Equipment Identity - Identité internationale de l'équipement mobile.
2. IMEI Software Version

Nb d'ID	ECReports XML complet		ECReports XML léger		ECReports sérialisé		Rapports du lecteur [3]	
	Taille en octet	Segment TCP	Taille en octet	Segment TCP	Taille en octet	Segment TCP	Taille en octet	Segment TCP
1	1283	1	1137	1	311	1	73	1
5	2443	2	2137	2	1063	1	5*73	1
10	3896	3	3390	3	2006	2	10*73	1
20	6802	5	5896	5	3892	3	20*73	1
30	9801	7	8471	6	5795	4	30*73	2
100	30794	22	26496	19	19116	14	100*73	5
200	60699	42	52185	36	38137	27	200*73	10

FIGURE 6.9 – *Taille des rapports en fonction du nombre d'identifiants*

– MSS : taille maximale d'un segment (Maximum Segment Size), nombre d'octet maximum dans un segment TCP (entête TCP ou IP mis à part) ;

Afin de calculer le MSS, on va soustraire la taille des entêtes IP et TCP au MTU. Dans la RFC 879, il est dit que le MTU d'Ethernet (10Mb) est de 1500 octets. Il est dit aussi que la taille des entêtes IP et TCP étaient comprises entre 20 et 60 octets. Toujours selon la RFC, dans la plupart des cas, ces entêtes font 20 octets. Ce qui nous donne :

$$MSS = MTU - sizeof(TCPHDR) - sizeof(IPHDR) \tag{6.1}$$

(avec TCPHDR l'entête TCP et IPHDR l'entête IP), soit $MSS = 1500 - 20 - 20 = 1460$ octets.

D'après ces chiffres, on peut en déduire qu'il est plus intéressant d'envoyer les évènements du lecteur que le rapport du F&A. Mais dans la configuration des lecteurs, on donne une période de répétition de l'opération. Ainsi, si le F&A construit un ECReports toutes les 10 secondes puis attend 3 secondes (les *boundarySpec* de l'ECSpecs, voir section 2.3) avant de recommencer l'opération, mais que le lecteur est configuré pour envoyer les ID à envoyer toutes les 4 secondes, alors le composant F&A aura reçu au moins deux fois chacun des évènements du lecteur pendant la construction, et il va probablement en recevoir encore une copie pendant qu'il attend. En fonction d'une certaine configuration du lecteur et du F&A, bien spécifique à l'opération à effectuer, il est plus intéressant d'envoyer le rapport du F&A. Mais, pour s'assurer de l'utilité d'un F&A embarqué dans un PDA/smartphone, il faut aussi prendre en compte le retour que peut avoir l'utilisateur grâce à l'écran. De plus, dans une opération d'inventaire, l'utilisateur du PDA ne peut pas savoir si son inventaire est bon sans l'agrégation sur le PDA, sinon, il ne voit que des ID défiler.

6.2.2 Un composant F&A réparti

Les performances matérielles, toujours meilleures, des appareils mobiles équipés d'un lecteur RFID (*e.g.* un PDA ou un smartphone) offrent une possibilité d'embarquer le composant F&A dans l'appareil mobile. Cela permet de filtrer et d'agréger les données dans l'appareil mobile et ainsi n'envoyer qu'un seul rapport à la fin de l'opération, augmentant la durée de vie de ses batteries.

L'opération d'inventaire est intéressante dans notre cas car elle traite de nombreux identifiants EPC. Quand un lecteur mobile sans-fil effectue cette opération, il envoie les EPC lus directement au composant F&A. Donc l'homme qui est derrière le lecteur n'a aucun retour sur son inventaire. Ce retour est un des avantages clés de l'utilisation d'un PDA et d'un F&A embarqué. Le deuxième avantage est que le lecteur et le PDA sont tous deux mobiles, et tous deux manipulés par l'homme faisant l'opération. Cela rend la distance entre le lecteur et le F&A nettement plus courte qu'avec un F&A classique qui peut être à l'autre bout du magasin/hangar, voir même encore plus loin, la transmission sans-fil est moins couteuse en consommation d'énergie. Au lieu d'envoyer plusieurs

paquet contenant les évènements de lecture par wifi vers un ordinateur fixe, le lecteur les envoie au PDA par bluetooth ou même par un câble (comme les deux objets sont très proches et mobiles, le lier à l'aide d'un câble n'est pas un problème en terme de mobilité).

Nous avons donc développé un composant F&A que nous avons embarqué dans un PDA muni d'un lecteur RFID. Afin d'optimiser notre composant, nous avons choisi de ne pas traduire les ECReports en XML dans le PDA, mais sur un hôte séparé. En effet, dans le graphe 6.10, nous remarquons que la version sérialisée des ECReports nécessite moins de paquet TCP à envoyer que les versions XML.

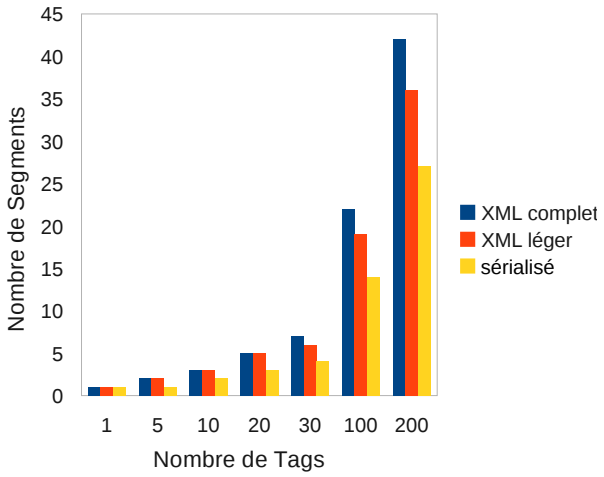


FIGURE 6.10 – *Nombre de segments à envoyer en fonction du nombre d'ID lus*

Notre composant F&A embarqué va donc filtrer et agréger les évènements des lecteurs, puis va envoyer son ECReports sérialisé vers un hôte, qui pourra alors le traduire et le transmettre aux applications demandeuses avec un autre mécanisme, par exemple ceux définis dans le standard d'EPCGlobal, afin de le rendre compatible avec l'intergiciel.

Nous avons implémenté cette solution ainsi qu'une application d'inventaire [50] sur un PDA dans le cadre du projet ICOM [51].

Dans ce chapitre, nous avons présenté une brique des standards EPCGlobal modifié afin d'accepter d'autres identifiants que les EPC. Cette brique englobe les standards ISO 14443 et 15693, normes des cartes à puces sans contacts. Elle englobe aussi les codes à barres GS1 tels que l'EAN/UPC ou ITF-14, *etc.*. C'est le premier pas vers un intergiciel général, capable de supporter tous types d'identifiant. Il ne suffit pas hélas d'avoir juste cette brique pour ajouter ces identifiants dans l'architecture de l'intergiciel, il faut aussi modifier les autres standards EPCGlobal afin qu'ils reconnaissent aussi ces identifiants non-EPC. En effet, si on ajoute ces identifiants dans les couches des lecteurs, mais que le composant F&A ne rapporte jamais ces ID car il ne reconnaît pas l'ID, l'application métier n'aura jamais connaissance de ces ID. Il faut donc propager ces modifications aux autres standards, ce que nous avons déjà fait pour le F&A.

Nous avons aussi présenté dans une deuxième partie un composant F&A pour l'embarqué. Les cibles de ce composant sont les PDA (ou smartphone) équipés d'un lecteur RFID. Ce composant filtre et agrège sur le PDA les évènements reçu du lecteur connecté, et va ensuite envoyer son rapport sérialisé à un hôte pour qu'il le traduise/transforme pour être conforme au standard d'EPCGlobal. Entre l'hôte et le PDA, le protocole de communication est le même que celui utilisé entre les nœuds dans notre machine à évènement distribuée (voir chapitre 3). Ainsi, on pourrait interroger le composant F&A embarqué dans le PDA à partir d'un nœud de ce réseau distribué.

Publications

1. Conférences internationales :
   - *Towards Unified Tag Data Translation for the Internet of Things.* L. Schmidt, N. Mitton, and D. Simplot-Ryl. Wireless Communication Society, Vehicular Technology, Information Theory and Aerospace & Electronics Systems Technology (VITAE'09), Aalborg, Denmark, 2009.
2. Développements logiciels :
   - AspireRFID Wiki - TDT. *http ://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation/TDT*
   - AspireRFID Wiki - Embedded ALE. *http ://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation.Filtering&Collection/EmbededALE*

Conclusion et perspectives 7

Nos travaux ont porté sur le passage à l'échelle des intergiciels RFID. Nous nous sommes intéressés à différents composants de cet intergiciel et avons pour chacun proposé une solution pour un passage à l'échelle efficace.

Ainsi, nous avons présenté une solution de distribution du composant de filtrage et d'agrégation des données en utilisant les tables de hachage distribuées (DHT). Ce système de DHT est adapté pour répondre au problème de passage à l'échelle de ce composant. Les données qui y sont partagées sont les données recueillies par les lecteurs. Chaque lecteur (la clé) est enregistré dans la DHT ainsi que l'URL du service du composant de filtrage et d'agrégation (un nœud). De cette manière, en connaissant le nom d'un lecteur et en interrogeant la DHT on peut accéder au nœud en charge de sa gestion. Cette solution fournit, en plus du passage à l'échelle, la transparence aux couches supérieures. Cela est rendu possible par un mécanisme de division des spécifications en fonction des lecteurs qui y sont impliqués. Ces parties de spécifications sont distribuées vers les nœuds ayant des lecteurs impliqués. Enfin, la fusion des rapports reçus en retour des nœuds permet de construire un unique rapport qui sera envoyé au destinataire des couches supérieures. Ainsi, ces trois étapes permettent une transparence pour ces dernières. Le coût de cette transparence n'est cependant pas négligeable et serait certainement optimisable. Il serait intéressant d'étudier ces possibilités d'optimisation. En ce qui concerne les lecteurs logiques distribués, en utilisant CAN à la place de Chord pour l'architecture pair-à-pair, avec un espace en trois dimensions couplé à une géolocalisation physique des nœuds F&A, on pourrait automatiser la définition de lecteurs logiques géographiques, ce qui faciliterait l'ajout de lecteur physique dans les lecteurs logiques composites.

Nous nous somme ensuite intéressés aux architectures auto-organisées permettant la gestion de larges stocks. Nous nous sommes placés dans le cadre d'une application devant gérer d'importantes quantités de données réparties sur différents sites géographiques (les entrepôts), chacun de ces sites disposant de bases de données EPCIS pour répertoriés les objets présents dans les différents entrepôts. Pour un accès facilité à ces données ainsi qu'une tolérance aux pannes, nous proposons une architecture permettant de distribuer ces données efficacement. Cette architecture permet de personnaliser le niveau de réplication des données, permettant de choisir le niveau de robustesse approprié à l'application. Elle fournit également un mécanisme efficace de gestion des requêtes, évitant ainsi d'inonder le réseau à chaque requête. Nous avons évalué par simulation la tolérance aux pannes de notre solution donnant des résultats prometteurs, et des résultats en situation réelle viendraient compléter cette évaluation.

Nous nous concentrons ensuite sur le composant ONS dont le comportement est relativement similaire à celui du DNS. L'ONS est effectivement le DNS de l'Internet des Objets. Nous avons exploré la piste de la distribution basée sur les tables de hachage distribuées. A chaque identifiant EPC est associé un ou plusieurs services. Ce service peut être, par exemple, un EPCIS, c'est-à-dire la base de données contenant les informations relatives à cet identifiant, ou encore un ONS local connaissant l'adresse de l'EPCIS. Une évaluation de la solution proposée permettrait de vérifier

son efficacité. Ces travaux s'intègre dans le cadre du projet WINGS, financé par l'ANR et initié par GS1, et visant à faire une évaluation du standard actuel de ONS. Les évaluations et les comparaisons pourraient permettre une évolution de ce standard lui offrant une bonne solution de passage à l'échelle.

Les travaux suivants ont concerné des problématiques annexes : l'intégration d'autres systèmes d'identification standardisés et l'intégration d'un composant de filtrage et d'intégration dans un lecteur mobile. Les systèmes d'identification standardisés dont l'intégration au composant logiciel du standard TDT comprennent les codes à barres GS1 et les normes ISO 14443 et 15693, les numéros de téléphone avec préfixes internationaux, les adresses MAC, les IMEI des GSM ainsi que les identifiants de iButton. Le composant de filtrage et d'agrégation développé pour des PDA ou des smartphones offre des perspectives pour de nombreuses applications. Nous avons ainsi déployé une application d'inventaire utilisant un lecteur mobile et affichant le rapport sur l'écran du PDA.

Des évaluations plus poussées de chacune de ces briques distribuées permettrait de connaître la viabilité de nos solutions, et ainsi les proposer lors des groupes de travail sur les standards EPCGlobal.

Bibliographie

[1] R. KAlinowski, M. Latteux et D. Simplot. Procédé de Détection d'Emissions Simultanées d'Etiquettes Electroniques, International patent WO0191037, nov. 2001. in french.

[2] ITU, *Internet Reports 2005 : The Internet of Things*, ITU, 2005.

[3] EPCGlobal Inc., *http ://www.epcglobal.org*

[4] Auto-ID Labs, *http ://www.autoidlabs.org*

[5] EPCGlobal Inc., *EPCglobal Architecture Framework (1.3)*

[6] EPCGlobal Inc., *EPCglobal Tag Data Standards Version 1.4*

[7] EPCGlobal Inc., *EPCglobal Tag Data Translation Version 1.4*

[8] EPCGlobal Inc., *EPCglobal Application Level Events (1.1.1)*

[9] EPCGlobal Inc., *EPCglobalEPC Information Services (1.0.1)*

[10] EPCGlobal Inc., *EPCglobal Object Naming Service (1.0.1)*

[11] EPCGlobal Inc., *EPCglobal Low Level Reader Protocol (1.0.1)*

[12] EPCGlobal Inc., *EPCglobal Reader Protocol (1.1)*

[13] EPCGlobal Inc., *EPCglobal Reader Management (1.0.1)*

[14] Aspire European Project, *http ://www.fp7-aspire.eu/*

[15] Fosstrak : Open Source RFID Software Platform, *http ://www.fosstrak.org/*

[16] Rifidi : Software Defined RFID, *http ://www.rifidi.org/*

[17] CUHK EPCGlobal RFID Middleware 1.0. *http ://mobitec.ie.cuhk.edu.hk/rfid/middleware/index.htm*

[18] logicAllow : rfid made easy, *http ://www.logicalloy.com/*

[19] LLRP Toolkit, *http ://www.llrp.org/*

[20] GS1, *GS1 General Specifications v10*

[21] DNS. Domain Name Service, *http ://www.howstuffworks.com/dns.html*

[22] J. P. Hubaux, Th. Gross, J. Y. Le Boudec, and M. Vetterli. Towards self-organized mobile ad hoc networks : the terminodes project. *IEEE Communications Magazine*, 39(1) :118–124, January 2001.

[23] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of ACM Mobicom*, Boston, MA, August 2000.

[24] Y. Xue, B. Li, and K. Nahrstedt. A scalable location management scheme in mobile ad-hoc networks. In *Proceedings of IEEE Conference on Local Computer Networks (LCN).*

[25] Napster. *http ://www.napster.com/*

[26] The Gnutella protocol specification, 2000. *http ://www9.limewire.com/developer/gnutella_protocol_0.4.pdf/*

[27] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord : a scalable peer-to-peer lookup protocol for internet applications. *Networking, IEEE/ACM Transactions on*, 11(1) :17–32, 2003.

[28] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker. A Scalable Content-Addressable Network. In *Proc. of ACM SIGCOMM*, San Diego, CA, August 2001.

[29] A. Rowstron and P. Druschel. Pastry : Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. of the IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, p.329-350, November 12-16, 2001

[30] K. Hildrum, J.D. Kubiatowicz, S. Rao and B.Y. Zhao. Distributed object location in a dynamic network In *Proc. of the 14th ACM Symp. on Parallel Algorithms and Architectures*, August 2002.

[31] L. Schmidt, R. Dagher, R. Quilez, N. Mitton et D. Simplot-Ryl. DHT-based distributed ALE engine in RFID Middleware *http ://hal.inria.fr/inria-00491795/PDF/RR-7316.pdf* Research Report 7316, INRIA, 2010

[32] Jae Geol Park, Heung Seok Chae, and Eul Seok So. A dynamic load balancing approach based on the standard rfid middleware architecture. In *ICEBE '07 : Proceedings of the IEEE International Conference on e-Business Engineering*, pages 337–340, Washington, DC, USA, 2007. IEEE Computer Society.

[33] Fagui Liu, Yuzhu Jie, and Wei Hu. Distributed ale in rfid middleware. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, pages 1–5, Oct. 2008.

[34] Open Chord, *http ://open-chord.sourceforge.net/*

[35] Distributed ALE, *http ://chercheurs.lille.inria.fr/schmidt/Research.html*

[36] LLRP Simulator, *http ://chercheurs.lille.inria.fr/schmidt/Research.html*

[37] A. Carneiro Viana, N. Mitton, L. Schmidt, M. Vecchio. A *k*-layer self-organizing structure for product management in stock-based networks In *IEEE ICEBE*, Shangaï, China, 2010. IEEE Computer Society. *to appear*

[38] Y. Busnel, M. Bertier, and A.M Kermarrec. Solist : A lightweight multi-overlay structure for wireless sensor networks. RR 6404, INRIA, 2007.

[39] A. C. Viana, M. Dias de Armorim, S. Fdida, and J. Ferreira de Rezende. Indirect routing using distributed location information. In *Proc. of IEEE PERCOM*, Washington, DC, USA, 2003. IEEE Computer Society.

[40] M. Dias De Amorim, S. Fdida, N. Mitton, L. Schmidt et D. Simplot-Ryl. Distributed Planetary Object Name Service : Issues and Design Principles *http ://hal.inria.fr/inria-00419496/PDF/RR-7042.pdf* Research Report 7042, INRIA, 2009

[41] WINGS : Widening interoperability for networking global supply chains, ANR Project. *http ://www.wings-project.fr/*

[42] S. Evdokimov, B. Fabian et O. Günther. Multipolarity for the object naming service *The Internet of Things*, 1–18, Springer, 2008

[43] ISO/IEC, *ISO/IEC FCD 14443-3 :1999*

[44] ISO/IEC, *ISO/IEC FCD 15693 :2000*

[45] L. Schmidt, N. Mitton, and D. Simplot-Ryl. Towards Unified Tag Data Translation for the Internet of Things. *Wireless Communication Society, Vehicular Technology, Information Theoryand Aerospace & Electronics Systems Technology (VITAE'09)*, Aalborg, Denmark, 2009.

[46] AspireRFID Wiki - TDT.
http ://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation/TDT

[47] GSM Alliance IMEI et IMEISV. *http ://www.3gpp.org/ftp/Specs/html-info/23003.htm*

[48] iButton. *http ://www.maxim-ic.com/products/ibutton/*

[49] J. Postel. TCP maximum segment size and related topics RFC879, IETF, 1983 *http ://www.ietf.org/rfc/rfc879.txt*

[50] AspireRFID Wiki - Embedded ALE. *http ://wiki.aspire.ow2.org/xwiki/bin/view/Main.Documentation.Filtering&Collection/EmbededALE*

[51] Pôle de compétitivité des Industries du COMmerce, Infrastructure pour le COMmerce du future. *http ://www.picom.fr/*

Liste des figures