Mémoire

présenté par

Nathalie Mitton

en vue de l'obtention du diplôme

HABILITATION À DIRIGER DES RECHERCHES

de l'Université de Lille 1
Sciences et Technologies

INTERNET DES OBJETS, AUTO-ORGANISATION ET PASSAGE À L'ÉCHELLE.

Soutenance prévue le 26/05/2011.
Numéro d'ordre : 40523.

Après avis de : Michel Banatre
Serge Fdida
Ivan Stojmenovic

Devant la commission d'examen formée de :

Michel Banatre (Rapporteur)
Directeur de recherche à l'INRIA - ACES
Michel Diaz
Directeur de recherche au CNRS - LASS
Serge Fdida (Rapporteur)
Professeur à l'Université Pierre et Marie Curie – Paris VI
Éric Fleury
Professeur à l'ENS LYON - INRIA DNET
David Simplot-Ryl
Professeur à l'Université de Lille 1 - INRIA POPS
Ivan Stojmenovic (Rapporteur)
Professeur à l'Université de Ottawa, Canada

Table des matières

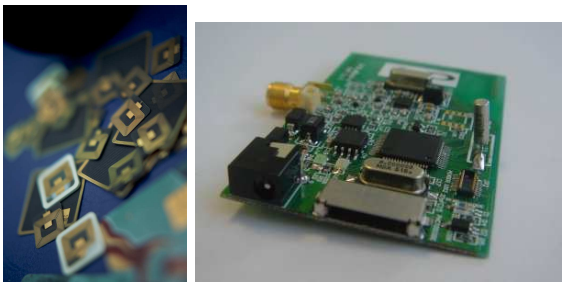
Chapitre 1

Introduction

L'Internet des objets [10] est un large sujet qui englobe tous les objets communicants comme les réseaux de capteurs, les systèmes RFID (Radio Frquency IDentification) ou encore les téléphones portables. Les avancées technologiques permettent de nouvelles applications de jour en jour. C'est dans ce contexte que j'ai effectué l'ensemble de mes recherches. Ces dernières ont d'abord porté sur les réseaux ad-hoc durant ma thèse [54]. Puis, je me suis concentrée sur les réseaux de capteurs [24] ou RFID active, semblables en certains points aux réseaux ad-hoc mais plus contraints. J'ai ensuite élargi mes recherches à la RFID passive, couvrant ainsi un plus large spectre de l'Internet des objets. Tous mes travaux considèrent des systèmes contraints en termes de ressources mémoire, processeur et énergétique.

La RFID passive

Les étiquettes RFID passives (ou identification par radio fréquence) sont composées d'une puce et d'une antenne (voir figure 1.1(a)). Elles n'embarquent aucune énergie propre et ne peuvent 'parler' que si elles sont alimentées par le champ d'un lecteur RFID. Les premières applications de la RFID passive comme on la connait maintenant sont des applications anti-vol. Les étiquettes se contentent de se manifester en traversant le portique de sortie de magasin, ce qui lève une alarme. Avec les avancées technologiques, les étiquettes ont embarqué de plus en plus de mémoire. Elles sont maintenant ré-inscriptibles et certaines parties de leur mémoire peut être protégée par mot de passe. Ces nouvelles fonctionnalités ouvrent la porte à de nouvelles applications. En effet, une étiquette RFID peut être utilisée comme un code-barre électronique qui en plus peut mémoriser chaque étape de la vie d'un produit. Cela permet de distinguer deux produits qui sont pourtant identiques, ce qui autorise des inventaires instantanés, des retours ciblés, une traçabilité facilitée. La technologie devenant plus accessible, elle commence à se propager et on la rencontre désormais à l'aéroport pour assurer le suivi des bagages, à la montagne dans nos



(a) RFID passive - © INRIA / Photo J. Wallace (b) Capteur sans fil © INRIA / Photo N. Fagot

FIGURE 1.1 – Etiquettes RFID, passive et active (capteurs sans fil).

forfaits de ski ou encore dans nos tickets de métro. Avec ce déploiement de la RFID, apparaît un nouveau besoin, celui de pouvoir échanger les données et informations portées par ces étiquettes entre plusieurs entreprises et donc de la nécessité d'un standard pour assurer l'interopérabilité. GS1 [31] est une organisation mondiale chargée de définir et diffuser des stdandards. Les standards EPCGlobal [21] sont les standards de GS1 liés à la technologie RFID, définissant l'identifiant des codes jusqu'à la définition des services d'échanges en passant par la caractérisation des lecteurs, des fréquences des étiquettes et des intergiciels. Les recherches que j'ai menées dans le domaine de la RFID passive se situent au niveau MAC (entre le lecteur et l'étiquette) mais aussi et surtout au niveau de plusieurs briques des intergiciels EPCGlobal afin de leur permettre de faire face à l'augmentation du nombre d'étiquettes et donc du volume de données à gérer et de passer à l'échelle.

De la RFID active aux réseaux de capteurs

Lorsque que les étiquettes RFID sont équipées d'une source d'énergie (batterie), elles peuvent alors communiquer de proche en proche jusqu'atteindre le lecteur (une seule étiquette a besoin de se trouver dans le champ du lecteur) (voir figure 1.1(b)). De nouvelles problématiques interviennent alors. Les étiquettes RFID forment alors un réseau sans fil et peuvent être vues comme un réseau de capteurs. De nouvelles applications sont alors possibles. Les capteurs peuvent être déployés pour surveiller un phénomène climatologique ou sismique (surveillance de feu de forêt, de volcan, etc). Répartis dans un parking souterrain dans lequel un incendie s'est déclaré, les capteurs peuvent identifier le foyer (grâce à des capteurs de chaleur) et guider les pompiers et secouristes. Placés sur des animaux, les capteurs permettent leur observation non intrusive par les biologistes. Bien d'autres applications ont également été envisagées. Cependant, cela soulève de nouvelles problématiques. Il s'agit maintenant d'établir des mécanismes permettant au réseau de s'organiser, de communiquer tout en

tenant compte des capacités restreintes des étiquettes (faible quantité mémoire, faible puissance de calcul) et en économisant sa batterie afin de survivre le plus longtemps possible. C'est dans ce contexte que j'ai mené le reste de mes recherches. Je me suis concentrée sur des protocoles permettant au réseau de s'auto-organiser et de bien fonctionner indépendamment de sa taille au travers de protocoles de routage et de contrôle de topologie. Les protocoles de routage permettent aux capteurs d'envoyer les données qu'ils ont prélevées sur leur environnement en utilisant un minimum de ressources. Le contrôle de topologie permet une meilleure utilisation des ressources en ne considérant qu'un sous-ensemble de liens les plus fiables.

Organisation du document

Ce document s'articule autour de trois chapitres principaux traitant des trois aspects majeurs de mes travaux, que sont les techniques pair-à-pair, le routage géographique et enfin le contrôle de topologie et la mobilité. Dans le chapitre 2, je présente la façon dont j'ai utilisé les tables de hachage distribuées, outil issu des techniques pair-à-pair pour permettre l'auto-organisation et le passage à l'échelle, aussi bien dans la RFID passive que active. Le chapitre 3 présente l'ensemble de mes travaux concernant les protocoles de routage géographique dans les réseaux de capteurs. Les protocoles de routage géographique semblent être les algorithmes de routage les plus adaptés aux réseaux de capteurs de par leur caractère local et sans mémoire. J'ai étudié plusieurs hypothèses dans ce domaine et proposé des algorithmes qui s'avèrent économes en énergie tout en garantissant la livraison du message. Enfin, le chapitre 4 présente les mécanismes que j'ai étudiés pour soit être plus robuste par rapport à la mobilité des capteurs qu'on ne contrôle pas (contrôle de topologie) soit comment exploiter la mobilité contrôlée d'actionneurs pour localiser des capteurs ou couvrir une zone. Ce mémoire se termine par un chapitre de conclusion et de perspectives où je donne les grandes lignes de mon programme de recherche.

Chapitre 2

Techniques P2P appliquées à la localisation et aux systèmes RFID

Les réseaux de capteurs comptent des centaines d'entités. Les étiquettes RFID vont se démultiplier dans les années à venir. Ainsi, qu'il s'agisse d'un réseau de capteurs ou de systèmes RFID, tous deux ont à traiter de nombreuses entités et doivent mettre en place des mécanismes robustes qui permettent le passage à l'échelle des primitives de ces réseaux de façon transparente pour l'utilisateur. Lorsque ces primitives peuvent être réalisées de manière locale sans surplus d'information à stocker, le passage à l'échelle se fait sans difficultés. Cependant, dans certains cas, les entités du réseau peuvent avoir besoin de récupérer ou stocker une information potentiellement dynamique. Cette information peut par exemple être la position d'un nœud dans un réseau de capteurs. Cette position est nécessaire à un autre nœud qui souhaiterait lui envoyer un message. Dans un système RFID, ces informations à stocker pour les besoins d'une compagnie peuvent être relatives à la traçabilité de centaines d'étiquettes. Une solution simple et facile à mettre en place est l'utilisation d'un serveur centralisé. Un tel serveur pourrait contenir la position de tous les capteurs dans le réseau de capteurs et pourrait répondre à la requête de tout nœud au sujet de tout autre. Un tel serveur pourrait être connecté à tous les lecteurs RFID d'une compagnie. Cependant, dans un pareil cas, ce serveur saturerait vite en termes de mémoire pour pouvoir stocker toutes les informations du réseau et en tant que ressources réseaux car il sera submergé par les requêtes et ne pourra pas y répondre. Il faut donc distribuer l'information. Cependant, distribuer l'information nécessite la mise en place de mécanismes qui identifient des lieux de rendezvous/stockage d'information dépendant de l'information à stocker. C'est ce que permettent les tables de hachage distribuées, outil issu des techniques Pair à Pair que nous avons utilisé afin de proposer des solutions à ces problèmes de passage à l'échelle et d'auto-organisation. Dans ce chapitre, je décris

rapidement dans un premier temps les tables de hachage distribuée ainsi que leurs mécanismes de base. Je détaille ensuite comment j'ai utilisé ces DHT dans les réseaux de capteurs pour localiser des nœuds, les nœuds enregistrant leur position relative auprès d'autres nœuds désignés par la DHT. Je me suis ensuite intéressée au passage à l'échelle de briques d'intergiciel RFID standard (EPCGlobal) dans le cadre de différents projets collaboratifs (FUI ICOM [35], ANR VERSO WINGS [78], FP7 IP ASPIRE [3]). Entre autres buts, ces projets ont en commun le respect des standards EPC Global pour la RFID afin d'assurer une certaine portabilité et le passage à l'échelle de certaines briques (GS1-EPCGlobal est d'ailleurs partenaire ou porteur de deux de ces projets). EPC Global [21] définit les spécifications auxquelles doit répondre un intergiciel RFID. Cet intergiciel a pour but de 'capturer' les données remontées par les lecteurs RFID, de les trier, filtrer, agréger, consolider avant de les fournir aux applications qui en auront besoin. Je décris donc comment à l'aide de DHT, j'ai distribué trois briques de l'intergiciel standard : l'ONS, l'ALE et l'EPC-IS.

2.1 Les tables de hachage distribuées

Les Tables de Hachage Distribuées (DHT - *Distributed hash table*) offrent un moyen de décorréler une information et sa position, permettant ainsi un passage à l'échelle. Les DHT fournissent une association générale entre une clef et toute sorte d'information (comme l'identité d'un nœud ou une position). Elles utilisent un espace d'adressage virtuel $\mathcal{V}$. Des partitions de cet espace virtuel sont allouées aux nœuds du réseau. L'idée est d'utiliser une fonction depuis un espace réel vers un espace virtuel $\mathcal{V}$. Cette fonction, dite de *hash*, permet aux nœuds d'identifier certains points de rendez-vous auprès desquels ils enregistrent les informations qu'ils détiennent (cette information pouvant être leur position). Cette fonction de *hash* est connue de tous les nœuds du réseau et peut ensuite être utilisée par n'importe lequel d'entre eux susceptible d'avoir accès à l'information stockée sur le lieu de rendezvous. Une information connue de tous (comme le nom de l'entité dont on cherche la position ou le type de donnée sauvegardée) est *hashée* en une clef ($hash(v) = clef_v$) de l'espace d'adressage virtuel $\mathcal{V}$. Les informations associées à cette clef (comme la position des nœuds ou les résultats d'une lecture RFID) sont ensuite stockées sur le (ou les) nœud(s) responsable(s) de la partition de l'espace virtuel à laquelle la clef appartient. Cette opération retournant le(s) nœud(s) responsable(s) d'une certaine clef dans les systèmes utilisant des DHT, est appelée *look-up*. Plus de détails au sujet des opérations de look-up sont donnés dans [4].

Les DHT sont issues des systèmes pair-à-pair dans lesquelles elles sont utilisées au niveau applicatif. La clef "hachée" dans ces systèmes de partage de fichiers est l'identifiant d'un fichier (clef = $hash(fichier)$). L'information associée à la clef et maintenue par le nœud responsable de cette clef est l'identité des nœuds du réseau qui détiennent ce fichier. Les adresses virtuelles des nœuds (ou partitions de $\mathcal{V}$ dont ils sont responsables) forment un réseau *overlay* (*c.à.d* un réseau virtuel basé sur le réseau physique qui maintient des liens logiques

entre les nœuds) sur lequel les requêtes sont routées. Ce qui différencie majoritairement les différentes propositions de réseaux pair-à-pair dans la littérature est la géométrie de ce réseau *overlay*. En effet, la forme de ce réseau va de l'anneau (Chord [72]) au graphe de De Bruijn (D2B [25]) en passant par des arbres (Tapestry [79], Kademlia [53], Tribe [75]), des espaces d-dimensionnels (CAN [65]), des structures en forme de papillon [52] ou encore des structures hybrides arbres-anneaux (Pastry [68]).

La figure 2.1 illustre le fonctionnement d'une table de hachage distribuée. Le nœud k détient une information C. Cette information peut être un fichier ou sa position. Il enregistre C sur le nœud i identifié comme nœud de rendez-vous par la DHT (figure 2.1(a)). Si le nœud j cherche une information au sujet de C (ou C directement), il effectue une phase de lookup (figure 2.1(b)). Le nœud i répond à j avec suivant les applications, soit l'adresse ou se trouve C (j peut alors directement contacter k) ou C lui-même (figure 2.1(c)).

Dans mes travaux, j'ai utilisé les tables de hachage dans les réseaux de RFID et dans les réseaux de capteurs et d'actionneurs mais fondamentalement, leur utilisation est la même. Dans chaque cas, on cherche à localiser un nœud, un serveur ou un lecteur.

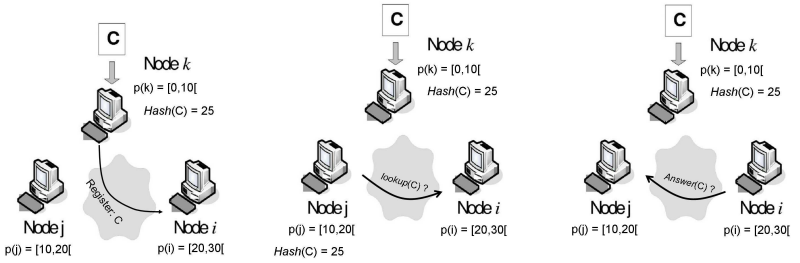


FIGURE 2.1 – (a) Le nœud k détient une information C, et l'enregistre sur le nœud i identifié comme nœud de rendezvous par la DHT. (b) Phase de lookup : j cherche une information sur C, il contacte i. (c) Réponse du Lookup avec l'information C.

2.2 Une DHT pour localiser dans les réseaux de capteurs

Dans les réseaux de capteurs, les tables de hachage distribuées sont généralement utilisées pour effectuer des opérations de routage indirect. Une opération de routage est qualifiée de *indirecte* si elle s'effectue en deux étapes : *(i)* le lookup qui permet de **situer** le nœud cible, puis, *(ii)* le routage qui permet à la source de **communiquer** directement avec le nœud ciblé.

Ce principe de routage est par exemple utilisé dans les réseaux de téléphonie GSM [1] ou dans le protocole Mobile IP [2], où la position de la destination est préalablement demandée respectivement aux HLR (Home Location Register) ou aux Home Agents avant d'établir directement la communication entre le demandeur et la destination. Cependant, un tel principe ne peut être directement appliqué dans un réseau de capteurs sans fil où tous les nœuds sont susceptibles de bouger, y compris les Home Agents potentiels.

J'ai utilisé les tables de hachage pour effectuer un routage indirect dans [57, 58, 59, 54] dans un contexte particulier. En effet, les nœuds avaient été regroupés en clusters en suivant l'algorithme développé pendant ma thèse [56, 55]. L'information 'hashée' est l'identifiant du nœud, l'information stockée est le nom du cluster auquel appartenait un nœud. Un table de hachage est déployée dans chaque cluster, chaque nœud du cluster étant responsable d'une partie de l'espace d'adressage. Lorsqu'un nœud enregistre sa position, il doit l'enregistrer dans son propre cluster mais aussi dans tous les clusters du réseau. Cette solution est adaptée à l'organisation en clusters de ces algorithmes car ces derniers formaient peu de clusters mais assez denses. L'overlay est un arbre défini durant la formation des clusters sur lequel on exécute un routage par intervalle sur l'espace virtuel afin d'atteindre le nœud de rendezvous. Une fois le cluster contenant le nœud recherché est connu, un autre algorithme de routage était mis en place pour l'atteindre.

Notre approche se détache des travaux de la littérature utilisant des DHT pour effectuer un routage indirect dans la mesure où dans les autres approches, les deux phases du routage indirect [70] (look-up et routage) sont toujours du même type : indépendantes de la DHT et effectuées dans l'espace physique ou dépendantes de la DHT et effectuées sur l'espace logique. Dans notre approche, les deux étapes de routage indirect sont effectuées différemment. Le look-up est effectué en utilisant un routage par intervalle sur les adresses virtuelles des points de rendez-vous alors que l'étape de routage s'effectue dans l'espace physique, indépendamment de $\mathcal{V}$.

2.2.1 Une DHT pour distribuer l'ONS dans les systèmes RFID

L'ONS. L'ONS [29] (ou Object Name Service) joue un rôle important dans le partage des données des étiquettes RFID. Il va permettre de récupérer des adresses de services disponibles en fonction de l'identifiant d'un produit lu sur un code barre (code GS1) ou sur une étiquette RFID (code EPC - Electronic Product Code) contenu dans la requête. L'ONS est une sorte d'annuaire qui permet à une entreprise de suivre la progression et le statut des marchandises depuis leur production, jusqu'à leur espace de vente. L'ONS permet de retrouver les adresses fournisseurs d'un produit (si enregistrés auprès d'EPCglobal), à l'image du DNS qui permet de retrouver les adresses IP d'un nom de domaine. L'ONS

1. http ://www.gsm.org
2. http ://www.ietf.org/rfc/rfc2002.txt

a d'ailleurs un fonctionnement identique à celui du DNS et repose sur lui. Son fonctionnement est illustré par la figure 2.2. La compagnie A recherche l'EPCIS (base de données du fournisseur contenant les informations produit) contenant des informations sur un EPC, en l'occurrence sur l'EPC 0614141.112345.400. L'application va alors demander à son ONS. 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).

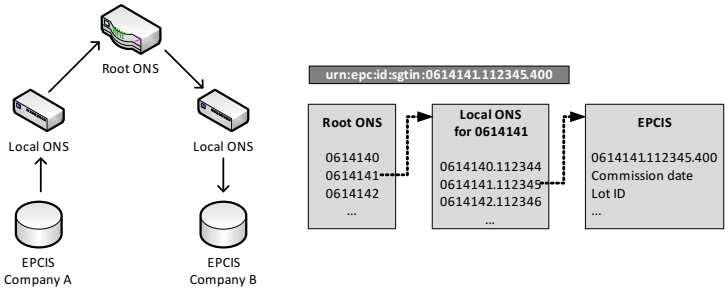


FIGURE 2.2 – Service de recherche du standard de l'ONS.

Problématique. Cette organisation centralisée actuelle de l'ONS soulève plusieurs problématiques. La première est d'ordre technique. En effet, comme tout système centralisé, ce système ne passe pas à l'échelle et n'est pas robuste face aux attaques. Dans un monde où le nombre d'étiquettes et d'identifiants RFID et donc de requêtes potentielles sur l'ONS tendent à augmenter, ce système ne s'avère pas robuste. La seconde problématique est d'ordre politique. En effet, ce serveur se situe actuellement aux États-Unis. Comme mesurer les requêtes lancées sur l'ONS sur un ensemble de produits revient à prendre le pouls économique d'un pays, les autres pays demandent un système qui pallierait cet inconvénient.

Un ONS distribué. Proposer un ONS distribué est donc nécessaire à différents points de vue. Il faut plusieurs racines ONS, connectées de façon à ce qu'aucune d'entre elles n'aient plus de responsabilités qu'une autre. La solution que nous avons proposée [2] repose sur des tables de hachage distribuées et plus précisément sur le système de coordonnées virtuelles de Chord [72].

Chord définit un simple espace d'adressage en 1-dimension, sur un cercle, de taille 2^K, où K est l'ordre du système et dépend de l'échelle du système. Une position sur le cercle peut être n'importe quelle valeur dans $[0; 2^K - 1]$. Traditionnellement, les systèmes existants utilisent $K = 128$. Chord définit également les processus de routage sur ce cercle pour atteindre le nœud recherché.

Nous voulons conserver les opérations de base définies par le service ONS actuel. Pour appliquer une DHT à ce système, nous considérons que :

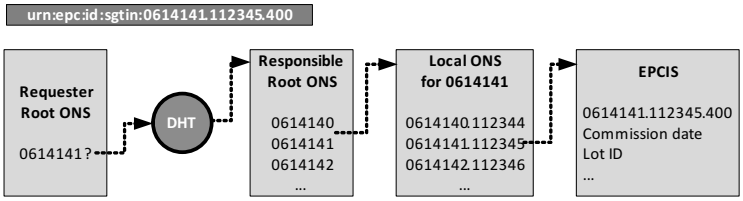


FIGURE 2.3 – ONS utilisé avec une DHT.

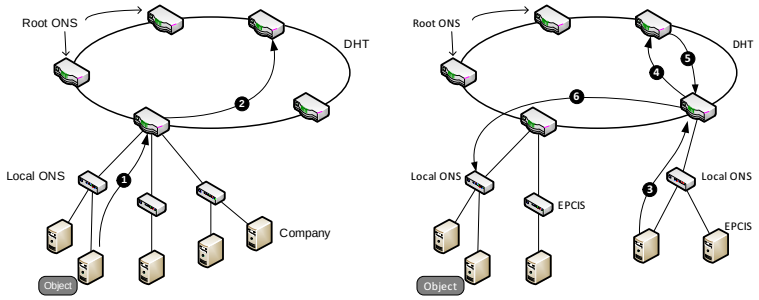
- Les racines ONS sont les nœuds de la DHT réparties sur l'espace d'adressage.
- Les ONS locaux sont les "clefs" permettant le service de localisation. Elles sont associées à une position dans l'espace d'adressage. La racine ONS gérant cette position correspondant est la racine qui détient le pointeur vers l'ONS local correspondant à l'objet.

La figure 2.3 illustre les modifications introduites dans le service de l'ONS. Ce principe permet au système d'être complètement distribué, indépendant du système sous-jacent et répondant aux contraintes geo-politiques. Deux tables de hachage sont utilisées, partagées par toutes les racines ONS et répartissant les identifiants sur un même espace. La première table de hachage f permet de positionner les objets (adresses ONS locaux) sur le cercle d'adressage virtuel. La seconde DHT g positionne les racines ONS.

Nous définissons deux opérations à effectuer sur les racines pour maintenir l'espace virtuel d'adressage : rejoindre et quitter. En effet, ces opérations sont effectuées lorsqu'un nouvel ONS intègre l'espace d'adressage et lorsqu'un autre le quitte. Ces opérations sont issues de Chord. Lorsqu'une nouvelle racine ONS r_i rejoint le système, elle doit d'abord contacter une des racines appartenant déjà à l'espace d'adressage. Cette dernière va l'aider à s'insérer dans le cercle. Pour cela, elle applique la deuxième DHT g sur l'identifiant ou adresse IP de r_i qui lui retourne la position p' dans l'espace d'adressage ($g(r_I) = p'$) et l'adresse de la racine ONS en charge de cette position, disons r_j. r_j est en charge de l'espace $[a, b[$ tel que $p' \in [a, b[$. Afin que r_i s'insère dans l'espace d'adressage, r_j doit céder une partie de son espace d'adressage à r_i. Il va lui donner l'espace $[p', b[$. En héritant de cette région de l'espace d'adressage, r_i hérite de l'ensemble des pointeurs enregistrés dans le système à ces positions. De plus, afin de maintenir l'anneau de l'espace d'adressage, r_i mémorise son précédent et son successeur dans l'espace d'adressage ainsi que leurs adresses. Il obtient donc comme précédent la racine r_j et comme successeur r_k l'ancien successeur de r_j. r_j change son successeur et r_k son précédent en r_i. De la même façon, lorqu'une racine r_i quitte le système, elle informe son précédent et son successeur dans l'anneau. Le précédent r_j de r_i récupère son espace d'adressage ainsi que les pointeurs correspondants. r_j change son successeur pour r_k, l'ancien successeur de r_i et r_k remplace son précédent par r_j.

Nous définissons ensuite trois opérations à effectuer sur les objets à stocker

dans la DHT : ajouter, requêter ou retirer un objet sur l'espace. Les opérations d'ajout et de retrait d'objet à enregistrer dans le système s'effectuent de la même façon. La figure 2.4(a) illustre les différentes étapes. La compagnie possédant le nouvel objet à enregistrer (ou à retirer) du système envoie un message à un ONS racine, quel qu'il soit (flèche ❶). La racine contactée est en charge d'insérer/retirer l'objet. Elle calcule la position p de l'objet dans l'espace virtuel en appliquant la DHT des objets sur l'identifiant de l'objet à ajouter/retirer ($f(Id) = p$). Une fois p obtenu, elle envoie un message à la racine r dont l'espace de responsabilité contient p (flèche ❷).



(a) Ajouter ou retirer un objet (b) Requêter un objet

FIGURE 2.4 – Les opérations sur les objets.

La figure 2.4(a) illustre les opérations de requête sur les objets qui consiste pour un utilisateur à récupérer le pointeur vers un ONS local particulier à partir d'un identifiant. Un utilisateur souhaitant connaître l'emplacement d'un objet contacte une racine ONS (flèche ❸). Il applique la DHT de l'objet f sur son identifiant ($f(id) = p$) qui lui retourne l'adresse de l'ONS responsable de cet objet qu'il contacte (flèche ❹). Si l'objet a bien été enregistré, ce dernier lui donne l'adresse de l'ONS local à contacter (flèche ❺). Ce dernier peut alors être contacté directement (flèche ❻).

Cette solution est en cours d'implémentation et d'expérimentation dans le cadre du projet WINGS [78]. Les premiers résultats sont prometteurs et encourageants. GS1 France souhaite les proposer au prochain groupe de travail de définition des standards de l'ONS.

2.2.2 Une DHT pour distribuer l'ALE dans les systèmes RFID

L'ALE. L'ALE [28] (Application Level Event) est le cœur d'un intergiciel RFID. C'est ce composant qui a la charge :

- de configurer les lecteurs physiques en lecteurs logiques,
- de recevoir les revendications des différentes applications,
- de collecter les données lues par les différents lecteurs, de les filtrer, agréger en fonction des demandes des applications,

– de renvoyer les données consolidées aux applications.

Un lecteur logique peut être soit un lecteur physique soit un ensemble de lecteurs, comme illustré par la figure 2.2.2. Les demandes des applications sont envoyées à l'ALE dans un fichier XML. Ce fichier va décrire :

– le ou les lecteurs logiques concernés par la requête,
– le temps de lecture,
– le temps entre deux lectures consécutives,
– le contenu du rapport. L'application peut demander :
  – la liste ou le nombre de tags lus durant la période définie (regroupés ou non suivant un filtre),
  – la liste ou le nombre de tags disparus depuis de la dernière lecture (regroupés ou non suivant un filtre),
  – la liste ou le nombre de tags apparus depuis la dernière lecture (regroupés ou non suivant un filtre),
  – plusieurs des listes précédentes.

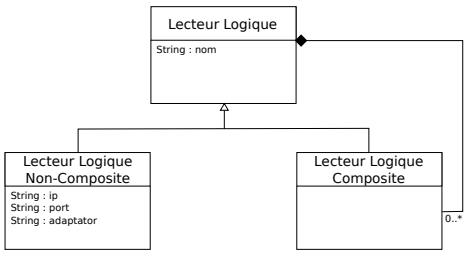


FIGURE 2.5 – Logical reader

Problématique. La figure 2.2.2 illustre une architecture EPCGlobal standard. Les lecteurs physiques récupèrent les données des étiquettes électroniques. Ces lecteurs physiques, configurés en lecteurs logiques, sont connectés au composant ALE lui-même connecté aux applications métiers. Comme tout système centralisé, on repère que cette architecture peut souffrir d'un goulot d'étranglement survenant entre les lecteurs et le composant ALE. Nos expérimentations ont montré que ce goulot intervenait rapidement, à moins de 9 lecteurs connectés à l'ALE, chacun avec une charge moyenne de 100 identifiants à remonter. Pour palier ce problème, deux solutions ont été proposées dans la littérature. La première propose de dupliquer les ALE [63] et appliquer un mécanisme d'équilibrage de charge entre les ALE. Ce mécanisme permet de migrer les demandes des applications d'une ALE trop chargée vers une autre. Cependant, toutes les demandes ne sont pas 'migrables' et le problème ne reste que partiellement résolu. La seconde proposition de la littérature est toujours de dupliquer les ALE et de leur confier à toute la même demande. Au lieu d'envoyer leurs résultats à l'application, les ALE les envoient à une ALE globale qui se charge de retirer les doublons entre les deux ALE, re-filtrer, regrouper et consolider l'ensemble des rapports. Cela ne fait donc que déplacer le goulot d'étranglement au niveau

de l'ALE globale.

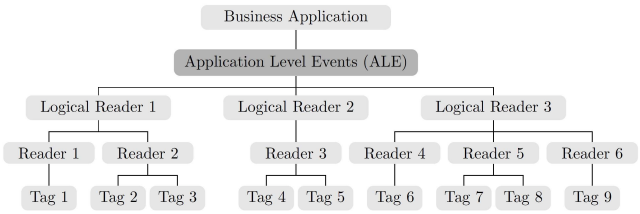


FIGURE 2.6 – Logical reader

Distribuer l'ALE. Afin de distribuer l'ALE, nous avons proposé d'utiliser plusieurs composants ALE connectés entre eux à l'aide d'une DHT [71]. Le fait d'utiliser une DHT permet de palier les problèmes rencontrés précédemment. En effet, la charge est répartie sur l'ensemble des composants ALE de façon transparente pour les applications, sans nécessiter une ALE globale devant collecter l'ensemble des rapports de toutes les ALE. Tout comme dans la section 2.2.1, nous utilisons Chord [72]. Nous avons choisi Chord pour sa simplicité et pour le fait qu'il répondait à nos besoins mais n'importe quel autre système P2P peut être utilisé. Dans notre solution, les éléments de la DHT se partageant l'espace d'adressage sont les composants ALE. Les objets à enregistrer sont les lecteurs physiques. Les mécanismes et opérations pour insérer/retirer une ALE sont les mêmes que pour insérer/retirer une racine ONS dans l'espace d'adressage décrits en section 2.2.1 (les clés utilisées sont les identifiants ou adresses IP des ALE). Les mécanismes pour enregistrer/requêter/supprimer un lecteur physique dans les ALE sont les mêmes que pour enregistrer/requêter/supprimer un ONS local dans la section 2.2.1 (les clés utilisées sont les identifiants des lecteurs). Je ne les redétaillerai donc pas ici.

L'idée est donc la même. Il faut juste adapter les mécanismes effectués par une ALE recevant une requête d'une application. Pour illustrer ces mécanismes, référons-nous à la figure 2.2.2 et supposons que l'application construise une spécification XML suivant les standards demandant un rapport des lecteurs logiques 1 et 2, rattachés à deux ALE différentes. L'application envoie ce fichier à n'importe laquelle des ALE de la DHT (la plus proche ou la moins chargée, par exemple), l'ALE 1 sur notre schéma (flèche ❷). Cette ALE analyse le fichier XML et se rend compte que ce dernier concerne deux lecteurs. Le lecteur 2 est sous la responsabilité de l'ALE 1 donc son rapport sera traité en local, alors que le lecteur 1 est inconnu à l'ALE 1. Pour le trouver, l'ALE 1 interroge la DHT en désignant l'ALE 5 (flèche ❸) qui lui répond où se trouve le lecteur 1, c'est à dire sous l'ALE 4 (flèche ❹). L'ALE 1 va donc scinder le fichier XML en deux. Cette opération de division de la spécification consiste à créer deux fichiers de spécifications : *(i)* l'un avec uniquement le lecteur 1 qui va être envoyé à l'ALE 4 grâce à l'adresse récupérée via la DHT (flèche ❺) ; *(ii)* l'autre avec uniquement le lecteur 2 et qui va rester en local sur l'ALE 1. Les deux ALE 1 et 4 vont

construire leur rapport en local. L'ALE 4 retourne alors son rapport à l'ALE 1 (flèche ❻). Cette dernière va alors fusionner les deux rapports en un seul, filtrant et agrégeant éventuellement à nouveau les données. Enfin, l'ALE 1 qui avait été directement contactée par l'application et reçu la spécification complète va alors pouvoir envoyer le rapport bien construit à l'application métier (flèche ❼). Nous illustrons les mécanismes de distribution de l'ALE en utilisant une spécification ne portant que sur deux lecteurs, mais les mécanismes sont les mêmes dans le cas de spécifications impliquant plusieurs lecteurs. L'ALE contactée par l'application devra contacter autant d'ALE que nécessaire, récupérer l'ensemble des rapports et les fusionner. 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.

Notre solution reste compatible avec les standards existants de EPCGobal. Nos expérimentations ont montré que l'overhead introduit par l'interrogation de la DHT était inférieur à 10% du trafic complet et que notre système permettait aisément le passage à l'échelle (lorsqu'une ALE est saturée à 9 lecteurs pour environ 100 tags par lecteur, notre solution distribuée avec deux lecteurs permet de repousser la limite à 15 lecteurs). Les étapes suivantes dans cet axe sont la mise en place de plus d'expérimentations afin d'étudier le comportement du système global avec un grand nombre d'ALE et de comparer ces résultats aux solutions de la littérature qui semblent moins robustes et moins transparentes. Nous participons également aux groupes de travail EPCGlobal de l'ALE pour échanger sur ce sujet.

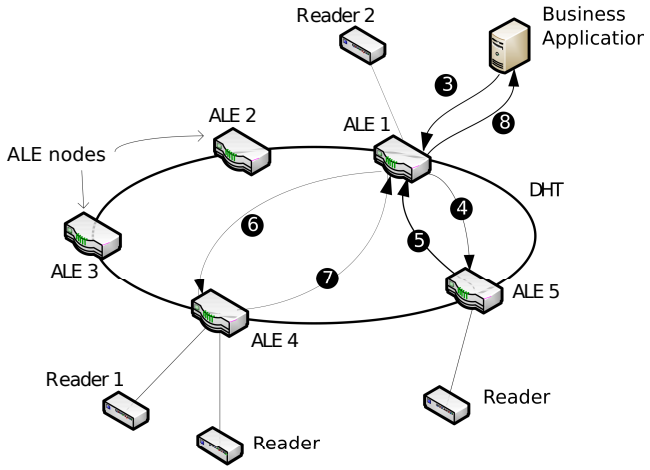


FIGURE 2.7 – Illustration de l'ALE distribuée.

2.2.3 Une DHT pour distribuer les bases de données dans les systèmes RFID

Toutes les informations traitées et formatées par l'ALE sont ensuite envoyées vers les applications, les bases de données et/ou l'EPC-IS. L'EPC-IS est une base de données dont les formats sont définis par EPCGlobal [27] afin que les entreprises soient en mesure de partager des données métier efficacement. Si on considère de grands entreprises réparties à l'échelle d'un pays voire de monde, le nombre de bases de données est très important ainsi que l'ensemble des données qui y sont sauvegardées. Ces bases de données sont elles aussi réparties et connectées en réseau. Afin de ne pas perdre d'information en cas de chute d'un site, ces données doivent être dupliquées de façon efficace. La possible augmentation du nombre d'entités dans le réseau ne doit pas le surcharger. Les données doivent être dupliquées pour assurer la fiabilité. Cependant, cette duplication doit se faire de manière efficace pour minimiser le surcoût qui lui est associé. Ces besoins nous ont été remontés par des acteurs de la grande distribution qui souhaitent pouvoir effectuer différents types de requêtes sur ces bases de données avec plusieurs niveaux de granularité. Il s'agit de 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 ?*). C'est pour répondre à ces besoins que nous avons proposé SENSATION [76], 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 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 liés à l'inondation).

SENSATION. Supposons qu'une entreprise souhaite dresser un inventaire des pantalons qu'elle a en stock sur l'ensemble de ses sites. Afin de ne pas avoir à interroger l'ensemble des bases de données, y compris celles qui ne possèdent pas de pantalons, SENSATION organise les bases de données en couches. Une couche peut représenter un type de produit (e.g. les pantalons) ou un lieu (site de Lille). Les couches vont être spécifiées en fonction de la granularité que souhaite l'entreprise. Est ce qu'on requête principalement les pantalons ou plus précisément les pantalons bleus (auquel cas on dressera une couche par couleur de pantalon) ou plus précisément encore des pantalons bleus de taille S (auquel cas on dressera une couche par couleur et par taille de pantalon) ? Une base de données appartient à chaque couche correspondant aux produits qu'elle contient.

SENSATION utilise deux tables de hachage distribuées. La première DHT permet d'atteindre la couche (ou les couches si plusieurs objets sont concernés) correspondant au type de l'objet pointé par la requête. La seconde DHT 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.

SENSATION s'inspire de SOLIST [8] qui propose un ensemble de services pour réseaux de capteurs, l'adaptant à notre contexte. Toutes les bases de données rattachées à une couche sont organisées en un espace virtuel d'adressage. Elles s'insèrent et se retirent de cet espace de la même façon que les racines ONS en section 2.2.1. Elles enregistrent les données qu'elles ont de la même façon que les ONS locaux sont enregistrés en section 2.2.1. Ainsi, lorsqu'une base de données contient des informations au sujet de pantalons, elle s'insère dans la base de données des pantalons. Elle utilise la DHT pour identifier sur quelle(s) autre(s) base(s) enregistrer les informations sur les pantalons qu'elle contient. La DHT va retourner p ensembles d'adresses de bases de données auprès desquelles enregistrer les données, où p correspond au degré de redondance souhaité. A noter que les informations enregistrées par un serveur A seront réparties sur les bases de l'espace d'adressage et non pas toutes sauvegardées au même endroit. Contrairement aux travaux présentés précédemment dans ce chapitre, nous n'utilisons pas Chord ici mais Tribe [75]. La raison est que Tribe offre une architecture en forme d'arbre respectant les proximités géographiques des nœuds. Une telle architecture est plus adaptée aux requêtes de type k-cast car une fois que k objets ont été trouvés, le parcours de l'arbre s'arrête. Bien que la structure sous-jacente de l'espace d'adressage soit différents, les mécanismes demeurent les mêmes.

Afin qu'une base de données contenant des pantalons puissent s'insérer dans la structure de la couche correspondante ou effectuer une requête sur les pantalons, elle doit dans un premier temps contacter une entité appartenant à la bonne couche. C'est là qu'intervient la seconde table de hachage. A partir de l'identifiant de l'objet recherché, la table de hachage retourne la position locale d'une base de données. Cette dernière n'appartient pas forcément à la couche recherchée mais elle connaît la base de données la plus proche appartenant à la bonne couche. Là encore, nous n'utilisons pas Chord mais CAN [65]. En effet, CAN prend en compte les positions géographiques des entités, permet de découper l'espace et d'assigner plusieurs points de rendezvous. CAN va découper l'espace réel en cellules. La table de hachage va retourner des coordonnées relatives dans chaque cellule. Ainsi, lorsqu'une base de données doit s'enregistrer, elle va contacter la base de données se trouvant dans sa cellule, limitant l'overhead.

Afin d'illustrer les mécanismes de SENSATION, considérons l'exemple suivant en nous appuyant sur la figure 2.8. Supposons que les carrés blancs représentent les nœuds de la couche des pantalons, organisés en structure d'arbre par Tribe.

Supposons maintenant que la base de données B souhaite dresser l'inventaire des pantalons répartis sur les différents sites d'une compagnie. Pour cela, B va utiliser la première DHT basée sur CAN pour identifier le point d'entrée. La DHT lui retourne la position relative du point d'entrée dans une cellule et B contacte le plus proche, ici $ep_{(2)3}$. $ep_{(2)3}$ retourne l'adresse de C à B qui est le nœud contact le plus proche. $ep_{(2)3}$ connaît C puisque ce dernier s'est enregistré au préalable auprès de $ep_{(2)3}$ en utilisant la même DHT. B va alors envoyer sa requête à C. Puisque C a préalablement rejoint la couche correspondant aux

pantalons, il peut transmettre le message à tous les nœuds gérant des pantalons s'étant enregistrés dans cette couche. La requête de dénombrement sera diffusée dans l'espace d'adressage et la réponse contiendra le nombre de pantalons de tout le réseau. Si la requête avait été du genre unicast ("Quel est le prix d'un pantalon ?"), C aurait répondu directement sans solliciter les autres nœuds de sa couche. Si la requête avait été du genre k-cast ("Me reste-t-il au moins k pantalons ?"), la diffusion de la requête aurait été arrêtée dès qu'un nombre suffisant de réponses eut été collecté.

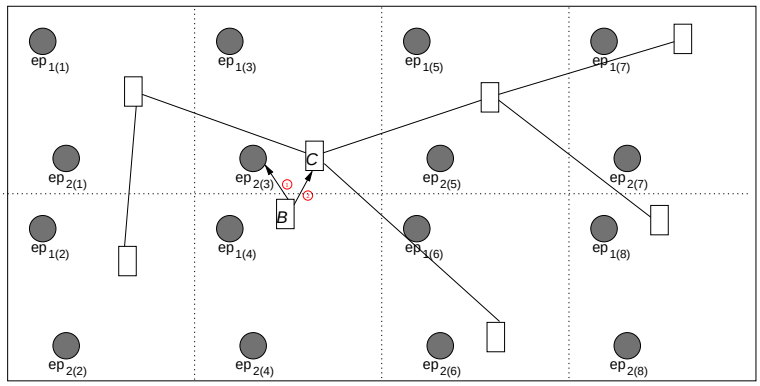


FIGURE 2.8 – SENSATION.

Les résultats de simulation de SENSATION montrent que l'overhead introduit par SENSATION pour enregistrer les données est faible (les données sont en majorité sauvegardées à deux sauts de leur point d'origine) et compensé d'une part par la baisse du trafic nécessaire pour récupérer une information (une information générée par un serveur lointain a pu être stockée plus proche de moi) et l'allègement des serveurs qui reçoivent moins de requêtes (ils ne reçoivent que les requêtes concernant des objets qu'ils stockent). De plus, SENSATION accroît la fiabilité des données (Pour une chute de 50% des bases avec un niveau de réplication seulement égal à 2, seuls 25% des données sont perdues).

2.3 Perspectives

Ainsi les tables de hachage distribuées se trouvent être d'excellents outils pour permettre le passage à l'échelle dans l'Internet des objets dans tout type d'architecture, filaire ou non, et à tout niveau. Nous les avons appliquées aux intergiciels RFID et aux réseaux de capteurs. Une perspective intéressante est dans un premier temps d'étudier la faisabilité de la mise en place d'une table de hachage distribuée globale et commune à tous les services d'abord au sein

d'un système RFID puis de l'étendre à tous les concepts de l'Internet des objets. Ces études vont dans le sens d'une homogénéisation des systèmes de l'Internet des objets, qu'il s'agisse de réseaux de capteurs, de RFID passive mais aussi de téléphone portable ou de codes-barres etc. C'est dans cette optique que j'ai commencé à mener des travaux dans le cadre des projets ICOM [35], ASPIRE [3] et WINGS [78].

2.4 Publications majeures

- N. Mitton and E. Fleury. Distributed node location in clustered multi-hop wireless networks. In *Technologies for Advanced Heterogeneous Networks : First Asian Internet Engineering Conference, (AINTEC 05)*, vol. 3837/2005, p. 112-127, Bangkok, Thailand, December 2005.
- A. Carneiro Viana, N. Mitton, L. Schmidt, and M. Vecchio. A k-layer self-forganizing structure for product management in stock-based networks. In In *Proc. of the 7th IEEE International Conference on e-Business Engineering (ICEBE 2010)*, Shanghai, China, 2010.
- M. Dias De Amorim, S. Fdida, N. Mitton, L. Schmidt, and D. Simplot-Ryl. Distributed planetary object name service : Issues and design principles. Research Report 7042, INRIA, 09 2009.

Chapitre 3

Routage géographique dans les réseaux de capteurs

Comme dans tout réseau, des protocoles de routage doivent être définis. Dans un réseau de capteurs sans fil, il s'agit de routage multi-sauts afin de permettre à chaque capteur de communiquer avec un puits qui n'est pas forcément à portée radio. Un réseau de capteurs est composé d'un ensemble de petites entités (capteurs/nœuds) ayant des capacités limitées en terme de calcul, de mémoire et d'énergie. Ainsi, les protocoles de routage 'classiques' développés pour des réseaux adhoc, qu'ils soient proactifs comme OLSR [13] ou réactifs comme DSR [38] ou AODV [64] ne sont pas adaptés. En effet, ces protocoles génèrent trop de données à stocker ou à envoyer, ce qui sature la bande passante et surtout épuise l'énergie des capteurs prématurément. C'est pourquoi je me suis surtout intéressée aux protocoles de routage géographiques. Ces derniers se basent sur des informations de position pour déterminer le prochain saut. Celle-ci peut être réelle (obtenue par un GPS par exemple) ou 'virtuelle'. Dans ce deuxième cas, il s'agit de définir un système de coordonnées sur lequel appliquer le protocole de routage.

De tels protocoles sont distribués, sans mémoire - *memory-less* - (*aucune information supplémentaire n'a besoin d'être stockée ni sur les nœuds ni dans les messages*) et locaux (*chaque nœud décide du prochain saut en se basant uniquement sur sa position, celle de la destination et celle de ses voisins*). Ces protocoles sont donc adaptés aux contraintes physiques des capteurs tout en permettant le passage à l'échelle.

Je me suis intéressée à la conception d'algorithmes de routage géographique aussi bien pour le cas où les positions GPS étaient disponibles et le cas où elles ne le sont pas, toujours dans un souci d'économie d'énergie et de livraison garantie. La conception des algorithmes que j'ai proposés se base sur l'analyse des forces et faiblesses de l'existant. Ainsi, dans chacune des sections suivantes (Section 3.2 pour les routages basés sur coordonnées réelles section 3.3 pour les routages basés sur coordonnées vituelles), afin de positionner mes travaux et de

comprendre mon cheminement, je présente quelques références de la littérature.

3.1 Préliminaires

Le modèle énergétique considéré dans la suite de ce chapitre est le plus utilisé dans la littérature. Pour envoyer un message au nœud v, le nœud u consommera :

$$energie(|uv|) \quad = \quad |uv|^{\alpha} + c \tag{3.1}$$

où $|uv| > 0$ est la distance euclidienne entre u et v ; c est le coût constant du traitement du signal ; α est une constante réelle (≥ 2) représentant l'atténuation du signal. Ce modèle implique que suivre une succession de courtes arêtes peut être aussi coûteux qu'une succession de longues arêtes à cause de la constante c. Comme démontré dans [67], il existe donc une longueur d'arête optimale en terme de consommation énergétique.

$$\mathrm{r}^* = \sqrt[\alpha]{\frac{\mathrm{c}}{\alpha - 1}} \tag{3.2}$$

Il faut néanmoins noter que les principes des algorithmes proposés ne reposent pas nécessairement sur ce modèle énergétique mais s'appliquent aisément à un modèle quelconque. C'est le cas par exemple du protocole EtE décrit sous-section 3.2.1. Toutefois, certains protocoles utilisent l'existence du "rayon optimal" r^* comme COMNET décrit sous-section 3.2.3 qui peut ne pas exister suivant le modèle énergétique utilisé.

3.2 Routage géographique basé sur coordonnées réelles

Dans cette section, nous nous intéressons aux protocoles de routage géographiques dans les cas où les nœuds connaissent leur position géographique. Cela suppose que les nœuds sont équipés d'un équipement de localisation (GPS ou Galileo) ou qu'ils sont capables d'inférer leur position à partir de leur voisinage.

Les premiers protocoles de la littérature cherchaient à réduire le nombre de sauts vers la destination. Dans *Greedy* [23] par exemple, le nœud portant le message l'envoie à son voisin le plus proche de la destination. Dans MFR (Most Forward Routing), le nœud source u choisit comme relai son voisin v en direction de la destination d dont la distance $|uv'|$ est la plus petite où v' est la projection de v sur (ud) afin de s'assurer qu'on ne s'éloigne pas trop de la ligne droite. Bien d'autres solutions gloutonnes ont été proposées [34] mais ce qui est important de noter et comme cela a été démontré dans [73, 5, 39], pour assurer un chemin sans boucle, le prochain saut doit toujours être choisi en direction de la destination. Bien que simple et facilement implémentables, ces protocoles gloutons ne cherchent pas économiser l'énergie et ne garantissent pas la livraison du message. En effet, bien que le réseau soit connexe, en cas de 'trou

de couverture', un nœud en charge du message peut n'avoir aucun de ses voisins dans la direction de la destination. C'est pourquoi les recherches ont porté sur les deux axes : *(i)* améliorer la consommation énergétique et *(ii)* garantir la livraison du message.

Afin d'améliorer la consommation énergétique, on suppose que les nœuds ont la possibilité d'ajuster leur portée radio. L'idée des algorithmes proposés est de tendre au maximum vers une succession d'arêtes de taille r^* (voir Eq. 3.2). Pour cela, dans COP [40] (cost-over-progress), quand le nœud u doit envoyer un message à la destination d, il envoie à son voisin v tel que le rapport entre le coût ($energie(|uv|)$ Eq. 3.1) sur le progrès parcouru une fois en v ($|ud| - |vd|$) est minimum (v qui minimise $\frac{energie(|uv|)}{|ud|-|vd|}$).

Afin de garantir la livraison du message, les auteurs de [7] ont introduit le *Face Routing*. Le Face routing nécessite que le réseau soit au préalable planarisé, *i.e.* que aucune arête n'en intersecte une autre. Pour cela, plusieurs algorithmes peut être utilisés, comme le graphe de Gabriel (GG) [7, 41] ou le graphe de voisinage relatif (RNG) [74]. Par exemple, le graphe de Gabriel garde une arête entre les nœuds u et v si et seulement si il n'existe aucun nœud à l'intérieur du cercle de diamètre $|uv|$. La planarisation divise le réseau en faces. Le Face routing s'applique sur le graphe planaire. La face contenant la droite (sd), où s est le nœud source et d le nœud destination, est traversée en suivant la règle de la main droite (ou gauche). Quand l'arête qui doit être suivie coupe la droite imaginaire (sd), le message change de face, et ainsi de suite jusqu'à atteindre la destination. La figure 3.1 illustre cet algorithme. Pour envoyer un message à d, s suit la première face (composée des nœuds $sabgh$) et le transmet à a. a applique le même algorithme et transmet à b. Si b continuait à suivre la même face, il enverrait le message à g traversant la droite (sd). Donc, b change de face et envoie le message à c. Et ainsi de suite. Le message arrive finalement à d en suivant le chemin $sabciefd$. Ainsi, bien que garantissant la livraison du message, l'utilisation de Face peut générer de longs chemins, comme par exemple quand s veut atteindre j sur la figure 3.1 (chemin vert). C'est pourquoi, pour éviter ceci, les auteurs de [7] proposent le routage GFG (Greedy-Face-Greedy). Ce dernier applique le routage *greedy* jusqu'à ce que le message soit délivré ou que le routage échoue en un nœud v. Dans ce dernier cas, Face est alors invoqué jusqu'à atteindre un nœud plus proche de la destination que v.

Bien que garantissant la livraison et produisant des chemins de longueur raisonnable, le GFG n'est pas efficace en énergie. C'est pourquoi les auteurs de [69] ont proposé SPFSP (Shortest Path Face Shortest Path), un protocole de type GFG considérant la consommation énergétique. Ce dernier aspect est introduit à travers le calcul d'un plus court chemin en énergie à chaque étape du GFG. Afin d'illustrer ce mécanisme à l'étape gloutonne, prenons l'exemple de la figure 3.2. Le nœud s en charge du message sélectionne d'abord un nœud cible suivant l'algorithme *greedy* [23], *i.e.* son voisin le plus proche de la destination d : nœud b. Plutôt que de transmettre directement le message à b, s calcule le plus court chemin en énergie de s à b parmi son voisinage : $scefb$. s considère également les nœuds c et e dans son calcul alors que ces nœuds ne sont pas

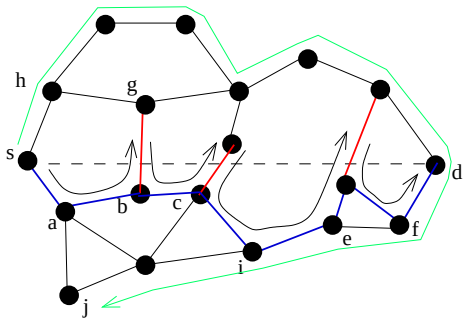


FIGURE 3.1 – Illustration du face routing. Le chemin de s à d apparait en bleu, celui de s à j en vert.

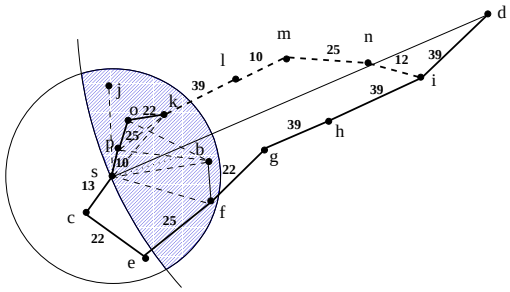
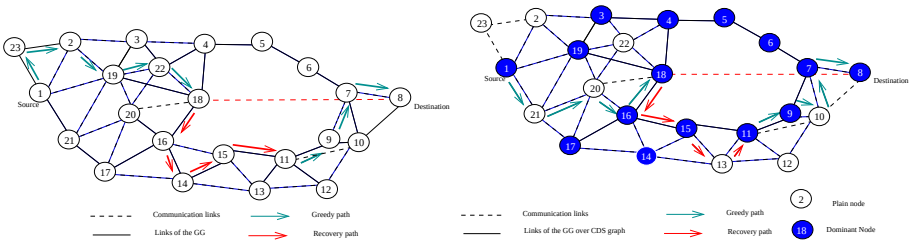


FIGURE 3.2 – Illustration de l'étape gloutonne de SPFSP et EtE.

dans la direction de la destination. Le message est envoyé suivant ce chemin jusqu'à atteindre le nœud f, qui est le premier nœud sur le chemin plus proche de d que s. Afin de s'assurer qu'il n'y aura pas de boucle, s doit inscrire le chemin dans le message. Si le routage échoue, un routage de type Face est utilisé. Si on suppose que le routage échoue au nœud s sur la figure 3.1, s applique Face uniquement pour déterminer le nœud cible a. Mais, là encore, plutôt que d'atteindre a directement s l'atteint en suivant le plus court chemin en énergie. Cependant, les simulations montrent que généralement, ce dernier procédé n'est pas utilisé puisque le plus court chemin en énergie de s à a est généralement l'arête sa elle-même. Cela est dû au fait que la planarisation avec GG conserve seulement des arêtes courtes. Du coup, le routage face est contraint de suivre une succession d'arêtes courtes, ce qui est coûteux en énergie.

3.2.1 Garantie de livraison et économie d'énergie de bout en bout

C'est en partant des observations sur le comportement des algorithmes précédents que j'ai contribué à la conception de EtE (End-To-End). EtE [20, 17, 32] est un protocole GFG qui garantit la livraison du message tout en minimisant



(a) Illustration de SPFSP.

(b) Illustration de EtE.

FIGURE 3.3 – Comparaison de SPFSP etd EtE. Les liens continus sont les arêtes du Graphe de Gabriel. Ce dernier est construit sur l'ensemble des nœuds du réseau dans SPFSP (a), ou seulement sur les nœuds dominants (en bleu) dans EtE (b).

la consommation énergétique à toutes les étapes. EtE se base sur SPFSP [69]. L'étape gloutonne est modifiée en deux points : *(i)* dans la façon dont le nœud cible est sélectionné, *(ii)* sur l'ensemble des nœuds considérés pour le calcul du plus court chemin. En effet, je voulais éviter que le chemin soit inscrit dans le message afin de limiter la consommation de bande passante et garantir les performances de l'algorithme même en cas de réseaux très denses. Pour cela, le nœud en charge du message ne considère pas l'ensemble de son voisinage comme dans SPFSP mais seulement ceux se trouvant en direction de la destination d. Sur la figure 3.2, s ne considère que les nœuds dans la surface hachurée bleue, *i.e.* b, f, j, k, o et p.

La sélection du nœud cible est modifiée de la façon suivante. Plutôt que de considérer le nœud le plus proche de la destination, on le choisit en se basant sur une méthode de coût sur progrès COP [40] où le coût considéré est le coût du plus court chemin en énergie de s vers son voisin u. Soient $v_0v_1...v_iv_{i+1}..v_n$, les nœuds constituant le plus court chemin en énergie de s vers u avec $v_0 = s$ et $v_n = u$. Le coût de ce chemin, noté $energie_{SP}(s, u)$ de s à u est tel que :

$$energie_{SP}(s, u) = \sum_{i=0}^{n-1} energie(|v_iv_{i+1}|).$$

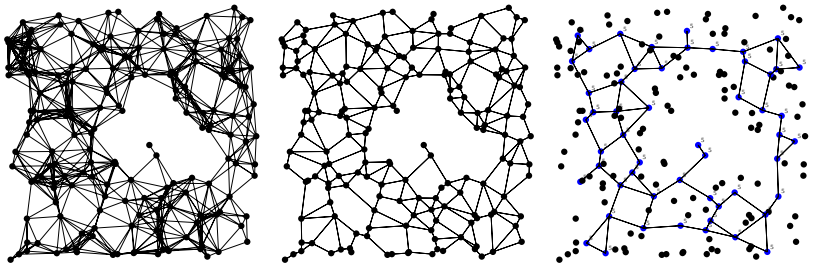
Le nœud s sélectionne donc comme nœud cible le nœud k qui minimise le ratio du coût du plus court chemin entre s et w sur le progrès procuré par k vers la destination d (*i.e.* u sélectionne son voisin w pour lequel $\frac{energie_{SP}(s,k)}{|sd||kd|}$ est le plus petit).

Une fois le nœud cible déterminé, le nœud s transmet le message au premier nœud sur le plus court chemin de s à k, *i.e.* nœud p sur la figure. Le nœud p réitère le même algorithme. Aucune boucle ne peut se produire puisqu'à chaque itération, le prochain saut est calculé parmi les nœuds en direction de la destination. Cet algorithme se répète jusqu'à atteindre la destination ou à atteindre un

nœud dont aucun de ses voisins le rapproche de la destination. Dans ce dernier cas, un routage de type Face mais économe en énergie est invoqué.

Comme déjà mentionné, le routage Face traditionnel garantit la livraison du message mais n'est pas efficace en énergie puisque le message peut suivre une succession de courtes arêtes (comparées à la longueur optimale r^*) puisque GG retire les longues arêtes. Afin de palier ce problème, EtE introduit une étape dans la planarisation du graphe. A partir du graphe original $G = (V, E)$ (Figure 3.4(a)), un ensemble dominant connecté (CDS) est calculé. Les nœuds source s et destination d pouvant ne pas appartenir à cet ensemble, ce dernier est étendu à s et d. Soit $G' = (V', E') \subset G$ où $V' \subset V$ est l'ensemble des nœuds dominants étendu à s et d et $E' \subset E$ est l'ensemble des arêtes de E entre les nœuds de V'. Enfin, puisque le routage Face s'exécute sur un graphe planaire, le graphe de Gabriel $G'' = (V', E'')$ est extrait de G' où $E'' \subset E'$ est l'ensemble des arêtes restantes dans le graphe planaire (Figure 3.4(c)).

Le routage Face de EtE est enfin exécuté sur le graphe G''. Ce routage Face garantit la livraison et ne considère pas de liens courts puisqu'éliminés par la construction du CDS.



(a) Graphe initial G (b) GG sur G (c) GG de G'

FIGURE 3.4 – Lorsque Face est exécuté sur le graphe G (a) (comme SPFSP), les messages suivent les arêtes de G (b) alors que les arêtes de G" (c) sont utilisées dans EtE.

Une fois le graphe G'' extrait, les mêmes mécanismes que dans [69] sont appliqués. Le nœud en charge du message s en mode 'Recovery' applique le routage Face sur G'' seulement pour déterminer le nœud b à atteindre pour joindre la destination. Cependant, l'arête sb n'est pas suivie directement puisque cette arête peut être trop longue d'un point de vue énergétique ($|sb| > r^*$). Le nœud b va donc être atteint à travers un plus court chemin en énergie. Si b est plus près de la destination que le nœud s ayant initié le routage Face, le nœud b repasse en mode glouton. Sinon, il détermine le prochain saut en appliquant le routage face sur le graphe planaire des CDS et l'atteint au travers un plus court chemin, et ainsi de suite...

La figure 3.3(b) illustre un exemple d'exécution de EtE. Le mode glouton fonctionne à partir du nœud 1 qui calcule le coût du plus court chemin vers ses voisins 2, 19 et 21. Le nœud 19 est choisi comme cible temporaire. Pour l'atteindre, 1 envoie le message à 21, premier nœud sur le plus court chemin

en énergie vers 19. Le nœud 21 détermine 20 comme cible et l'atteint via le lien direct, chemin le moins coûteux dans son cas. 20 sélectionne 18 et envoie le message à 16 qui relaie à 18. Le routage glouton s'arrête en 18 puisqu'aucun des voisins de ce dernier le rapproche de la destination. Le routage Face est invoqué pour suivre les arêtes 18-16 (directement), 16-15 (directement) et 15-11 (remplacé par 15-13-11 moins coûteux en énergie). Le routage glouton prend alors le relais en 11 en sélectionnant 10 via 9, 9 sélectionnant 7 délivrant le message à la destination 8.

Ainsi, EtE est le premier algorithme de routage géographique qui garantit la livraison du message et réduit la consommation énergétique dans les deux étapes gloutonne et de 'recovery'.

3.2.2 Anycasting avec garantie de livraison et économie d'énergie

Je me suis également intéressée aux algorithmes anycasting. Dans les routages anycasting, il s'agit d'acheminer les messages d'un capteur vers un puits ou actionneur du réseau, quel qu'il soit. J'ai contribué à la conception du premier algorithme *anycasting* géographique localisé qui garantit la livraison du message et minimise la consommation énergétique dans un réseau multi-puits [62]. Comme dans tout algorithme géographique, on suppose que tout nœud connait sa position, celle de ses voisins et celles des destinations, *i.e.* des puits. Notre proposition se décline en trois variantes : GFGA, COPA, EEGDA, chacune inspirée du GFG [7], de COP [40] et de EtE [20]. Chacune de ces variantes consiste en des phases gloutonnes suivies de 'recovery' quand nécessaire et garantissent la livraison du message sous réserve que la composante connexe du réseau contenant la source du message contienne également au moins un puits. Les variantes de l'algorithme diffèrent de la façon suivante : GFGA utilise comme métrique le nombre de sauts et s'applique quand les nœuds n'ont pas la capacité d'adapter leur portée. COPA et EEGDA cherchent à minimiser la consommation énergétique et diffèrent dans la complexité de calcul qu'elles offrent.

Les trois variantes établissent une route de la source vers l'un des puits. Durant la phase de construction, l'algorithme tente d'atteindre un seul puits. La caractéristique de ces algorithmes est que cette destination peut changer au cours du chemin en fonction de la topologie du réseau. Le routage *Anycasting* commence du nœud portant le message s vers le puits le plus proche de s en distance euclidienne, noté $S(s)$. Cependant, il se peut que s soit déconnecté de $S(s)$ ou plus proche en nombre de sauts d'un autre puits.

Dans la phase gloutonne de GFGA, le nœud source s envoie le message à son voisin v dont la distance à un puits est la plus petite, *i.e.* qui minimise $|vS(v)|$. Dans COPA, la source envoie le message à son voisin v qui obtient le plus faible ratio entre le coût énergétique $energie(|sv|)$ pour atteindre v sur le progrès qu'il apporte vers un puits $(|sS(s)| - |vS(v)|)$. EEGDA améliore le gain en énergie de COPA en déterminant le prochain saut v comme dans EtE, c'est à dire comme étant le nœud qui fournit le plus faible ratio entre le coût en énergie du plus court chemin vers v $(energie_{SP}(s, v))$ sur le progrès qu'il

apporte $(|sS(s)| - |vS(v)|)$. Bien qu'apportant de meilleurs résultats, EEGDA a une complexité de calcul plus importante que COPA. Il faut donc étudier le compromis entre les deux afin de choisir l'algorithme le plus adapté aux besoins de l'application.

Si un nœud s ayant un message à envoyer n'a aucun voisin lui permettant de réduire la distance à un puits, le mode 'recovery' est appliqué. GFGA applique un algorithme FACE classique alors que COPA et EEGDA utilisent le même algorithme que EtE. Le routage repasse en mode glouton dès qu'il atteint un nœud v pour lequel la distance vers au moins un puits est plus petite que la distance séparant le nœud s qui a invoqué le mode 'recovery' de son puits le plus proche.

EEGDA est le premier algorithme de routage anycasting pour réseau de capteurs qui garantisse la livraison du message et réduise la consommation énergétique.

3.2.3 Routage géographique assisté par actionneurs

Les protocoles présentés précédemment sont applicables dans le cas où les capteurs ne contrôlent pas leur mobilité. Les avancées technologiques de ces dernières années nous permettent cependant de voir apparaitre des robots et/ou actionneurs, que l'on peut voir comme des capteurs dont la mobilité contrôlée peut être exploitée pour améliorer les algorithmes et en particulier les algorithmes de routage. Il a d'ailleurs été montré [77] que déployer quelques actionneurs parmi les capteurs apporte les mêmes performances qu'accentuer la densité du réseau de capteurs. Cet aspect est assez nouveau. A ce jour et à ma connaissance, très peu de propositions apparaissent dans la littérature. De plus, les seuls protocoles qui apparaissent proposent des re-positionnements des actionneurs qui génèrent soit des mouvements zig-zag [30] soit des déconnexions du réseau [49].

C'est pourquoi j'ai contribué à la conception du protocole de routage CoMNet [33] (Connectivity preservation Mobile routing protocol for actuator and sensor NETworks) qui prend avantage de la mobilité des actionneurs afin d'optimiser la consommation énergétique du routage. A ce jour, CoMNet est le premier algorithme de routage géographique pour réseau de capteurs et d'actionneurs qui *(i)* considère la consommation énergétique des nœuds à la fois pour envoyer un message et pour se déplacer et *(ii)* préserve la connexité du réseau.

L'idée de CoMNet est la suivante. Un ensemble dominant connecté est calculé. Cette opération s'effectue de façon locale à partir des seules informations de voisinage. L'idée est que seuls les nœuds non dominants peuvent être déplacés et dans la limite des portées radios des nœuds dominants. Ainsi, on garantit qu'à chaque étape du routage, la connexité du réseau est préservée. Puis, lorsqu'un nœud u doit envoyer un message à un nœud d, il calcule pour chacun de ses voisins v en direction de d une position v' où déplacer v (v' pouvant être la position initiale de v si v est dominant) et le ratio coût sur progrès de v. Le coût considéré est la somme du coût d'émission C_e du message vers v plus le coût de déplacement C_m de v en v'. Le progrès considéré est la réduction de la distance

qu'il reste à parcourir pour atteindre d, *i.e.* $|ud| - |v'd|$, une fois v déplacé en v'.

CoMNet se décline en trois variantes qui diffère dans le calcul de la nouvelle position v' de v. Chacune des trois variantes répond à des applications et environnements différents. Les deux premières variantes considèrent le fait qu'un nœud u envoie en général une série de messages à un nœud d, alors que la troisième variante s'applique surtout lorsqu'une route est utilisée occasionnellement.

La première variante, *CoMNet-Orouting* s'applique quand le coût de déplacement est très supérieur au coût d'émission. L'idée est donc de réduire au maximum la distance à parcourir. Pour cela, v' est la projection de v sur la droite (uv). Dans ce cas, v se rapproche de u. C'est pourquoi le message est envoyé après que v se soit déplacé. Le coût d'envoi C_e considéré est donc égal à $C_e = energie(uv') + \epsilon$ où ϵ représente le coût d'envoi d'un beacon à v pour lui dire de se déplacer. Sur la figure 3.5(a), s doit choisir entre ses voisins A_1, A_2 et A_3. Pour cela, pou chacun d'eux, il calcule le coût de déplacement de ce nœud sur la droite sd (noté en rouge sur la figure), plus le coût d'envoi d'un message vers la prochaine position de ces nœuds (noté en bleu sur la figure) sur le progrès vers la destination (noté en noir sur la figure). Il obtient un ratio de $\frac{1+1}{1} = 2$ pour A_1, de $\frac{2+1}{2} = 1.5$ pour A_2 et de $\frac{5+2}{3} = 2.3$ pour A_3. s va donc envoyer un message à A_2 qui se déplacera en A'_2.

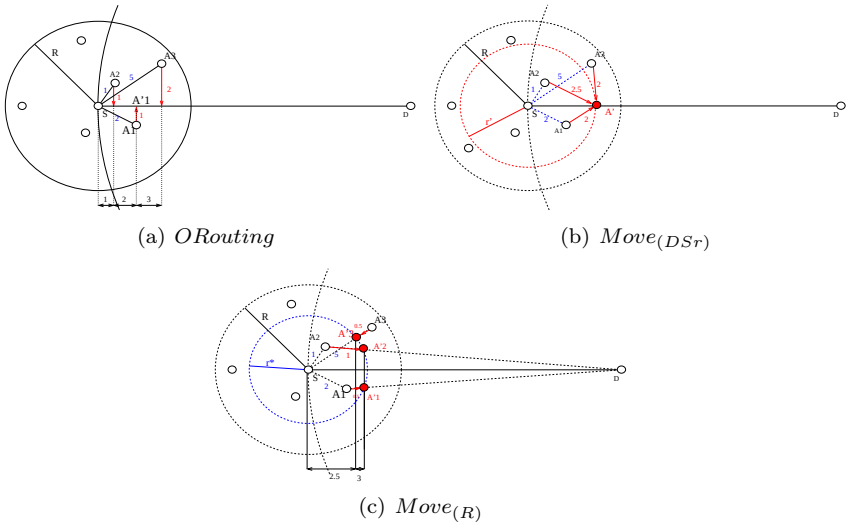
La seconde variante, *CoMNet-Move*$_{DSR}$, s'applique quand émettre coûte beaucoup plus cher que se déplacer. Auquel cas, on va chercher à faire en sorte que la distance entre deux nœuds soit égale à la longueur optimale de transmission (voir Eq. 3.2). Pour cela, on va chercher à aligner les nœuds entre u et d en les espaçant d'une distance proche de r^*, *i.e.* r' tel que

$$r' = \begin{cases} \frac{|SD|}{n^*} & \text{if } r^* - \frac{|SD|}{n^*} < \frac{|SD|}{n^*-1} - r^* \\ \frac{|SD|}{n^*-1} & \text{sinon} \end{cases}$$

Le message est envoyé à v avant que v ne se déplace et le coût d'envoi considéré est $energie(|uv|)$. Sur la figure 3.5(b), s doit choisir entre ses voisins A_1, A_2 et A_3. Pour cela, pour chacun d'eux, il calcule le coût de déplacement de ce nœud vers la position A' telle que $|sA'| = r^*|$ (noté en rouge sur la figure), plus le coût d'envoi d'un message vers le nœud avant qu'il se déplace (noté en bleu sur la figure). Le progrès est le même quel que soit le nœud choisi puisque tous se déplacent en A'. s obtient un coût de $1 + 2.5 = 3.5$ pour A_1, de $2 + 2 = 4$ pour A_2 et de $5 + 2 = 7$ pour A_3. s va donc envoyer un message à A_1 qui se déplacera en A'.

Enfin, la troisième variante, *CoMNet-Move*$_R$ s'applique quand la route n'est utilisée qu'occasionnellement. Elle va chercher à optimiser les transmissions et à limiter les déplacements. Pour cela v va se déplacer à une distance r^* de u sur la droite (vd). Le message est envoyé à v avant que v ne se déplace et le coût d'envoi considéré est $energie(|uv|)$. Sur la figure 3.5(c), s doit choisir entre ses voisins A_1, A_2 et A_3. Pour cela, pou chacun d'eux, il calcule le coût de déplacement de ce nœud à l'intersection du cercle de rayon r^* centré en s et soit de la droite (Ad) si $|sA| < r^*$ soit de la droite (sA) (noté en rouge sur la

figure), plus le coût d'envoi d'un message vers le nœud avant qu'il se déplace (noté en bleu sur la figure) sur le progrès vers la destination (noté en noir sur la figure). Il obtient un ratio de $\frac{2+0.5}{3} = 0.83$ pour A_1, de $\frac{1+1}{3} = 0.66$ pour A_2 et de $\frac{5+0.5}{2.5} = 2.2$ pour A_3. s va donc envoyer un message à A_2 qui se déplacera en A'_2.



(a) *ORouting* (b) $Move_{(DSr)}$

(c) $Move_{(R)}$

FIGURE 3.5 – Illustration de CoMNet. Les flèches rouges montrent les déplacements possibles des nœuds avec le poids associé. Les liens bleus représentent les envois associés de leur coût.

3.3 Routage géographique basé sur coordonnées virtuelles

Ainsi, les protocoles de routage géographiques présentent des solutions de routage très prometteuses pour les réseaux de capteurs sans fil. Cependant, ils nécessitent la connaissance de coordonnées géographiques qui ne sont pas toujours disponibles. En effet, les équipements GPS restent coûteux, consommateurs d'énergie et ne fonctionnent pas en environnement intérieur. C'est pourquoi ces dernières années ont vu apparaitre des protocoles qui attribuent des coordonnées virtuelles aux nœuds pour ensuite proposer un routage géographique sur ces dernières.

La méthode la plus répandue pour attribuer des coordonnées 'virtuelles' suppose l'existence d'ancres ou de *landmarks* dans le réseau. Ces ancres sont des points fixes qui à l'initialisation, diffuse un message balise contenant sa distance à l'ancre en nombre de sauts (donc 0). Chaque nœud le recevant incrémente la distance à l'ancre dans le message avant de le relayer. Ainsi, à la fin de cette phase, chaque nœud u construit un vecteur de coordonnées

$V(u) = [v_u^1, v_u^2, \ldots, v_u^n]^{-1}$ à n dimensions où n est le nombre d'ancres et v_u^i est la distance en nombre de sauts le séparant de l'ancre i. La figure 3.6(a) montre un exemple de système de coordonnées avec 3 ancres. Le nœud 4 par exemple a pour coordonnées $(2, 2, 3)$ car il se situe respectivement à 2, 2 et 3 sauts des ancres $L1$, $L2$ et $L3$.

Le vecteur V peut alors être utilisé tel quel comme coordonnées (comme dans la plupart des solutions) ou servir de base au calcul de celles-ci. Par exemple, GLIDER [22] calculera les coordonnées $\Gamma_u = \Gamma_u^1, \Gamma_u^2, \ldots, \Gamma_u^n$ du nœud u de la façon suivante :

$$\Gamma_u^i = (v_u^i)^2 - \mu$$

avec $\mu = \frac{1}{n} \sum_1^n (v_u^i)^2$.

Une fois les coordonnées calculées, afin de pouvoir établir un routage géographique, il faut choisir une métrique de distance afin que chaque nœud soit capable de calculer la distance entre lui-même ou ses voisins et la destination.

VCap [9] et JUMPS [6] par exemple, utilisent la distance de Hamming, *i.e.* la distance de Hamming $d_h(u, v)$ entre les nœuds u et v est telle que

$$d_h(u, v) = \sum_{i=1}^{n} |v_u^i - v_v^i|$$

VCap et JUMPS appliquent alors l'algorithme *Greedy* [23] vu précédemment dans le cas où les coordonnées géographiques sont disponibles sur cet espace de coordonnées. Ainsi, dans VCap, un nœud u devant envoyer un message au nœud d l'enverra à son voisin v tel que $d_h(v, d)$ est minimum. GLIDER [22] choisit également le voisin le plus proche de la destination mais en considérant son système de coordonnées et une pseudo-distance euclidienne $d_{se}(v, d) = \sum_{i=1}^{n} \left(v_u^i - v_d^i\right)^2$ entre v et d.

Bien d'autres métriques et calcul de coordonnées basées sur les distances aux ancres ont été proposées [50] que je ne détaillerai pas dans ce document. J'ai cependant montré [19, 18, 60] que la distance de Hamming appliquée sur le système de coordonnées V (distance aux ancres en nombre de sauts) s'avéraient être les plus efficaces non seulement en termes de complexité de calcul et de mémoire mais également au regard des taux de livraison du message.

Bien que présentant des avantages (simplicité, routage géographique même si coordonnées non disponibles, etc), ces algorithmes de routage ne considèrent pas la consommation énergétique des nœuds ni ne garantissent la livraison du message. C'est pourquoi, je me suis intéressée à la conception des algorithmes suivants : VCost [18] qui prend en compte les consommation d'énergie, LTP [12] qui garantit la livraison du message et enfin HECTOR [61] qui combine les avantages de LTP et VCost.

3.3.1 Économie d'énergie

De la même façon que VCap [9] transpose l'algorithme [23] sur coordonnées virtuelles, VCost [18] transpose l'algorithme COP [40]. En effet, dans VCost,

on suppose que les nœuds sont capables d'estimer la distance euclidienne qui les sépare de leurs voisins. VCost utilise le même système de coordonnées que VCap et la distance de Hamming. Ainsi, un nœud s porteur d'un message pour le nœud d va l'envoyer à son voisin v tel que le ratio $\frac{energie(|sv|)}{d_h(s,d)-d_h(v,d)}$ soit minimum.

VCost obtient les mêmes taux de livraison que VCap pour une consommation énergétique diminuée de 30%.

Bien que simple à mettre en place, un système de coordonnées basé sur la distance à des ancres ne permet pas à lui seul de garantir la livraison du message. En effet, un tel système perd l'unicité des coordonnées (plusieurs nœuds peuvent avoir des coordonnées identiques en étant très éloignés l'un de l'autre). De la même façon, contrairement à la distance euclidienne qui donne des valeurs réelles, la distance de Hamming retourne des valeurs entières, ce qui génère plus de distances identiques entre deux voisins et oblige l'algorithme à choisir au hasard. C'est pourquoi, au travers la conception de LTP (Labeled Tree Protocol), je me suis intéressée à l'élaboration d'un autre système de coordonnées qui permet de palier les biais évoqués ci-dessus.

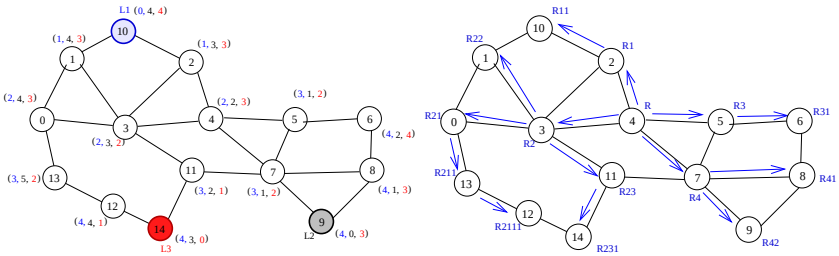
3.3.2 Garantir la livraison

LTP (Labeled Tree Protocol) construit un système de coordonnées virtuelles basé sur la construction d'un arbre. Les coordonnées ou étiquettes sont attribuées aux nœuds conjointement avec la construction de l'arbre. A l'initialisation, un nœud prend le rôle de racine. Ce nœud peut être une ancre ou n'importe quel nœud du réseau. Ce nœud est étiqueté R (comme Racine). Puis, à chaque étape, chaque nœud fraîchement étiqueté demande à ses voisins non étiquetés de se manifester. Il leur attribue alors à chacun une étiquette. Si $l(u)$ est l'étiquette (ou label) du nœud u, le k^{th} voisin non étiqueté de u se verra attribuer l'étiquette $l(u)k$. La figure 3.6(b) illustre l'étiquetage du réseau où la racine est le nœud 4 qui a l'étiquette R. Le nœud 13 est étiqueté $R211$ puisque premier fils du nœud 0 étiqueté $R21$. L'arbre construit les plus courts chemins en nombre de sauts entre la racine et n'importe quel autre nœud. L'étiquetage ainsi construit intègre les routes entre toute paire de nœuds du réseau, le chemin dans l'arbre qui est unique et existe toujours tant que le graphe sous-jacent est connexe.

La distance utilisée dans l'arbre est basée sur la taille de l'étiquette et du préfixe commun entre les étiquettes de deux nœuds, qui donnent la distance en nombre de sauts dans l'arbre entre ces nœuds. Ainsi, la distance entre le nœud a et le nœud b est $d_T(a,b) = ||l(a)| - |l(c)|| + ||l(c)| - |l(b)||$ où c est le premier ancêtre commun entre a et b et $|l(a)|$ est la taille de $l(a)$. Sur la figure 3.6(b) la distance entre les nœuds 9 et 5 est donc $d_T(9,5) = ||l(9)| - |l(4)|| + ||l(4)| - |l(5)|| = |3-1| + |1-2| = 3$.

Une fois le système de coordonnées et le calcul de distance établis, LTP effectue un routage glouton. Un nœud u souhaitant envoyer un message à un nœud d l'envoie à son voisin v tel que $d_T(v,d)$ est la plus petite. Ainsi, dans le pire des cas, le message suivra l'arbre jusqu'à atteindre sa destination mais peut très bien suivre des 'raccourcis', liens entre les branches de l'arbre.

Cet algorithme a le grand avantage de garantir la livraison. Mais il souffre de deux inconvénients. Le premier est que le message peut suivre des routes très longues inutilement, ce qui est consommateur d'énergie et de bande passante. Un moyen de palier à la longueur des routes est de construire plusieurs arbres. A chaque saut, le nœud en charge du message l'envoie à son voisin dont la distance à la destination est la plus courte quelque soit l'arbre (un message peut changer d'arbre au cours du chemin). Cependant, cela ne répond pas à la consommation d'énergie. Cet inconvénient est résolu au travers de l'algorithme HECTOR [61]. Le second inconvénient tient dans le fait que la taille des étiquettes n'est pas bornée et peut être arbitrairement longs, ce qui nécessite de l'espace mémoire sur les nœuds et des paquets de données plus longs. Des solutions ont été apportées dans [11].



(a) Illustration des coordonnées dans VCap ou VCost (b) Illustration de l'étiquetage dans LTP

FIGURE 3.6 – Assigner des coordonnées virtuelles avec VCost (a) ou LTP (b). HECTOR attribue les deux systèmes à chaque nœud.

3.3.3 Garantir la livraison en économisant l'énergie

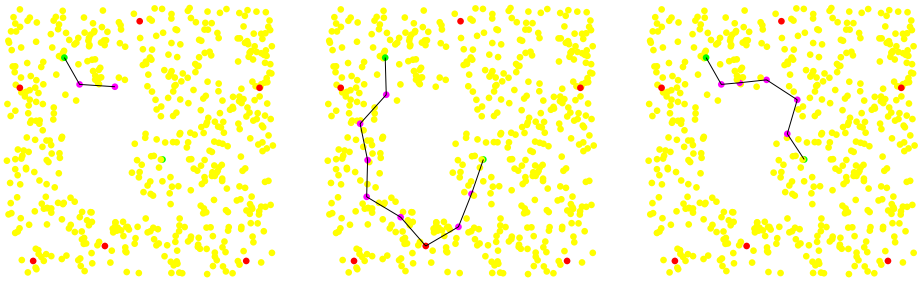
Comme constaté précédemment, les protocoles de routage géographique basés sur coordonnées virtuelles ne permettent pas à la fois de garantir la livraison du message et d'économiser l'énergie. C'est pourquoi, en me basant sur l'étude des avantages de VCost et LTP, j'ai participé à l'élaboration de HECTOR. HECTOR [61] - Hybrid Energy effiCient Tree-based Optimized Routing - est à ma connaissance le premier algorithme de routage géographique basé sur coordonnées virtuelles qui à la fois garantit la livraison du message et cherche à réduire la consommation énergétique. Pour cela, HECTOR se base sur un double système de coordonnées, un système basé sur un arbre (les coordonnées T), comme dans LTP et un issu de distances à des ancres (les coordonnées V), comme dans VCost.

HECTOR utilise deux mesures de distance, une pour chaque système de coordonnées : la distance de Hamming d_V (définie en section 3.3) pour mesurer un progrès sur les coordonnées de type VCost, et la d_T distance définie en section 3.3.2 pour évaluer les distances sur les coordonnées de type LTP. Ainsi, les décisions de routage sont prises de façon à ce que le prochain saut sur la route

fournisse toujours un progrès (positif voire nul) par rapport aux coordonnées T. En effet, les coordonnées T et la distance associée sont utilisées pour assurer la livraison du message. Les coordonnées V sont utilisées pour réduire le facteur d'élongation de la route.

Un nœud u en charge d'un message pour le nœud d va procéder de la façon suivante. u considère dans un premier temps l'ensemble de ses voisins v lui permettant de réduire la distance d_V ($d_V(v,d) < d_V(u,d)$) et de réduire ou égaler la distance d_T ($d_T(v,d) \leq d_T(u,d)$) vers la destination d. Soit N' cet ensemble de nœuds. Si $N' \neq \emptyset$, u envoie le message au nœud $w \in N'$ qui a le ratio coût $energie(|uw|)$ sur progrès sur coordonnées V ($d_V(u,d) - d_V(w,d)$) le plus faible. Si $N' = \emptyset$, u considère uniquement les nœuds v lui permettant de réduire la distance T vers d ($v|d_T(u,d) > d_T(v,d)$). Il existe toujours au moins un nœud v dans ce dernier cas du fait de la construction de l'arbre. u envoie alors le message au nœud v qui a le ratio coût $energie(|uv|)$ sur progrès sur coordonnées T ($d_T(u,d) - d_T(w,d)$) le plus faible.

Le comportement de HECTOR est illustré par la figure 3.6. Prenons le cas où le nœud 14 (L_3) veut envoyer un message au nœud 5. Dans ce cas, dans VCost, le nœud 14 envoie le message au nœud 11 où le routage échoue puisque 11 n'a aucun voisin lui permettant de réduire la distance d_V vers le nœud 5 ($d_V(11,5) = d_V(7,5)$. HECTOR permet d'éviter ce genre de situation en assurant un progrès au regard des coordonnées T. Ainsi, avec HECTOR, le nœud 11 choisit le nœud 3 puisque $d_T(3,5) < d_T(11,5)$. Le message va alors suivre le chemin $14-11-3-4-5$.



(a) VCost (b) LTP (c) HECTOR

FIGURE 3.7 – Illustration des chemins suivis par chaque algorithme. VCost échoue après deux sauts, LTP remonte jusqu'à la racine de l'arbre. HECTOR, en combinant les deux, livre le message en suivant un chemin plus direct.

La figure 3.7 montre des chemins suivis par les différents algorithmes. HECTOR permet de délivrer le message en suivant un chemin assez direct et en optimisant la dépense énergétique au long du chemin.

Il faut noter également que ce même algorithme s'applique lorsque les nœuds ne sont pas capables d'adapter leur portée (on a alors $energie(|uv|) = 1 \, \forall u, v$). Le système de coordonnées V peut être remplacé par tout autre type de coordonnées, y compris des coordonnées géographiques. L'arbre T permet d'assurer

la garantie de livraison du message.

Les difficultés de HECTOR résident dans la maintenance des systèmes de coordonnées, en particulier celui basé sur l'arbre, qui limite son utilisation en cas de mobilité des nœuds.

3.4 Perspectives

Ainsi, les routages géographiques sont adaptés aux réseaux de capteurs de par leurs aspects sans mémoire, local et distribué. Dans ce domaine, j'ai proposé de nouveaux protocoles cherchant à garantir la livraison du message tout en optimisant la consommation énergétique, et ce, que les coordonnées géographiques soient connues ou non. Ces algorithmes ont chacun été novateurs et performants dans leur catégorie vis à vis des algorithmes de la littérature. Cependant, bien que performants, ces algorithmes souffrent du manque de mise en situation réelle. Ce problème est récurrent dans la littérature du fait du caractère fastidieux des expérimentations et coûteux d'un point de vue matériel.

Maintenant, avec la mise en place de la plateforme SensLAB [1], je compte mettre l'accent sur l'expérimentation. C'est ce que j'ai commencé à étudier au travers de la thèse de Tony Ducrocq, des études publiées dans [51, 14, 16] ainsi qu'au travers de projets collaboratifs comme Intelligent Data Center ou SVP [2]. L'idée serait d'étudier l'impact de la couche radio réelle sur le fonctionnement et le comportement des algorithmes. Ces observations devraient permettre d'extraire des modèles ou des informations directement liées à la couche physique et de les intégrer dans la conception des protocoles afin de les rendre les plus robustes possible face à ces aléas. La conception des algorithmes devra également considérer le fait que les informations dont disposent les nœuds peuvent être erronées (du fait des aléas radio et de la mobilité subie des nœuds) et de s'adapter en conséquence.

Une autre orientation à prendre est l'intégration d'actionneurs dont la mobilité est contrôlée et permet d'améliorer les performances du routage. C'est ce que j'ai commencé à étudier au travers de CoMNet (voir section 3.2.3) et que je compte poursuivre au travers la thèse de Nicolas Gouvy [3]. En effet, CoMNet ne s'intéresse qu'à la partie gloutonne de l'algorithme de routage. Il ne permet pas de garantir la livraison du message.

3.5 Publications majeures

– C. Burin des Roziers, G. Chelius, T. Ducrocq, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, T. Noel, and J. Vandaele. Using SensLAB as a first class scientific tool for large scale wireless sensor network experiments. In *Proc. of the IFIP/TC6 NETWORKING 2011*, Valencia, Spain, 2011. to appear.

1. www.senslab.info
2. http ://surveiller-prevenir.fr
3. www.lifl.fr/ gouvy

– E. Chávez, N. Mitton, and H. Tejeda. Routing in wireless networks with position trees. In *Proc. of the 6th International Conference on AD-HOC Networks & Wireless (Ad Hoc Now 07)*, Morelia, Mexico, September 2007.
– E. Elhafsi, N. Mitton, B. Pavkovic, and D. Simplot-Ryl. Energy-aware georouting with guaranteed delivery in wireless sensor networks with obstacles. *International Journal of Wireless Information Networks*, 16(3) :142-153, 2009.
– N. Mitton, D. Simplot-Ryl, and I. Stojmenovic. Guaranteed delivery for geographical anycasting in wireless multi-sink sensor and sensor-actor networks. In *Proc. of the 28th Annual IEEE Conf. on Computer Communications (INFOCOM 2009)*, Rio de Janeiro, Brazil, April 2009. Short paper.
– N.Mitton, T. Razafindralambo, D. Simplot-Ryl, and I. Stojmenovic. Hector is an energy efficient tree-based optimized routing protocol for wireless networks. In *Proc. of 4th the Int. Conf. on Mobile Ad-hoc and Sensor Networks (MSN 2008)*, Wuhan, China, December 2008.

Chapitre 4

Contrôle de topologie et mobilité

Afin de pouvoir auto-organiser un réseau sans fil et appliquer des algorithmes efficaces, il faut tout d'abord récupérer les informations relatives à la topologie du réseau. Un nœud u du réseau doit pouvoir efficacement identifier ses voisins, *i.e.* les nœuds avec qui il partage un lien radio et communiquer directement. Comme nous le verrons en section 4.1, avoir des tables de voisinage consistantes en présence de mobilité des nœuds n'est pas trivial. Une fois ce voisinage acquis, afin de permettre le passage à l'échelle et assurer de meilleures performances aux algorithmes reposant sur ces informations de voisinage, le nœud u peut appliquer un contrôle de topologie et ne considérer qu'un sous-ensemble de ces liens (section 4.2). Lorsque des actionneurs mobiles sont disponibles, ils peuvent être d'une grande aide pour contrôler la topologie du réseau et localiser les capteurs 4.3. On peut en effet couvrir une zone en déployant les actionneurs (section 4.4.1) ou visiter les capteurs dont la mobilité n'est pas contrôlée afin de prélever les données qu'ils ont à fournir (section 4.4.2).

4.1 Découvrir son voisinage

4.1.1 Sans information de position

La découverte de voisinage est un mécanisme qui permet à chaque nœud d'établir la liste des autres nœuds à portée radio. Cette étape est très importante dans la vie d'un réseau puisque tous les algorithmes de niveau supérieur (routage, ordonnancement, etc) reposent sur les tables de voisinage construites par ces algorithmes.

La découverte de voisinage se fait traditionnellement au travers de l'envoi périodique d'un message HELLO. Sur réception d'un tel message envoyé par le nœud v, le nœud u apprend l'existence de v en tant que son voisin et l'inscrit dans sa table de voisinage, ou met à jour l'entrée correspondante si v y apparait

déjà. Lorsqu'aucun message de v n'est reçu par u au bout d'un certain temps (généralement à 3 fois la fréquence d'envoi des messages HELLO), u efface v de la liste de ses voisins.

Les verrous consistent en l'adaptation des fréquences d'envoi et de rafraîchissement de la table afin d'avoir des tables à jour même en cas de mobilité des nœuds sans saturation des ressources. Les fréquences d'envoi de ces messages et celle de rafraîchissement de la table de voisinage doivent s'adapter à leur environnement car envoyer trop de messages pourrait saturer inutilement la bande passante alors que en envoyer trop peu ne permettrait pas de détecter tous les voisins. De la même façon, effacer trop vite des entrées de la table de voisinage pourrait effacer des liens toujours existants et ne pas rafraîchir assez souvent conduirait à des tables obsolètes dans lesquelles figureraient encore des voisins hors de portée.

TAP [36, 37] (Turn-Over based Adaptive Hello Protocol) est un protocole qui adapte dynamiquement la fréquence d'envoi des messages HELLO en fonction de la mobilité des nœuds. Un nœud u va donc observer son voisinage et comptabiliser le nombre de nœuds N_{new} qui apparaissent dans son voisinage durant une période de temps Δt. u en déduit une valeur de 'turnover' définie comme le ratio r entre N_{new} et la taille de son voisinage :

$$r = N_{new} \times \frac{1}{f_{\text{HELLO} \times time}}, \tag{4.1}$$

où f_{HELLO} est la fréquence d'envoi des messages HELLO de u et $time$ la période à laquelle r est calculé. L'idée est que plus un nœud est mobile par rapport à son voisinage, plus ce dernier changera et donc, plus r sera influencé.

Dans TAP, nous établissons par une étude stochastique le ratio optimal r_{opt} à viser qui fournirait le meilleur compromis entre consistance des tables de voisinage et fréquence d'envoi des messages (voir [36] pour plus de détails). Il s'avère que r_{opt} est indépendant de la vitesse absolue des nœuds. Afin d'atteindre une fréquence d'envoi optimale par rapport à sa mobilité relative, chaque nœud u compare la valeur de son r à r_{opt}. Si $r > r_{opt}$, cela signifie que la fréquence actuelle du nœud u est trop faible. A l'inverse, si $r < r_opt$, le fréquence d'envoi peut être réduite. u va donc adapter sa fréquence d'envoi de la façon suivante :

$$f_{\text{HELLO}} = \left\{ \begin{array}{ll} f_{\text{HELLO}} - \frac{f_{\text{HELLO}}}{\alpha} \times g(r) & \text{if } r \leq r_{\text{opt}}, \\ f_{\text{HELLO}} + \frac{f_{\text{HELLO}}}{\alpha} \times g(r) & \text{sinon.} \end{array} \right\} \tag{4.2}$$

où α est une constante permettant de contrôler la vitesse de convergence, et $g(r)$ est une fonction du turnover observé r qui permet de spécifier comment la fréquence f_{HELLO} doit être modifiée. La fonction g utilisée doit être telle que $g(r_{opt}) = 0$ (on ne modifie pas une fréquence HELLO qui conduit déjà à un turnover optimal) et telle que plus r est différent de r_{opt}, plus $g(r)$ doit être élevé afin de faire converger plus rapidement la fréquence d'envoi. Plusieurs fonctions peuvent donc être utilisées. Dans l'évaluation de TAP, nous avons utilisé la

fonction g la plus simple répondant aux critères ci-dessus :

$$g(r) = \begin{cases} (\frac{r - r_{\text{opt}}}{r_{\text{opt}}})^2 & \text{si } r < 2 \times r_{\text{opt}}, \\ 1 & \text{sinon.} \end{cases} \tag{4.3}$$

Les résultats montrent que TAP stabilise rapidement et que les tables de voisinage sont précises à 96%. NLA [1] (Neighborhood Lifetime Algorithm) se base sur TAP et permet d'adapter dynamiquement la fréquence de rafraichissement des tables de voisinage. NLA se base sur l'historique de chaque voisin et sa fréquence d'envoi de messages HELLO. Dans TAP, plus la fréquence d'envoi est élevée, plus le nœud est mobile et donc, plus il pourra être retiré des tables rapidement.

4.1.2 Avec des informations de position

TAP est ainsi le premier algorithme de découverte de voisinage sachant adapter dynamiquement la fréquence d'envoi des messages HELLO sans utiliser d'équipement de localisation comme les GPS. Dans ARH [46], nous exploitons les bénéfices que peuvent nous fournir les informations de localisation. Dans ARH, un paquet HELLO contient la position du nœud. Chaque nœud u prédit sa propre position future à chaque intervalle de temps en se basant sur un modèle ARMA simplifié. Nous utilisons le modèle proposé en [26]. Si sa prédiction s'avère être différente de sa position réelle, u envoie un nouveau message HELLO contenant sa position. Chacun de ses voisins v aura effectué le même calcul de prédiction de position sur u. Ils supposeront cette position exacte tant qu'ils ne reçoivent pas de nouveaux messages HELLO du nœud u. Sur réception d'un message HELLO de u, v corrigera son modèle ARMA suivant [26] et renseignera la nouvelle position de u dans sa table de routage. Ainsi, chaque nœud connait la position de ses voisins à tout moment, même s'ils n'émettent pas de messages HELLO et peut en déduire si ces derniers se trouvent toujours dans son voisinage. La figure 4.1 illustre l'algorithme déroulé par les nœuds dans ARH.

Les erreurs sont limitées. En effet, si u suppose que v a quitté son voisinage alors que v est toujours à portée radio, le nœud v qui calcule la même prédiction que u va se rendre compte de l'erreur et envoyer un message HELLO signalant ainsi sa présence. u corrige alors sa table et les paramètres de son modèle de prédiction pour v. Cependant, si u suppose v dans son voisinage alors que celui-ci l'a quitté, alors, même si v se rend compte de l'erreur et émet un message HELLO, le nœud u ne le recevra pas et maintiendra v dans son voisinage. Une solution à ce problème serait d'émettre un message HELLO à fréquence minimale. Cependant, cela revient à la problématique de base qui est 'quelle sera cette fréquence minimale ?'. Ainsi, afin de répondre à ce problème, chaque nœud u va également maintenir une autre suite ARMA simplifiée qui prédit le temps entre deux messages HELLO de son voisin v qui va lui-même maintenir ce même modèle pour lui-même. Ainsi, sur non-réception d'un message HELLO de v, u considère que ses modèles sont justes (sinon, v aurait envoyé un message) ou

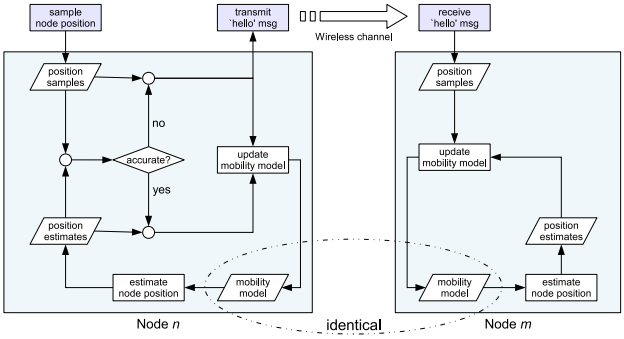


FIGURE 4.1 – Exécution de ARH. Le nœud n transmet un message HELLO 'hello' message en fonction de la précision de l'estimation de sa position calculée sur un modèle ARMA simplifié. Le nœud m maintient un modèle pour n identique.

que v n'est plus dans son voisinage (même si v envoie un message pour corriger son modèle, u ne le reçoit pas).

Les résultats montrent que ARH, en exploitant les information de position des nœuds, permet d'obtenir des tables de voisinage aussi précises que TAP tout en envoyant moitié moins de messages HELLO. Ainsi, utiliser les informations de localisation apporte un gain significatif en énergie et bande passante (moins de messages) mais nécessite une plus grande complexité mémoire et de calcul afin de maintenir les suites ARMA pour chaque voisin plus pour lui-même.

4.2 Planariser un graphe

Utiliser un graphe planaire (où aucune paire d'arêtes ne s'intersecte) est d'une grande aide dans beaucoup d'algorithmes de réseaux de capteurs. Comme on a pu le voir dans le chapitre 3, toutes les solutions permettant de garantir la livraison d'un message dans le cadre d'un routage géographique repose sur l'hypothèse que le graphe a été au préalable planarisé. Habituellement, les techniques utilisées sont le Graphe de Gabriel [7] ou le graphe de voisinage relatif (RNG) [74]. Ces techniques ont les avantages d'être locales (chaque nœud n'a besoin de connaître que la position de ses voisins) et distribuées.

Le degré $\Delta(G)$ d'un graphe G est le degré maximal des nœuds. On cherche souvent à garder $\Delta(G)$ aussi petit que possible et à le borner par une constante. En effet, un degré faible en communications sans fil implique moins de contention et d'interférences. Les techniques de planarisation locale citée ci-dessus produisent des graphes de degré non borné. Le RNG original a été modifié par Li [48] pour palier ce problème nécessitant que chaque nœud ait un identifiant unique, ce qui n'est pas toujours faisable en réseau sans fil.

Avec Hypocomb [47], nous avons voulu proposer une nouvelle famille de graphes planaires géométriques, complètement différents des graphes connus

précédemment.

4.2.1 Un graphe planaire

Hypocomb se base sur un graphe que nous appelons *Besh (Blocked-mesh)*. Etant donné un ensemble de sommets dans le plan euclidien, Besh se construit en tirant des rayons de façon synchrone depuis chaque sommet v dans les quatre directions nord, sud, est, ouest notés respectivement R_v^{north}, R_v^{west}, R_v^{south} et R_v^{east}. Les rayons se bloquent lorsqu'ils se rencontrent. Les règles de blocage sont définies comme suit : Soit $T = \{north, west, south, east\}$. Pour chaque $dir \in T$, $\overline{dir}$ est la direction opposée de dir et $\hat{dir}$ est l'ensemble des directions orthogonales à dir. Par exemple, si $dir = nord$, alors $\overline{dir} = sud$ et $\hat{dir} = \{ouest, est\}$.

Definition 1 (Règles de blocage) *$\forall a, b \in V$, $a \neq b$ et $\forall dir, dir' \in T$, $dir \neq dir'$, si R_a^{dir} et $R_b^{dir'}$ se rencontrent en un point u, R_a^{dir} est bloqué seulement dans l'un des cas suivants :*

1. *$|au| > |bu|$;*
2. *$|au| = |bu|$, $dir' \in \hat{dir}$ et $dir = east$ or $west$;*
3. *$|au| = |bu|$ et $dir' = \overline{dir}$.*

Ainsi, le rayon le plus long est bloqué ou, en cas de rayons de même longueur, les rayons horizontaux bloquent les rayons verticaux. On dit alors que 'b bloque a en u' si les rayons de a et b se rencontrent au point u.

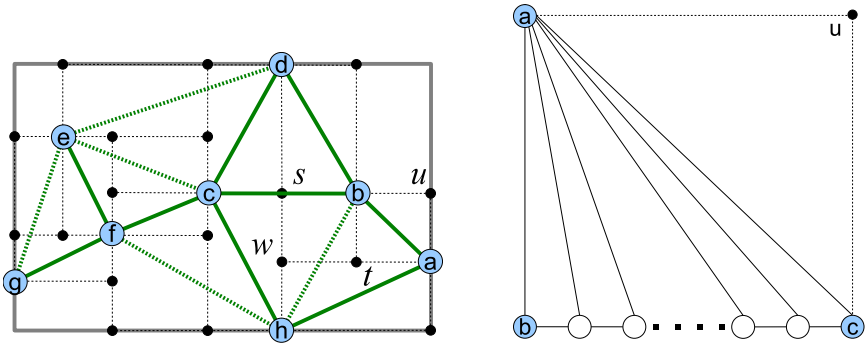
La figure 4.2(a) illustre la structure d'un Besh créé à partir de 8 sommets $a, b, \ldots, h$, dont la bordure est définie par le rectangle gris. Les ronds pleins représentent les nœuds du Besh, les lignes en pointillé sont les arêtes du Besh. Ainsi, b bloque a en u parce que les rayons issus de a et b sont d'égales longueur en u et que le rayon est de b est horizontal et donc prioritaire. b et c se bloquent mutuellement en s car de directions opposées.

Une fois Besh établi, le graphe Hypocomb est construit. Une arête existe dans Hypocomb entre les nœuds a et b si et seulement si il existe une relation de blocage entre a et b, cad si a bloque b ou b bloque a ou si a et b se bloquent mutuellement. Les arêtes de Hypocomb apparaissent en vert sur la figure 4.2(a).

Dans [47], nous prouvons que Hypocomb est connexe, planaire avec un degré non borné (ou borné par le nombre de nœuds dans le graphe). Ce dernier point est illustré par la figure 4.2(b) où le nœud a a un degré égal à la cardinalité du graphe moins 1. Les propriétés de Hypocomb sont donc les mêmes que celles offertes par les réductions de graphe de la littérature.

4.2.2 Un graphe planaire au degré borné

A partir de Hypocomb, nous avons alors proposé 'Reduced Hypocomb' (RHC) un graphe planaire et connexe dont le degré est borné par 6. RHC se base également sur le Besh. Une arête entre les sommets a et b existe dans RHC si et seulement si dans le Besh a bloque b ET b bloque a. Ainsi, sur la figure 4.2(a),



(a) Besh et Hypocomb (b) Degré infini

FIGURE 4.2 – Illustrations de Besh et Hypocomb.

l'arête entre b et h existe dans Hypocomb car b bloque h en s mais n'existe pas dans RHC car b n'est pas bloqué par h, son rayon s'arrêtant en t (bloqué par a) et n'atteignant pas le rayon de b. Les arêtes de RHC apparaissent en vert plein sur la figure 4.2(a).

De façon évidente, RHC est un sous-graphe connexe de Hypocomb et est donc toujours planaire. En effet, RHC reste connexe car Hypocomb est connexe et si une arête ab est retirée de Hypocomb c'est que l'un des rayons est préalablement bloqué par un autre sommet u. Cela signifie que soit il existe une arête entre a et u et entre b et u (et donc RHC est connexe), soit les arêtes au et/ou bu ont été retirées. Dans ce dernier cas, cela signifie qu'il existe un sommet v qui a empêché le maintien de au (et/ou bu) dans RHC et que le chemin avu existe dans RHC ou qu'il existe un sommet qui empêche le maintien de ce chemin, etc. Ainsi, si une arête entre deux sommets est retirée dans RHC, c'est qu'il existe un autre chemin pour relier ces deux sommets. RHC reste donc connexe.

Nous montrons que le degré d'un nœud dans RHC est inférieur ou égal à 6. Cela tient aux possibilités de blocages mutuels pouvant intervenir dans le voisinage d'un nœud. Pour mieux comprendre l'idée de la preuve, considérons la figure 4.3(a). Cette figure montre le cas où le nœud a a 6 voisins : b, c, d, e, f et g, soit le nombre de voisins maximal. En effet, a et b se bloquent mutuellement en u ; a et c se bloquent mutuellement en w ; a bloque g et c en x et est bloqué par g en z et par c en y ; a bloque f et d en v et est bloqué par f en z et par d en y. Voyons si ce degré peut être étendu. Pour ça, considérons le quart du voisinage de a dans la zone zac. Si un nœud t se situait à droite de la droite (xg) alors le rayon *sud* de t interagirait avec le rayon *ouest* de c, ce qui ferait que seulement l'un de ces deux nœuds pourrait alors établir un blocage mutuel avec a et donc être son voisin. Si ce nœud t se situait au-dessus de la droite (zg), son rayon *ouest* ne pourrait pas bloquer le rayon *nord* de a puisque ce dernier est bloqué en z et donc t ne pourrait être voisin de a. Enfin, si un nœud t se situait à l'intérieur ou sur la frontière du carré $axgz$, alors son rayon *sud* bloqueraient

le rayon *est* de a bloquant ainsi sa relation de voisinage avec g et c et son rayon *ouest* bloquerait le rayon *nord* de a qui ne sera donc plus bloqué par g.

Ainsi, RHC est un graphe connexe, planaire et de degré borné par 6.

4.2.3 Un graphe planaire au degré borné et local

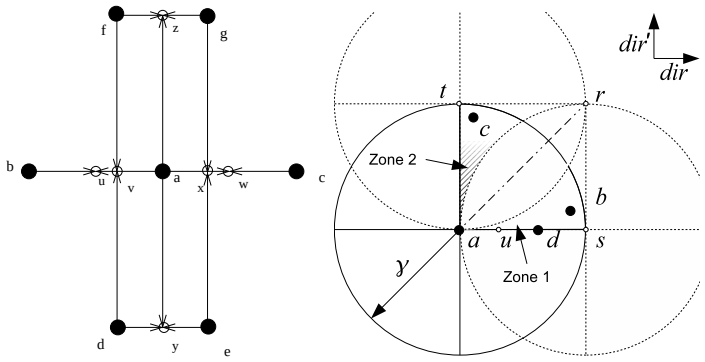
Bien qu'utiles, Hypocomb et RHC restent des graphes qui pour être construits nécessitent une connaissance globale du réseau. C'est pourquoi nous les avons déclinés en une variante qui utilise des informations purement locales : le graphe "Local Hypocomb" (LHC). Pour le construire, chaque nœud u calcule le RHC sur son voisinage uniquement (on suppose un graphe Unit Disk Graph UDG). u considère dans un premier temps toutes les arêtes se trouvant dans son voisinage (arêtes vers ses voisins et entre ses voisins) et calcule le RHC sur cette vision du graphe. Le graphe résultant est appelé LHC et nous prouvons qu'il reste connexe et planaire.

LHC est connexe du fait que le UDG est connexe et qu'une arête entre deux sommets est retirée dans ce graphe si et seulement si il existe un autre chemin entre ces sommets (comme pour RHC). RHC étant planaire, LHC reste planaire. De plus, nous avons montré que le degré des nœuds dans LHC est borné par 8. L'idée de la preuve est la suivante. Si on s'intéresse au nœud a sur la figure 4.3(b), comme LHC ne concerne que les voisins de a dans le graphe UDG, seuls les nœuds se trouvant à l'intérieur du cercle centré en a peuvent influencer le degré de a. Considérons l'arc de cercle *tas*. Dans le pire cas, il ne peut y avoir qu'un nœud dans chaque surface hachurée pouvant influencer le degré de a. En effet, si un second nœud se trouvait dans la zone 1 avec b, il bloquerait un des rayons de b avant que ce dernier n'atteigne a et donc a n'aurait pas de lien avec b. De même s'il existait un nœud x dans la zone 2 (intersection des cercles), alors ce nœud bloquerait le rayon *ouest* de b et le rayon *sud* de c, et donc a n'aurait que x comme voisin dans le quart de cercle *tas*. Ainsi, dans chaque quart de cercle, a peut avoir au plus 2 voisins, ce qui au total donne un degré maximum de 8 voisins.

Ainsi, LHC est un graphe local, planaire, connexe et de degré borné par 8.

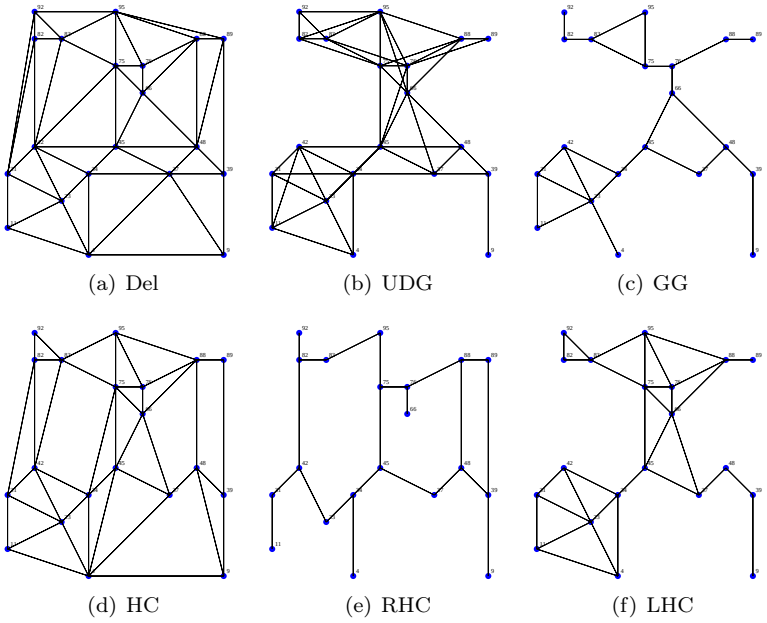
LCH est le premier graphe planaire strictement local, de degré borné qui ne se base pas sur l'identité des nœuds. Il peut alors servir de graphe planaire alternatif dans des algorithmes de routage avec garantie de livraison dans les réseaux de capteurs sans fil (voir Chapitre 3).

Afin d'illustrer les différents types de graphes, la figure 4.4 compare les différentes constructions à partir d'un même ensemble de nœuds. 'Del' représente le diagramme de Delaunay, qui est une planarisation avec connaissance globales à même titre que Hypocomb et RHC. UDG représente le graphe du disque unitaire sur lequel le graphe LCH est calculé.



(a) RHC (b) LHC

FIGURE 4.3 – Bornes sur les degrés dans RHC et LHC.



(a) Del (b) UDG (c) GG

(d) HC (e) RHC (f) LHC

FIGURE 4.4 – Famille de graphes sur une même distribution de nœuds.

4.3 Localisation assistée par actionneurs mobiles

Un des problèmes majeurs dans les réseaux de capteurs est la localisation des nœuds. En effet, comme on a pu le voir en Chapitre 3, de nombreux protocoles de routage ou de découverte de voisinage reposent sur l'hypothèse que les nœuds connaissent leur position. Cependant, cela n'est pas si trivial. Avec la baisse des coûts de production des puces GPS, on peut imaginer que tous les capteurs peuvent en être équipés. Malheureusement, le GPS ne fonctionne pas partout (comme dans des zones souterraines, tunnels, parkings, etc) et il faut donc envisager des solutions alternatives. C'est ce que nous avons voulu proposer avec DREAMS (DeteRministic bEAcon Mobility Scheduling) [45] en prenant avantage de la mobilité contrôlée offerte par les actionneurs.

L'idée est qu'un actionneur connaissant sa position visite les capteurs un par un pour leur permettre de se localiser de la façon suivante. Un capteur s à portée radio de l'actionneur a évalue la distance $|sa|$ en se basant sur la puissance du signal reçu. A partir de cette distance et de la position réelle de l'actionneur, le capteur détermine sa position. DREAMS décrit la façon dont l'actionneur a visite les capteurs pour qu'ils puissent déterminer leur position. L'avantage de DREAMS est qu'il n'utilise que de petits messages HELLO sur de faibles portées, ce qui réduit les coûts en termes de localisation mais induit un délai dans la localisation puisqu'il nécessite un contact radio avec l'actionneur. Le but principal de DREAMS est donc de réduire le parcours de l'actionneur afin de réduire le délai.

Au départ, l'actionneur émet un message balise et se déplace au hasard dans le réseau jusqu'à percevoir un message Hello de au moins un capteur. L'actionneur sélectionne un capteur comme étant la cible à localiser. Ce premier capteur est appelé Racine. Une fois que le nœud en cours de visite est localisé, ce dernier recommande un de ses voisins non visités à l'actionneur qui changera alors de cible. Ainsi, l'actionneur visite les nœuds du graphe en effectuant un parcours en profondeur du réseau en suivant un arbre. L'actionneur s'arrête lorsqu'il est retourné à la racine et que cette dernière n'a plus aucun voisin non visité.

Lorsqu'un nœud capteur s est visité, l'actionneur effectue une suite de mouvements afin de que s puisse se localiser, chaque mouvement le rapprochant de s. s utilise la puissance du signal reçu sur le message de l'actionneur qui contient la position de ce dernier. Pour illustrer les déplacements de l'actionneur, prenons la figure 4.3. S représente le capteur visité et p représente la première position à laquelle de l'actionneur détecte le nœud S. L'actionneur estime sa distance à S. Notons d cette distance. L'actionneur va alors se déplacer sur une distance de $\frac{d}{2}$ dans une direction perpendiculaire à celle qu'il avait en arrivant en p. Disons qu'il se déplace en p_1 sur la figure 4.3. Arrivé en p_1, l'actionneur estime de nouveau sa distance à S. Si cette dernière est supérieure à d ou si S est devenu hors de portée, l'actionneur fait demi-tour et rejoint la position p_2 qui est à distance $\frac{d}{2}$ de p. En p_2, S estime de nouveau sa distance à S, d'. On a nécessairement $d' < d$. L'actionneur effectue alors une rotation de $\frac{\pi}{2}$ et avance sur une distance

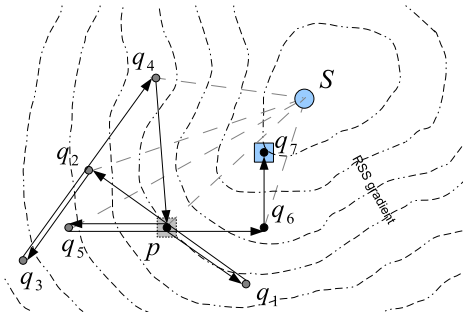


FIGURE 4.5 – DREAMS - Mouvement de l'actionneur autour des capteurs.

de $\frac{d'}{2}$ et ainsi de suite. A chaque étape, l'actionneur estime sa distance à la cible : d. Soit il s'est rapproché depuis sa dernière position et effectue une rotation de $\frac{\pi}{2}$ et avance sur une distance $\frac{d}{2}$, soit il s'est éloigné et fait marche arrière. Tout au long de son parcours, l'actionneur envoie sa position, ce qui permet à la cible S d'estimer sa distance à l'actionneur, et en appliquant un filtre sur la liste des estimations, S détermine sa position. Lorsque cette dernière devient stable, S est considéré localisé et recommande un de ses voisins à l'actionneur qui change alors de cible.

Dans le pire cas, ces suites de mouvements s'effectuent autour de chaque nœud du réseau. Cependant, afin de limiter le temps de parcours de l'actionneur, nous exploitons la communication omni-directionnelle des liens radios. Ainsi, lorsque l'actionneur envoie ses messages pour que sa cible S se localise, les nœuds se trouvant dans son voisinage reçoivent également le message et peuvent commencer à se localiser. Ainsi, surtout en environnement dense, des nœuds sont localisés sans avoir été visités. Si ces nœuds sont des feuilles dans l'arbre ou si toutes les branches issues de leurs fils sont localisées, ces nœuds ne seront pas visités. Sinon, lorsque l'actionneur les prendra comme cible, ils pourront de suite recommander un de leurs fils à l'actionneur, lui évitant de graviter autour d'eux.

Les travaux similaires de la littérature se focalisent surtout sur le parcours des actionneurs sans vérifier que les nœuds se trouvant sur le parcours soient localisés ou non. Ainsi, DREAMS donne de meilleurs résultats que ces différents travaux en ce qui concerne la localisation des nœuds.

Concernant la longueur du parcours de l'actionneur, DREAMS dépend non seulement de la zone dans laquelle les capteurs sont déployés mais également de la densité de capteurs qui y est déployée. Nous avons testé DREAMS sur plusieurs modèles d'arbres. Un nœud fraîchement localisé recommande à l'actionneur soit son voisin le plus proche, soit un au hasard, soit son voisin le plus proche parmi ses voisins dans le LMST [42]. La solution minimisant le parcours de l'actionneur est cette dernière (le voisin recommandé à l'actionneur est le plus proche parmi les voisins LMST) qui permet des parcours de longueur deux à trois fois plus grande que les solutions de la littérature en fonction de la den-

sité du réseau mais pour une erreur de localisation 4 fois plus petite. Lorsque nous étendons DREAMS à l'utilisation de plusieurs actionneurs, la distance à parcourir par actionneur diminue. A partir de 5 actionneurs, DREAMS parcourt les mêmes distances que les solutions de la littérature.

4.4 Utilisation d'actionneurs mobiles pour la couverture

Un autre avantage que peut procurer la mobilité contrôlée des actionneurs est de pouvoir améliorer la couverture d'une zone de plusieurs façons. En effet, les actionneurs peuvent *(i)* soit déployer ou re-positionner des capteurs afin que ces derniers couvrent au mieux une zone déterminée *(ii)* soit agir eux-mêmes en tant que capteurs et se déployer afin de couvrir une zone donnée *(iii)* soit visiter les capteurs régulièrement afin de récupérer leurs données. Dans le premier cas, le nombre d'actionneurs peut être limité alors que dans le second cas, le nombre d'actionneurs va dépendre de la surface à couvrir et le réseau d'actionneurs doit rester connexe. Enfin, dans le dernier cas, le réseaux d'actionneurs doit rester connexe mais ce n'est pas une nécessité pour le réseau de capteurs. Je me suis jusqu'à maintenant intéressée à la couverture assistée par actionneurs dans les deux derniers cas, c'est à dire comment optimiser la couverture d'une zone avec obstacles par un réseau d'actionneurs et comment visiter de façon optimale un ensemble de capteurs mobiles dont on ne contrôle pas le mouvement.

4.4.1 Utilisation d'actionneurs mobiles pour la couverture de zone avec obstacles

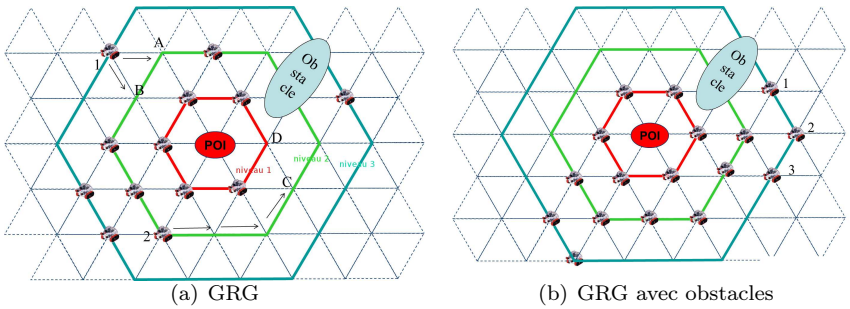
Nous avons considéré le problème suivant. Soit un ensemble d'actionneurs devant s'auto-déployer autour d'un point d'intérêt à surveiller. Les actionneurs sont autonomes et connaissent leur position géographique. Les actionneurs communiquent entre eux par lien radio. On suppose que deux actionneurs peuvent communiquer directement si la distance qui les sépare est inférieure à R. La position géographique du point d'intérêt est connue par les actionneurs.

Nous sommes partis d'un travail existant [43]. En effet [43] propose une solution pour que ces actionneurs se déploient de façon efficace et uniforme autour d'un point d'intérêt. Nous l'avons étendu afin que ce même déploiement puisse également se faire en présence d'obstacles.

Afin de mieux cibler notre contribution, je décris d'abord le travail [43] dont nous sommes partis. Comme chaque actionneur connait la position du point d'intérêt, il est capable de calculer un maillage hexagonal de rayon $3\sqrt{(R)}$ (R étant la portée de communication des actionneurs) centré sur le point d'intérêt comme illustré par la figure 4.6(a). Pour déployer les actionneurs autour de ce point d'intérêt de façon homogène tout en gardant la connexité des actionneurs, il 'suffit' que les actionneurs se positionnent sur les points du maillage autour du point d'intérêt (POI). 2 actionneurs étant sur deux points voisins du maillage

peuvent donc communiquer. On définit comme étant le niveau d'un actionneur la distance à laquelle il se trouve du POI sur l'hexagone. Par exemple, sur la figure 4.6(a), la ligne rouge représente le niveau 1, la verte le niveau 2, la bleue le niveau 3, etc.

Dans [43], afin de déployer, les actionneurs, une méthode 'Glouton-Rotation-Glouton' est appliquée. L'idée est qu'initialement, chaque actionneur se positionne sur le point de maillage le plus proche de sa position initiale et ne se déplace par la suite que sur ce maillage. Tant que cela est possible, l'actionneur se rapproche du POI de manière gloutonne. Il passe à un niveau inférieur. Par exemple, sur la figure 4.6(a), l'actionneur 1 peut se déplacer gloutonnement en position A ou B si ces emplacements sont libres. Si les emplacements sont occupés par d'autres actionneurs, l'actionneur va tourner autour du POI sans changer de niveau jusqu'à trouver un emplacement libre pour un déplacement glouton. Par exemple, sur la figure 4.6(a), l'actionneur 2 va tourner autour du POI jusqu'à l'emplacement C où il va pouvoir retourner en mode glouton et se déplacer en position D. [43] décrit alors tout un ensemble de règles de priorité pour empêcher les collisions sur un emplacement et cesser une rotation.

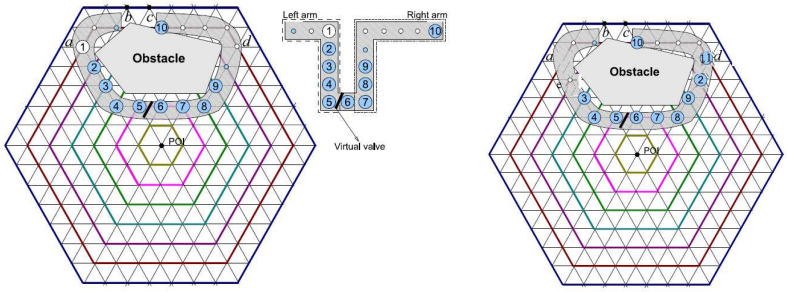


(a) GRG (b) GRG avec obstacles

FIGURE 4.6 – Illustration de couverture d'un point d'intérêt.

Bien que [43] s'avère être très performant en environnement libre, il échoue lorsqu'il existe un ou plusieurs obstacles sur la zone à couvrir. Les actionneurs sont bloqués comme le montre la figure 4.6(b). Les actionneurs 1, 2 et 3 sont bloqués au niveau 3 alors que le niveau 2 n'est pas complet. Ma contribution [44] a été d'étendre ce travail afin de permettre une couverture homogène autour du POI tout en maintenant la connexité du réseau d'actionneurs en présence d'obstacles. Pour ce travail, nous nous sommes inspirés du comportement d'un liquide qui s'écoule dans un tube en U.

Un tube en U, comme illustré sur la figure 4.7(a) se compose de 3 sections : un bras gauche, un bras droit et une base. Lorsque le tube contient un liquide, le liquide est au même niveau dans les deux bras du tube. Nous considérons que le point d'intérêt est le centre d'attraction du liquide et que le tube encercle l'obstacle comme représenté en figure 4.7(a). L'idée est ensuite la suivante. Nous supposons qu'un actionneur est capable de détecter s'il côtoie un obstacle et donc s'il est ou non dans le tube. Si un actionneur se trouve bloqué par la faute



(a) Le tube virtuel (b) GRG et le tube

FIGURE 4.7 – Le liquide s'écoule de façon à équilibrer la pression dans les différentes branches du tube.

d'un obstacle, il glisse dans le tube, soit par un mouvement glouton comme dans GRG (comme pourrait le faire un actionneur se trouvant en position d sur la figure 4.7(a)), soit par un mouvement de rotation comme dans GRG (comme pourrait faire l'actionneur 8 si l'actionneur 7 sortait du tube) ou soit en remontant dans le tube pour équilibrer la pression.

Pour ce faire, nous autorisons les diffusions de message dans le tube. Un actionneur se retrouvant bloqué dans le tube envoie un message qui se propage uniquement dans le tube en indiquant le niveau auquel il se trouve. Par exemple, sur la figure 4.7(b), l'actionneur 11 arrive en position d dans le tube et va être bloqué. Il va donc diffuser un message dans le tube en indiquant son niveau (niveau 6). Ce message va atteindre l'actionneur 3 qui va alors se déplacer en position e. Et ainsi de suite, les actionneurs vont glisser dans le tube jusqu'à équilibrer la pression des deux côtés. Ayant atteint le niveau 6, l'actionneur peut alors continuer l'algorithme de base et se mettre en mode rotation.

Tout un nombre de règles ont de plus été conçues dans [44] afin d'éviter les boucles (un actionneur ayant traversé le tube doit s'arrêter lorsqu'il rencontre ce même tube de nouveau afin de ne pas tourner indéfiniment) et empêcher les collisions entre actionneurs dans le tube (les actionneurs se trouvant dans le tube sont prioritaires).

Les résultats montrent que notre approche permet un déploiement efficace qui converge rapidement en ne générant que peu de messages superflus dans le tube (dépendant de la taille de l'obstacle, entre 2 et 3 messages par actionneurs en moyenne).

4.4.2 Utilisation d'actionneurs mobiles pour la couverture de nœuds mobiles

Lorsque les capteurs sont mobiles, comme par exemple portés par des animaux et qu'on ne contrôle pas leur déplacement, il est difficile de maintenir la connexité et s'assurer la remontée d'informations vers un nœud puits. Dans de

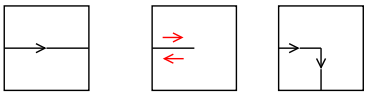


FIGURE 4.8 – Trois façons de traverser une cellule.

pareils cas, l'utilisation d'actionneurs mobiles peut s'avérer utile. En effet, si les actionneurs visitent régulièrement les capteurs pour prélever les informations qu'ils portent, ces derniers n'ont plus besoin de les acheminer en utilisant le réseau de capteurs. Les actionneurs doivent donc assurer une couverture des nœuds mobiles de façon régulière afin que les mémoires des capteurs ne soient pas saturées. Si on suppose que nous n'avons aucune information *a priori* sur le modèle de mobilité des capteurs qui sont incontrôlables, ce problème revient à ce qu'un ensemble d'actionneurs couvrent une zone donnée. C'est l'idée de COVER [66] qui propose un algorithme assurant que sous hypothèse que le nombre d'actionneurs est suffisant, chaque capteur sera visité au moins une fois toutes les $\frac{1}{f_{min}}$ secondes, f_{min} étant la fréquence minimale de visite des capteurs par un actionneur pour assurer la non saturation de leur mémoire.

Le but de COVER est d'équilibrer la charge entre les actionneurs, c'est à dire de faire en sorte que chaque actionneur surveille un nombre équivalent de capteurs.

Pour répondre à ce problème, COVER suppose que chaque actionneur connait sa position géographique et peut délimiter des cellules dans l'espace. On suppose que chaque cellule contient au maximum T_{max} capteurs (Si on considère que les capteurs sont portés par des animaux, disons des éléphants, et qu'une cellule fait $10m^2$, il n'est pas abberant de considérer que chaque cellule contient au plus 3 capteurs) et est un carré de côté l. l est choisi tel que entre deux visites d'un actionneur, un capteur ait pu traverser au plus une cellule : $l \leq \frac{v_c}{f_{min}}$ où v_c est la vitesse maximale à laquelle un capteur se déplace.

COVER attribue à chaque actionneur i un ensemble de Z_i cellules. Initialement, si la zone totale à couvrir contient C cellules et qu'on dispose de N actionneurs, chaque actionneur reçoit $\frac{C}{N}$ cellules. Cet ensemble de cellules est traversé en boucle par l'actionneur. Comme le montre la figure 4.8, l'actionneur peut traverser une cellule de trois façons différentes, chacune sur une distance l. Si un actionneur entre dans une cellule, il est en capacité de récupérer les informations concernant tous les capteurs se trouvant dans cette cellule en moins de $\frac{l}{V}$ secondes.

Pour assurer que tous les capteurs sont visités au moins une fois toutes les $\frac{1}{f_{min}}$ secondes sachant qu'entre deux visites ils peuvent se déplacer de au plus une cellule, alors chaque cellule doit être visitée au moins deux fois toutes les $\frac{1}{f_{min}}$ secondes.

Ainsi, si un actionneur se déplace à une vitesse maximale V et que chaque cellule doit être visitée au moins deux fois toutes les $\frac{1}{f_{min}}$ secondes, le parcours maximal de l'actionneur ne peut dépasser $\frac{V}{2f_{min}}$. Cela conduit à un nombre maximum de cellules qui peuvent être attribuées à un actionneur : $Z_{max} =$

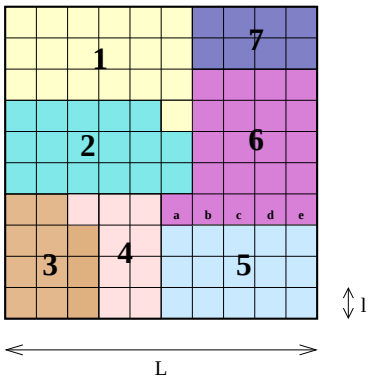


FIGURE 4.9 – Illustration d'une zone de 10×10 cellules couverte par 7 actionneurs.

$\frac{V}{2lf_{min}}$.

Ces bases étant posées, COVER va adapter dynamiquement le nombre de cellules entre les actionneurs de façon à ce qu'ils couvrent tous plus ou moins le même nombre de capteurs en ne se basant que sur des informations locales. Chaque actionneur sait le nombre de capteurs se trouvant dans la zone qu'il surveille et échange cette information avec ses voisins ainsi que le nombre de cellules qu'il visite.

Si le nombre total de capteurs à visiter est M alors, dans l'idéal, chacun des N actionneurs devrait surveiller $\frac{M}{N}$ capteurs. Ainsi, localement, chaque actionneur va tenter de maintenir le nombre de capteurs proche de cet optimum. Pour cela, lorsqu'un actionneur i détecte qu'il surveille un nombre de capteurs supérieur à $\frac{M}{N} + \theta$, i va céder des cellules à un de ses voisins. Pour cela, i choisit un de ses voisins dont le nombre de cellules visitées est inférieur à Z_{max} (afin de garantir la fréquence de visite) et dont le nombre de capteurs surveillés est inférieur à $\frac{M}{N} + \theta$. θ est une constante telle que $\theta > 2T_{max}$ utilisée pour éviter que deux actionneurs voisins s'échangent éternellement une cellule.

Une fois le voisin à qui léguer une cellule est choisi, il s'agit de choisir la cellule à donner. Cette cellule doit se trouver à la frontière entre les parcours des deux actionneurs et telle que les zones des actionneurs restent connexes et le plus compacte possible. Pour cela, un actionneur va chercher à réduire le périmètre de sa zone. Pour illustrer cela, prenons la figure 4.9. Si l'actionneur m_6 doit céder une cellule à l'actionneur $m - 5$, il a le choix entre les cellules a, b, c, d et e. Si m_6 donnait la cellule b, sa zone serait déconnectée. Si m_6 donnait la cellule c ou d, le périmètre de sa zone serait élargi de $2l$ alors que s'il choisit la cellule a ou e, son périmètre est réduit de $2l$. Finalement, afin d'optimiser sa trajectoire, m_6 choisit la cellule a.

COVER propose également un mécanisme supplémentaire pour le cas où un actionneur devant céder une cellule se trouve entouré d'actionneurs ne pouvant plus recevoir de nouvelles cellules (soit parce qu'ils surveillent suffisamment de

capteurs ou qu'ils visitent un nombre maximal de zones), il puisse déclencher que au moins un de ses voisins cède un cellule à un autre actionneur ou déclenche lui-même ce mécanisme.

Les résultats montrent que COVER réussit à remplir ses objectifs d'une manière simple, locale et distribuée. COVER équilibre la charge entre les actionneurs.

4.5 Perspectives

Ce chapitre s'est principalement intéressé à la mobilité, subie ou contrôlée, dans un réseau de capteurs et d'actionneurs dans le contrôle de topologie. En effet, les algorithmes de la littérature ne tiennent pas toujours compte de ces mobilités. Mon but est de prendre en compte la mobilité subie dans les algorithmes en proposant des contrôles de topologie distribués et locaux qui prennent en paramètre des éléments de la couche physique comme la dynamique du voisinage d'un nœud afin de les rendre robustes vis à vis de cette mobilité. C'est par exemple ce que je commence à regarder dans le cadre de la thèse de Jovan Radak. Ainsi, des algorithmes plus haut niveau, comme par exemple des algorithmes de routage, reposant sur un tel contrôle de topologie, seront également plus robustes face à la mobilité subie des nœuds. Enfin, la mobilité contrôlée des actionneurs s'avère être un atout puissant que j'espère explorer plus en amont. Ce chapitre décrit les premières études que j'ai menées qui considèrent des actionneurs. J'ai supposé que soit le réseau était entièrement composé d'actionneurs [44] soit les actionneurs venaient récupérer les informations de capteurs qui n'ont pas besoin de communiquer entre eux [66, 45]. Mon but est de poursuivre ces études dans des cas où les actionneurs et les capteurs forment un seul et même réseau et où ils peuvent s'entraider.

4.6 Publications

- F. Ingelrest, N. Mitton, and D. Simplot-Ryl. A turnover based adaptive hello protocol for mobile ad hoc and sensor networks. In *Proc. of the 15th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 07)*, Bogazici University, Istanbul, Turkey, October 2007.
- X. Li, N. Mitton, I. Ryl, and D. Simplot. Localized sensor self-deployment with coverage guarantee in complex environment. In *Proc. of the 8th Int. Conf. on AD-HOC Networks & Wireless (Ad Hoc Now 09)*, vol. 5793 of Lecture Notes in Computer Science, p. 138-151, 2009.
- X. Li, N. Mitton, and D. Simplot-Ryl. Mobility prediction based neighborhood discovery in mobile ad hoc networks. In *Proc. of the IFIP/TC6 NETWORKING 2011*, Valencia, Spain, 2011. to appear.
- X. Li, N. Mitton, I. Simplot-Ryl, and D. Simplot-Ryl. A novel family of geometric planar graphs for wireless ad hoc networks. In *Proc. 30th*

IEEE Int. Conf. Computer Communications (INFOCOM 2011), Shanghai, China, 2011. to appear.

– T. Razafindralambo, N. Mitton, A. Carneiro Viana, M. Dias de Amorim, and K. Obraczka. Adaptive infrastructure deployment for data gathering in intermittently connected networks. In *Proc. of the 8th IEEE Pervasive Computing and Communication (PerCom 2010)*, Mannheim, Germany, 2010.

Chapitre 5

Conclusion et perspectives

5.1 Conclusion

Mes travaux de recherche s'articulent autour de l'auto-organisation et le passage à l'échelle dans la RFID active et passive, plus précisément au travers des intergiciels RFID et des réseaux de capteurs et d'actionneurs sans fil. J'ai jusqu'à maintenant étudié et proposé des solutions à différents niveaux pour ces différents types de réseaux : auto-organisation, localisation, routage, contrôle de topologie.

Ainsi, j'ai appliqué des techniques pair-à-pair à la localisation dans les réseaux de capteurs et à différentes briques d'un intergiciel RFID afin de permettre dans tous les cas le passage à l'échelle. L'application de tables de hachage distribuées permet d'alléger les composants du réseau considéré (capteurs, brique ALE, ONS, bases de données) aussi bien en termes de capacité de stockage, charge et sollicitation de requêtes en répartissant les données sur l'ensemble des composants. Les requêtes sont mieux réparties dans le réseau et moins de nœuds sont sollicités lors de son envoi. Cela permet également, dans le cadre de l'ONS, de répondre à un enjeu géopolitique sur les responsabilités d'une racine ONS. Ces mécanismes ont été conçus de telle sorte que leur utilisation est transparente pour l'utilisateur et, dans le cas des briques de l'intergiciel RFID, ils restent compatibles avec les standards actuels.

J'ai également étudié plusieurs options de routage géographique dans les réseaux de capteurs aussi bien dans le cas où les coordonnées géographiques des nœuds sont disponibles que quand elles ne le sont pas. Dans le premier cas, j'ai étendu les solutions de la littérature pour proposer une solution de routage économe en énergie de bout en bout, c'est à dire un protocole de type Glouton-Recouvrement-Glouton où chaque étape du routage est économe en énergie. Dans le deuxième cas, les performances du routage s'appuient sur le type du système de coordonnées déployé ainsi que sur le type de distance calculée sur ces coordonnées. J'ai tout d'abord proposé un protocole économe en énergie inspiré des solutions où les positions sont disponibles et étudié l'impact de ce

système et de cette distance. Puis, j'ai étendu ces travaux en proposant un autre système de coordonnées basé sur la construction d'un arbre, ce qui a permis la conception d'algorithmes garantissant la délivrance du paquet. Enfin, en combinant les différentes approches et systèmes de coordonnées, j'ai contribué à l'élaboration du premier protocole de routage géographique sur coordonnées virtuelles qui soit à la fois économe en énergie et garantissant la délivrance du message.

Enfin, je me suis intéressée au contrôle de topologie dans un réseau de capteurs en cas de mobilité. Cette mobilité peut être soit subie soit contrôlée. Le contrôle de topologie est une auto-organisation du réseau qui permet un passage à l'échelle facilité par le fait qu'il permet à chaque entité de se concentrer seulement sur les liens importants de son voisinage. Dans ce cadre, je me suis d'abord intéressée à la découverte de voisinage en environnement mobile et proposé deux solutions efficaces, une dans le cadre où les nœuds ne connaissent pas leur position, une lorsqu'ils la connaissent. La découverte de voisinage est une primitive très importante dans un réseau de capteurs vu que l'essentiel des protocoles de routage et d'auto-organisation reposent sur elle. Notre travail est d'autant plus important que peu de solutions prennent en compte le caractère dynamique des nœuds. De la même façon, plusieurs algorithmes reposent sur l'hypothèse que le graphe a été au préalable planarisé. J'ai donc proposé une famille de graphes planaires dont l'un a un degré borné et se calcule de façon locale et distribuée. Enfin, j'ai tiré parti du fait que dans certains cas, la mobilité des nœuds n'était pas subie mais contrôlée pour proposer un algorithme de localisation ainsi que des algorithmes de couverture de surface efficace ou de nœuds.

5.2 Programme de recherche

Ainsi, mes recherche ont jusqu'à maintenant porté sur deux types de réseaux : les réseaux de capteurs et d'actionneurs et les systèmes RFID. Cependant, j'aimerais prolonger ces travaux sur deux axes : horizontalement et verticalement. Horizontalement car à mon sens, bien que réunis sous les mêmes concepts (Internet des objets) et familles (RFID active et passive), dans les faits, les réseaux de capteurs/actionneurs et les systèmes RFID sont à ce jour et en pratique encore trop distincts. L'intégralité des technologies réunies sous la bannière de l'Internet des Objets devrait être homogénéisée afin de coopérer et de ne former qu'un seul réseau collaboratif. Verticalement car les solutions que j'ai proposées à ce jour, bien que performantes sous le regard de modèles théoriques et de simulations, souffrent encore du manque d'expérimentations qui permettraient de valider et permettre la mise en pratique des solutions étudiées aussi bien dans les réseaux de capteurs que dans les systèmes RFID.

5.2.1 Vers un Internet des Objets unifié

Pour unifier l'Internet des objets, il faut dans un premier temps que les systèmes hétérogènes qui le composent (téléphones portables, ordinateurs, lecteurs

RFID, capteur sans fil, etc) soient interconnectés. Avec l'essor des nouvelles technologies sans contact (ZigBee, WiFi, Bluetooth, NFC, etc), cela devient une réalité. Il faut cependant unifier les services sous-jacents afin de permettre de nouvelles applications comme l'interrogation à distance d'un lecteur RFID ou d'une commande domotique via un téléphone portable. Tous les objets et évènements doivent s'interconnecter en respectant les standards en cours, être interactifs et construire un réseau intelligent et autonomique. Chaque entité est considérée comme un objet éventuellement mobile qui doit pouvoir être dynamiquement identifié et éventuellement contrôlé. Ceci doit se faire de façon distribuée et robuste pour pouvoir supporter l'ajout et le retrait dynamique d'objets et d'évènements dans le système ainsi que leur mobilité (subie ou contrôlée) dans le réseau. Dans le cadre du projet Aspire [3], nous avons étudié l'extension des standards actuels de nommage d'évènements et d'objets RFID à différents systèmes de l'Internet des objets (téléphones, ordinateurs, normes ISO, etc) pour une redirection des évènements commune à tous. Je compte étendre ces recherches pour qu'également la configuration des objets soit facilitée et ouverte à tout système hétérogène.

Les différents systèmes de l'Internet des objets peuvent également s'interconnecter à un niveau différent. Prenons par exemple un lecteur RFID mobile. Un tel lecteur peut communiquer ses lectures d'étiquettes RFID sur ondes radio. Il peut alors être vu comme un nœud mobile dans un réseau de capteurs, permettant de combiner les avantages de chaque technologie et permettant par exemple de géolocaliser finement une étiquette RFID grâce à la technologie des réseaux de capteurs. Inversement, les évènements RFID seront des données d'entrée au réseau de capteurs à même titre qu'une indication de température ou de luminosité. Je souhaite étudier comment intégrer la RFID passive dans un réseau de capteurs (RFID active), les évènements RFID dans les technologies des réseaux de capteurs et inversement. Les lecteurs RFID répondant aux standards EPC [21] doivent répondre à certaines règles et être configurables au travers d'une ALE. Alors qu'un lecteur RFID fixe connecté à un réseau en filaire est facilement configurable, notamment avec des outils tels que AspireRFID [3], configurer à distance un lecteur mobile peut se révéler plus fastidieux. Je m'intéresserai à des moyens de configuration facile et transparente pour un utilisateur. En effet, la configuration des lecteurs mobiles au travers du réseau devra prendre en compte les besoins des applications et utilisateurs et des caractéristiques du réseau et des lecteurs afin d'en déduire le meilleur compromis. Plus la partie de l'intergiciel déportée sur les lecteurs est importante, plus les données seront filtrées et agrégées et le réseau sera moins sollicité, ce qui lui assurera de meilleures performances. En contrepartie, cela nécessite des lecteurs plus puissants. Il s'agira d'établir les mécanismes à mettre en place qui seront capables d'établir le meilleur compromis et de déployer de façon efficace les lecteurs RFID mobiles.

De la même façon, un téléphone portable peut également être vu comme un nœud d'un réseau de capteurs et d'actionneurs. Il peut en plus jouer le rôle de passerelle vers le réseau 3G. Le téléphone portable devra donc également être facilement configurable.

5.2.2 De la théorie à l'expérimentation

Je souhaite également étendre mes recherches en profondeur. En effet, la plupart des solutions présentées dans ce mémoire ont été évaluées au travers d'une étude théorique et par simulation. Peu d'entre elles ont été expérimentées. Ce constat est le même pour la plupart des solutions de la littérature. Cependant, si nous prenons l'exemple du routage géographique (Chapitre 3), nous avons proposé des solutions très performantes, meilleures que les autres propositions de la littérature en termes de complexité et de consommation énergétique. Nous avons prouvé que sous certaines hypothèses, ces protocoles garantissaient la délivrance du message à son destinataire. Pour cela, le graphe doit être planaire. Les hypothèses et données entrantes des protocoles de routage, auto-organisation, diffusion, ordonnancement d'activité sont récurrentes dans toutes les propositions de la littérature, à savoir :

- le modèle de communication sous-jacent est le modèle du disque unitaire, *c.à.d.* que tout nœud situé dans un disque de rayon R centré autour de l'émetteur recevra le message envoyé, tout nœud en dehors de ce disque ne recevra pas le message ; R étant la portée de communication des nœuds, proportionnelle à la puissance d'émission,
- un nœud est capable d'estimer la distance exacte qui le sépare d'un de ses voisins en se basant sur l'intensité du signal reçu par ce même nœud, la puissance d'émission étant connue,
- la table de voisinage d'un nœud est exacte,
- ...

Ce ne sont que quelques exemples mais ces hypothèses sur lesquelles reposent les différents protocoles de la littérature ont peu de chances d'être vérifiées dans un environnement réel et mobile pour tout type de matériel, rendant inutilisables certaines propositions de la littérature. C'est pourquoi je compte étudier les limites de ces hypothèses dans des environnements réels et réalistes. A partir d'une meilleure compréhension de l'impact d'un environnement réaliste et de l'analyse des informations qui sont vraiment disponibles pour le protocole, j'étudierai des moyens de contourner ces hypothèses ou de tirer parti de leurs caractéristiques. Les nouvelles solutions cette fois, tiendront compte des aléas liés à un environnement réaliste et à la mobilité des nœuds. Les protocoles doivent prendre en données entrantes des données de l'environnement quand elles sont disponibles (comme la puissance d'un signal et/ou la vélocité d'un nœud) et non plus utiliser des données extraites de ces informations. Il faut donc aborder ainsi une approche *cross-layer* entre les couches physiques, l'environnement et les couches de niveau 3 de routage et d'auto-organisation. Pour cela, chaque algorithme se doit d'être non seulement étudié analytiquement et par simulation mais également expérimenté en environnement réel afin de valider les hypothèses sur lesquelles il repose. La plateforme SensLAB [14, 15], plateforme de capteurs à large échelle, donnera un moyen de faire ces expérimentations et de comprendre les comportements des algorithmes. Les retours de ces expérimentations seront réinjectés dans les analyses théoriques et les simulations de façon à produire des protocoles adaptés à un environnement réaliste. Une fois les phénomènes phy-

siques mieux assimilés, certains protocoles existants devront donc être repensés en conséquence afin de pouvoir permettre un comportement optimal au réseau de capteurs.

Suivant l'application et le type de matériel, des protocoles parfaitement adaptés dans certains cas ne fonctionneront plus dans d'autres. Par exemple, certains algorithmes ne fonctionneront que dans des environnements statiques et deviendront inutilisables dans un réseau dynamique. Ainsi, de la même façon qu'il doit y avoir des interactions entre les protocoles et l'environnement, il faut établir des liens entre les applications et les protocoles. De la même façon, si le réseau n'est pas constitué uniquement de capteurs mais comprend également des actionneurs ayant la capacité d'agir sur leur environnement (comme par exemple en se déplaçant pour assurer la connexité du réseau), ces échanges entre couches deviennent bilatéraux. Une fois encore, dans ces cas-là, chaque protocole doit pouvoir s'adapter aux applications et à son environnement, et sera validé par des expérimentations qui viendront enrichir la théorie et la simulation. L'un de mes axes de recherche sera également d'étudier comment la mobilité des actionneurs peut améliorer les performances du réseau. Comment les actionneurs, membres du réseau de capteurs et d'actionneurs, vont pouvoir faciliter le routage, limiter la consommation énergétique, faciliter la localisation des capteurs, étendre le réseau soit de façon temporaire pour une application précise en servant eux-mêmes de relais, soit de façon permanente en déployant de nouveaux capteurs etc.

De plus, ces expérimentations permettront également le *design* de protocoles plus élaborés en mettant en exergue les points forts et faibles de chaque action. Par exemple, les protocoles de routage et d'acheminement de données pourront intégrer de l'agrégation de données et l'expérimentation permettra de mieux évaluer le gain en énergie et bande passante que cela produit mais également de mesurer finement les délais et latences, métriques difficiles à modéliser avec exactitude.

5.2.3 Résumé

Mon projet a pour but de concevoir et maîtriser une infrastructure pour un Internet des objets unifié. Il se veut répondre à des enjeux naissants de la société et permet d'intégrer des résultats de recherche de plusieurs milieux techniques, sociaux et économiques. Étudier les réseaux de capteurs, actionneurs et RFID avec des perspectives à plus long terme permet d'envisager de nouvelles applications qui elles-mêmes soulèvent de nouveaux enjeux.

Bibliographie

[1] A. Ahmad Kassem and N. Mitton. Adapting dynamically neighbourhood table entry lifetime in wireless sensor networks. In *Proc. of the 10th International Conference on Wireless Communications and Signal Processing (WCSP 2010)*, Suzhou, China, 2010. Best paper award.

[2] M. Dias De Amorim, S. Fdida, N. Mitton, L. Schmidt, and D. Simplot-Ryl. Distributed planetary object name service : Issues and design principles. Research Report 7042, INRIA, 09 2009.

[3] FP7 Aspire. Fp7 ICT IP Project advanced sensors and lightweight programmable middleware for innovative RFID enterprise applications (ASPIRE). `http://www.fp7-aspire.com`, 2008.

[4] H. Balakrishnan, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Looking up data in P2P systems. *Communications of the ACM*, 46(2) :43–48, February 2003.

[5] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (dream). In *Proc. of the* 4^{th} *ACM Annual Int. Conference on Mobile Computing and Networking (MOBICOM)*, pages 76–84, Dallas, Texas, 1998.

[6] F. Benbadis, J-J. Puig, M.Dias de Amorim, C. Chaudet, T. Friedman, and D. Simplot-Ryl. Jumps : Enhanced hop-count positioning in sensor networks using multiple coordinates. *International Journal on Ad Hoc & Sensor Wireless Networks*, 2008.

[7] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proc. of the* 3^{rd} *Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Comm. (DIAL-M)*, pages 48–55, Seattle, WA, USA, August 1999.

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

[9] A. Caruso, S. Chessa, S. De, and A. Urpi. Gps free coordinate assignment and routing in wireless sensor networks. In *Proc. of the* 24^{th} *Conference of the IEEE Communications Society (INFOCOM)*, volume 1, pages 150–160, Miami, USA, March 2005.

[10] H. Chaouchi. *Internet of Things. Connecting Objects.* Wiley and Sons, January 2010.

[11] E. Chávez, M. Fraser, and H. Tejeda. Proximal labeling for oblivious routing in wireless ad hoc networks. In *Proc. of the 8th International Conference on AD-HOC Networks & Wireless (Ad Hoc Now'09)*, pages 360–365, 2009.

[12] E. Chávez, N. Mitton, and H. Tejeda. Routing in wireless networks with position trees. In *Proc. of the 6th International Conference on AD-HOC Networks & Wireless (Ad Hoc Now'07)*, Morelia, Mexico, September 2007.

[13] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. OLSR - Optimized Link State Routing Protocol, October 2003. RFC 3626.

[14] C. Burin des Roziers, G. Chelius, T. Ducrocq, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, T. Noel, and J. Vandaele. Using SensLAB as a first class scientific tool for large scale wireless sensor network experiments. In *Proc. of the IFIP/TC6 NETWORKING 2011*, Valencia, Spain, 2011. to appear.

[15] C. Burin des Roziers, G. Chelius, E. Fleury, A. Fraboulet, A. Gallais, N. Mitton, and T. Noel. Senslab : Very large scale open wireless sensor network testbed. In *Proc. of the 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2011)*, Shanghai, China, 2011. to appear.

[16] T. Ducrocq, J. Vandaele, N. Mitton, and D. Simplot-Ryl. Large scale geolocalization and routing experimentation with the senslab testbed. In *Demo. in Proc. of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2010)*, San Francisco, USA, 2010. Best Demo Award.

[17] E. Elhafsi, N. Mitton, B. Pavkovic, and D. Simplot-Ryl. Energy-aware georouting with guaranteed delivery in wireless sensor networks with obstacles. *International Journal of Wireless Information Networks*, 16(3) :142–153, 2009.

[18] E. H. Elhafsi, N. Mitton, and D. Simplot-Ryl. Cost over progress based energy efficient routing over virtual coordinates in wireless sensor networks. In *Proc. of the 1st IEEE International Workshop From Theory To Practice in Wireless Sensor Networks (t2pWSN'2007)*, Helsinki, Finland, June 2007.

[19] E. H. Elhafsi, N. Mitton, and D. Simplot-Ryl. Cost over progress based energy-efficient routing protocol over virtual coordinates in wireless sensor networks. In *9èmes Rencontres Francophones sur les aspects Algorithmiques des Télécommunications, ALGOTEL 2007*, Ile d'Oléron, France, Mai 2007.

[20] E. H. Elhafsi, N. Mitton, and D. Simplot-Ryl. End-to-end energy efficient geographic path discovery with guaranteed delivery in ad hoc and sensor networks. In *Proc. of the 19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'08)*, Cannes, France, September 2008.

[21] EPC. EPC global. `http://www.epcglobalinc.org`.

[22] Q. Fang, J. Gao, L.J. Guibas, V. de Silva, and L. Zhang. GLIDER : gradient landmark-based distributed routing for sensor networks. In *Proc. of the* 24^{th} *Conference of the IEEE Communications Society (INFOCOM)*, volume 1, pages 339–350 vol. 1, Miami, USA, March 2005.

[23] G.G. Finn. Routing and addressing problems in large metropolitan-scale. *Internetworks, ISI Research Report ISU/RR-87-180*, March 1987.

[24] E. Fleury and D. Simplot-Ryl. *Réseaux de capteurs : théorie et modélisation*. Hermess, January 2009.

[25] P. Fraigniaud and P. Gauron. An overview of the content-addressable network D2B. In *Proc. of the 22nd ACM Symposium on Principles of Distributed Computing (PODC'03)*. ACM, July 2003.

[26] A. Ghaddar, T. Razafindralambo, I. Simplot-Ryl, D. Simplot-Ryl, and S. Tawbi. Towards energy-efficient algorithm-based estimation in wireless sensor networks. In *Proc. of the 6th International Conference on Mobile Ad-hoc and Sensor Networks (MSN'10)*, Hangzhou, China, 2010.

[27] EPC Global. EPC information systems. `http://www.gs1.org/gsmp/kc/epcglobal/epcis/`, 2007.

[28] EPC Global. The Application Level Events (ALE) specification. `http://www.gs1.org/gsmp/kc/epcglobal/ale/`, 2008.

[29] EPC Global. Object naming service (ONS) standard, version 1.0.1. `http://www.epcglobalinc.org/standards/ons`, 2008.

[30] D.K. Goldenberg, J. Lin, and A.S. Morse. Towards mobility as a network control primitive. In *Proc. of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, pages 163–174, september 2004.

[31] GS1. Gs1, *GS1 General Specifications v10*. `www.gs1.org`.

[32] E. Hamouda, N. Mitton, B. Pavkovic, and D. Simplot-Ryl. Vers un protocole de routage géographique économe en énergie de bout en bout avec garantie de livraison. In *Proc. 12-iemes Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications (Algotel 2010)*, Belle Dune, France, 2010.

[33] E. Hamouda, N. Mitton, and D. Simplot-Ryl. Energy efficient mobile routing in actuator and sensor networks with connectivity preservation. In *Submitted to AdHocNow*, 2011.

[34] T.-C. Hou and V. Li. Transmission range control in multihop packet radio networks. *Communications, IEEE Transactions on [legacy, pre - 1988]*, 34(1) :38–44, 1986.

[35] FUI ICOM. Infrastructure pour le commerce du futur, 2008.

[36] F. Ingelrest, N. Mitton, and D. Simplot-Ryl. A turnover based adaptive hello protocol for mobile ad hoc and sensor networks. In *Proc. of the 15th IEEE International Symposium on Modeling, Analysis, and Simulation of*

Computer and Telecommunication Systems (MASCOTS'07), Bogazici University, Istanbul, Turkey, October 2007.

[37] F. Ingelrest, N. Mitton, and D. Simplot-Ryl. *Réseaux de capteurs : théorie et modélisation (Collection Architecture, Applications, Service)*, volume IX, chapter Applications à la diffusion. Lavoisier, 2009.

[38] D.B. Johnson, D. A. Maltz, and J. Broch. DSR : The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad Hoc Networking*, 5 :139–172, 2001.

[39] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. of the 11th Canadian Conference on Computational Geometry CCCG*, Vancouver, Canada, 1999.

[40] J. Kuruvila, A. Nayak, and I. Stojmenovic. Progress and location based localized power aware routing for ah hoc sensor wireless networks. *International Journal on Distributed Sensor Networks (IJDSN)*, 2 :147–159, 2006.

[41] J. Li, L. Gewali, H. Selvaraj, and V. Muthukumar. Hybrid greedy/Face routing for ad-hoc sensor networks. In *Proc. of the Digital System Design, EUROMICRO Systems (DSD)*, volume 0, pages 574–578, Los Alamitos, CA, USA, 2004.

[42] N. Li, J. Hou, and L. Sha. Design and analysis of an mst-based topology control algorithm. *IEEE Tran. on Wireless Communications*, 4(3) :1195–1206, 2005.

[43] X. Li, H. Frey, N. Santoro, and I. Stojmenovic. Localized sensor self-deployment with coverage guarantee. *ACM SIGMOBILE Mobile Computing and Communications Review.*, 12(2) :50–52, 2008.

[44] X. Li, N. Mitton, I. Ryl, and D. Simplot. Localized sensor self-deployment with coverage guarantee in complex environment. In *Proc. of the 8th Int. Conf. on AD-HOC Networks & Wireless (Ad Hoc Now'09)*, volume 5793 of *Lecture Notes in Computer Science*, pages 138–151, 2009.

[45] X. Li, N. Mitton, I. Ryl, and D. Simplot-Ryl. A novel sensor localization scheme by mobile actors. In *Proc. of the 10th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 339–340, New Orleans, USA, 2009. ACM. Poster abstract.

[46] X. Li, N. Mitton, and D. Simplot-Ryl. Mobility prediction based neighborhood discovery in mobile ad hoc networks. In *Proc. of the IFIP/TC6 NETWORKING 2011*, Valencia, Spain, 2011. to appear.

[47] X. Li, N. Mitton, I. Simplot-Ryl, and D. Simplot-Ryl. A novel family of geometric planar graphs for wireless ad hoc networks. In *Proc. 30th IEEE Int. Conf. Computer Communications (INFOCOM 2011)*, Shanghai, China, 2011. to appear.

[48] X.-Y. Li. Approximate mst for udg locally. In *Proc. of The 9th Annual International Computing and Combinatorics Conference (COCOON)*, volume 2697, pages 364–373, 2003.

[49] H. Liu, A. Nayak, and I. Stojmenovic. Localized mobility control routing in robotic sensor wireless networks. *LNCS, Mobile Ad-Hoc and Sensor Networks*, 4864 :19–31, 2007.

[50] K. Liu and N. Abu-Ghazaleh. Aligned virtual coordinates for greedy routing in wsns. In *Proc. of the 3rd IEEE Int. Conference on Mobile Adhoc and Sensor Systems (MASS 2006)*, pages 377–386, October 2006.

[51] M. Lukic, B. Pavkovic, N. Mitton, and I. Stojmenovic. Greedy geographic routing algorithms in real environment. In *Proc. of the 5th Int. Conf. on Mobile Ad-hoc and Sensor Networks (MSN 2009)*, pages 86–93, Wu Yi Mountain, China, December 2009.

[52] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy : A scalable and dynamic emulation of the butterfly. In *Proc. of the 21st ACM Symposium on Principles of Distributed Computing (PODC'02)*, 2002.

[53] P. Maymounkov and D. Mazieres. Kademlia : A peer-to-peer information system based on the XOR metric. In *Proc. of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, MIT Faculty Club, Cambridge, MA, USA, March 2002.

[54] N. Mitton. *Auto-organisation dans les réseaux sans fil multi-sauts à large échelle*. PhD thesis, INSA de Lyon, Lyon, Mars 2006.

[55] N. Mitton, A. Busson, and E. Fleury. Self-organization in large scale ad hoc networks. In *Proc. of the 3rd Annual Mediterranean Ad Hoc Networking Workshop, MED-HOC-NET 04*, Bodrum, Turkey, June 2004.

[56] N. Mitton, A. Busson, and E. Fleury. Efficient broadcasting in self-organizing sensor networks. *International Journal of Distributed Sensor Networks (IJDSN).*, 2, 2006.

[57] N. Mitton and E. Fleury. Distributed node location in clustered multi-hop wireless networks. In *Technologies for Advanced Heterogeneous Networks : First Asian Internet Engineering Conference, (AINTEC'05)*, volume 3837 / 2005, pages 112 – 127, Bangkok, Thailand, December 2005.

[58] N. Mitton and E. Fleury. Distributed node location in clustered multi-hop wireless networks. In *Locality Preserving Distributed Computing Methods (LOCALITY'05)*, Cracow, Poland, September 2005.

[59] N. Mitton and E. Fleury. Distributed node location in clustered multi-hop wireless networks. *GESTS International Transaction on Computer Science and Engineering.*, 21(1), December 2005.

[60] N. Mitton, T. Razafindralambo, and D. Simplot-Ryl. *Theoretical Aspects of Distributed Computing in Sensor Networks*, chapter Position-Based Routing in Wireless Ad Hoc and Sensor Networks. Springer, 2010.

[61] N. Mitton, T. Razafindralambo, D. Simplot-Ryl, and I. Stojmenovic. Hector is an energy efficient tree-based optimized routing protocol for wireless networks. In *Proc. of 4th the Int. Conf. on Mobile Ad-hoc and Sensor Networks (MSN 2008)*, Wuhan, China, December 2008.

[62] N. Mitton, D. Simplot-Ryl, and I. Stojmenovic. Guaranteed delivery for geographical anycasting in wireless multi-sink sensor and sensor-actor networks. In *Proc. of the 28th Annual IEEE Conf. on Computer Communications (INFOCOM 2009)*, Rio de Janeiro, Brazil, April 2009. Short paper.

[63] J.G. Park, H.S Chae, and E.S. So. A dynamic load balancing approach based on the standard rfid middleware architecture. In *Proc. of the 4th IEEE International Conference on e-Business Engineering (ICEBE '07)*, pages 337–340, Washington, DC, USA, 2007.

[64] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing, July 2003. IETF. RFC 3561.

[65] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. of the 2001 conference on applications, technologies, architectures, and protocols for computer communications (Sigcomm'01)*, pages 161–172. ACM Press, 2001.

[66] T. Razafindralambo, N. Mitton, A. Carneiro Viana, M. Dias de Amorim, and K. Obraczka. Adaptive infrastructure deployment for data gathering in intermittently connected networks. In *Proc. of the 8th IEEE Pervasive Computing and Communication (PerCom 2010)*, Mannheim, Germany, 2010.

[67] V. Rodoplu and T. Meng. Minimizing energy mobile wireless networks. *IEEE Journal on Selected Areas*, 17(8) :1333–1347, 1999.

[68] A. Rowstron and P. Druschel. Pastry : Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, Heidelberg, Germany, November 2001.

[69] J. A Sanchez and P. M. Ruiz. Exploiting local knowledge to enhance energy-efficient geographic routing. In *Proc. of the 2nd Int. Conference on Mobile Ad-hoc and Sensor Networks MSN'06*, pages 567–578, December 2006.

[70] N. Santoro and R. Khatib. Labeling and implicit routing in networks. *The computer Journal*, 28 :5–8, 1985.

[71] L. Schmidt, R. Dagher, R. Quilez, N. Mitton, and D. Simplot-Ryl. DHT-based distributed ALE engine in RFID middleware. Research Report 7316, INRIA, 06 2010.

[72] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord : A scalable peer-to-peer lookup service for Internet applications. In *Proc. of the 2001 conference on applications, technologies, architectures, and protocols for computer communications (Sigcomm'01)*, pages 149–160. ACM Press, 2001.

[73] I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel Distributed Systems (TPDS)*, 12(10) :1023–1032, 2001.

[74] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4) :261–268, 1980.

[75] A. C. Viana, M. Dias de Armorim, S. Fdida, and J. Ferreira de Rezende. Indirect routing using distributed location information. In *In Proc. of the 1st IEEE International Conference on Pervasive Computing and Communications (PERCOM 03)*, page 224, Washington, DC, USA, 2003. IEEE Computer Society.

[76] A. Carneiro Viana, N. Mitton, L. Schmidt, and M. Vecchio. A k-layer self-organizing structure for product management in stock-based networks. In *Proc. of the 7th IEEE International Conference on e-Business Engineering (ICEBE 2010)*, Shanghai, China, 2010.

[77] W. Wang, V. Srinivasan, and K-C Chua. Extending the lifetime of wireless sensor networks through mobile relays. *IEEE/ACM Trans. Netw.*, 16(5) :1108–1120, 2008.

[78] ANR VERSO WINGS. Widening interoperability for networking global supply chains. `http://www.wings-project.fr/`, 2009.

[79] B. Y. Zhao, L. Huang, J. Stribling, S.C. Rhea, A.D. Joseph, and J. Kubiatowicz. Tapestry : A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.