République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Batna-2

Faculté des Mathématiques et de l'Informatique

Thèse

en vue d'obtention du diplôme de

Doctorat LMD en Informatique

Titre

MAS-CO : Un modèle multi-agents pour objets coopératifs

Présentée Par

Nadjib BENAOUDA

Soutenue le 28 Juillet 2019

Devant le jury composé de :

Président	Souheila BOUAM	MCA, Université de Batna-2
Rapporteur	Ammar LAHLOUHI	MCA, Université de Batna-2
Examinateurs	Makhlouf ALIOUAT	Prof, Université de Sétif-1
	Faiza TITOUNA	MCA, Université de Batna-2

À mes parents !

Remerciements

Mes remerciements vont, tout d'abord, au « Grand Allah » qui m'a guidé dans le bon chemin et qui m'a donné toute cette force physique et mentale afin d'accomplir, au mieux, ce présent travail.

Mes remerciements vont aussi, à mon encadreur, docteur Ammar Lahlouhi, pour m'avoir proposé un sujet d'actualité, et de m'avoir suivi au cours de la réalisation de mon projet de thèse. Je le remercie vivement pour ses précieux conseils, pour ses riches discussions et pour tout l'effort qu'il a consenti afin que je puisse mener ce projet à terme et cela même avec les soucis de santé qu'il a connus durant ces dernières années. À cette occasion, je lui souhaite un bon rétablissement et un bon retour pour davantage de production scientifique.

Je témoigne également toute ma reconnaissance à Mme Souheila Bouam, Maître de conférences classe A en Informatique à l'université de Batna-2 et chef-Adjointe de la Post-Graduation, pour l'honneur qu'elle me fait de présider ce jury.

Je tiens à exprimer toute ma gratitude à Mr Makhlouf Aliouat, Professeur en Informatique et directeur du Laboratoire des Réseaux et des Systèmes Distribués (LRSD) à l'université de Sétif-1, pour avoir accepté de lire ma thèse et évaluer mon travail.

Je tiens à exprimer aussi toute ma gratitude à Mme Faiza Titouna, Maître de conférences classe A en Informatique à l'université de Batna-2, pour avoir bien voulu juger mon travail et faire partie de mon jury de thèse.

Je remercie également, le professeur Noureddine Zerhouni, chercheur à l'institut FEMTO-ST en France, pour m'avoir accueilli au sein de son équipe de recherche. Je le remercie également avec le docteur Christophe Varnier pour les diverses discussions que j'ai pu entretenir avec eux durant ma période de stage.

Un grand merci aussi, à mes parents, pour leurs conseils précieux et soutien constant tout le long de la réalisation de ma Thèse.

Je finis par remercier, tous les chercheurs que j'ai eu l'honneur de travailler avec eux durant ces années de ce projet de thèse, et qui m'ont permis, par leurs discussions, d'acquérir des connaissances solides dans le domaine de la recherche scientifique. Merci aussi, à toute personne m'ayant apporté le moindre appui au cours de la réalisation de mon travail.

Résumé

Dans cette thèse, nous nous intéressons au domaine des objets coopératifs en étudiant un cas particulier de ce domaine qui est celui des réseaux de capteurs sans fil. Ces réseaux de capteurs ont connu une grande prévalence à cause de leur potentialité à prélever et à faire communiquer une multitude de mesures qui sont pertinentes à notre environnement physique. Sur le plan technique, ils sont également connus pour les ressources restreintes des nœuds capteurs les constituant. Parmi ces ressources, l'énergie limitée qui est mise à la disposition de ces nœuds reste la plus contraignante. Le besoin de les faire fonctionner d'une manière autonome pour de longue durée a suscité les scientifiques à proposer diverses techniques de conservation d'énergie. Parmi ces techniques, nous nous intéressons dans cette thèse à celle d'agrégation des données et de contrôle des puissances de transmission des nœuds.

Cette thèse traite deux principaux problèmes. Le premier consiste à pouvoir établir des structures de routage qui maximisent l'agrégation des données et minimisent la puissance totale de transmission de ces structures, tout en veillant à la délivrance fiable des données agrégées au nœud puits. Pour résoudre ce problème, nous nous sommes basés sur les algorithmes de colonies de fourmis, notre solution multi-agents, et également sur l'algorithme LMST afin de proposer notre premier protocole PALDA (Power-efficient routing based on Ant-colony-optimization and LMST for in-network Data Aggregation). Pour ce qui est du deuxième problème, nous nous sommes intéressés à la délivrance, en temps-réel, des données collectées. Cette considération de cet aspect temps-réel requiert le besoin à pouvoir assurer un compromis entre cette délivrance en délais bornés et la quantité d'énergie qui peut être consommée suite à cette collection. Pour résoudre ce problème et toujours en se basant sur les algorithmes de colonies de fourmis, nous proposons dans cette thèse notre deuxième protocole DPIDA (Delay-bounded and Power-efficient routing for In-network Data Aggregation).

Les deux protocoles ont été évalués en utilisant le simulateur J-sim. Pour cela, nous avons intégré, dans ce dernier, nos diverses propositions avec quatre autres protocoles de la littérature qui ont été utilisés pour des fins de comparaison. Une multitude de simulations ont été conduites et plusieurs instantanées ont été générées avec la considération de divers facteurs clés pertinents à l'utilisation de réseaux de capteurs sans fil. Les résultats de simulation ont montré la supériorité de nos propositions.

Abstract

In this thesis, we are interested in the field of cooperating objects by studying a particular case of this domain which is that of wireless sensor networks. These sensor networks have known a high prevalence because of their potential to collect and communicate a multitude of measurements that are relevant to our physical environment. Technically, they are also known for the limited resources of the sensor nodes constituting them. Among these resources, the limited energy that is available to these nodes remains the most restrictive. The need to operate them autonomously for a long time has prompted scientists to propose various energy conservation techniques. Among these techniques, we focus in this thesis on that of data aggregation and transmission power control.

This thesis deals with two main problems. The first one is relevant to the ability to establish routing structures that maximize data aggregation and minimize the total transmission power of nodes constituting these structures, while ensuring the reliable delivery of the aggregated data to the sink node. To solve this problem, we rely on ant colony algorithms, our multi-agent solution, and also on the LMST algorithm to propose our first protocol which is called PALDA (Power-efficient routing based on Ant-colony-optimization and LMST for in-network Data Aggregation). Regarding the second problem, we are interested in the real-time delivery of the collected data. This consideration of this real-time aspect requires the need to deal with the energy-latency trade-offs. To solve this problem and still based on ant colony algorithms, we propose in this thesis our second protocol called DPIDA (Delay-bounded and Power-efficient routing for In-network Data Aggregation).

Both protocols were evaluated using J-sim simulator. For this evaluation, we have included in this simulator, our various proposals with four other protocols of the literature that were used for comparison purposes. A multitude of simulations were conducted and several snapshots were generated with the consideration of various key factors relevant to the use of wireless sensor networks. The simulation results showed the superiority of our proposals.

Table des matières

Table des figures

Introduction

1.1 Contexte général

D'un jour à un autre, notre vie a changé complètement avec l'avènement et l'émergence de nouvelles technologies. Ces technologies qui concernent principalement le domaine de l'information et de la communication ont connu une grande prévalence ces dernières années dans différents domaines qui interviennent dans notre vie quotidienne en donnant naissance ainsi à de nouveaux concepts ainsi qu'à de multiples tendances prometteurs. Ainsi, l'immense émergence des réseaux téléphoniques sans fil avec les progrès remarquables qu'ils ont connus, et aussi les avancements spectaculaires concernant les technologies des réseaux informatiques, et en particulier l'Internet ont contribué en grande partie à ce changement. Encore, des technologies comme l'informatique dite ubiquitaire ou pervasive, les systèmes embarqués, les réseaux de capteurs sans fil, ainsi que d'autres technologies ont contribué aussi, chacune à sa façon, et d'une manière remarquable à ce changement et sont devenues, par conséquent, des domaines de recherche très actifs et très porteurs.

L'émergence de ces technologies, en plus des progrès réalisés dans le domaine de l'intégration et de la miniaturisation, ont conduit à l'émergence d'une nouvelle vision où l'utilisateur, proprement dit, est entouré et il est mis à sa disposition divers objets et dispositifs intelligents, sous la forme de divers appareils, de machines et d'entités physiques visant, à la fin, le bien être de l'être humain. Ce nombre d'objets, selon plusieurs études de marché [1], va continuer à croître d'une façon énorme. Encore, ces objets ne sont pas isolés, ces objets sont dotés de capacités de calcul, de communication et d'interaction avec l'environnement, permettant ainsi à ces objets de communiquer et de coopérer entre eux et avec d'autres entités, y compris avec l'être humain, et ce, afin d'apporter de la valeur ajoutée.

Par ailleurs, la coopération entre ces différents objets représente un concept important pour faire face aux difficultés dues à la complexité croissantes des applications, ainsi qu'une clé de voûte pour la résolution de nombreux problèmes faisant interve-

nir plusieurs entités. En fait, c'est à travers cette coopération que des applications innovantes émergent.

Le domaine des objets coopératifs a émergé suite à ce contexte. Ces objets coopératifs sont dans le cas le plus général, des petits appareils informatiques dotés de capacités de communication sans fil et qui sont en mesure de coopérer et de s'organiser d'une manière autonome en un réseau de capteurs, d'actionneurs et d'unités de traitement afin de réaliser une tâche commune [1]. Un réseau de capteurs sans fil illustre bien cette coopération [1]. Un tel réseau est constitué d'un ensemble d'objets qui peuvent mener individuellement des tâches de perception, d'actionnement, de communication et de calcul. Cependant, l'objectif final visé n'est atteint qu'avec la mise en coopération de tous ces objets.

Le domaine des objets coopératifs est très vaste et envisage un grand éventail d'applications et systèmes. En fait, il envisage la mise en connexion et en coopération d'un grand nombre de dispositifs embarqués comme les réseaux de capteurs et d'acteurs, des appareils électroménagers, des smartphones, etc. Vu le caractère général de ce domaine et les directives de conception qui varient d'un système à un autre, nous nous focalisons dans cette thèse sur l'étude d'un exemple particulier de ces objets coopératifs qui est celui des réseaux de capteurs sans fil.

1.2 Problématique

Les nœuds capteurs sans fil sont des petits dispositifs qui sont dotés d'une multitude de capteurs, d'un micro-contrôleur et d'un émetteur-récepteur radio. Les capteurs de ces nœuds leurs permettent à prélever des mesures pertinentes à leur environnement physique pour, à la suite, les délivrer à d'autres entités pour plus de traitement. La possibilité à collecter et à communiquer ces informations connexes à notre environnement physique ont favorisé l'applicabilité de ces réseaux de capteurs sans fil à divers domaines applicatifs comme celui des soins de santé, de la domotique, de l'industrie, d'agriculture, de surveillance de l'environnement, etc.

Les nœuds capteurs sont généralement alimentés par des batteries de capacités restreintes, et donc, qui les rendent opérationnels pour une durée de temps limitée. En outre, le remplacement de ces batteries peut être couteux et les nœuds capteurs peuvent être déployés dans des environnements hostiles où l'intervention humaine peut être impossible. Dans ces conditions, les nœuds capteurs doivent être en mesure de fonctionner en toute autonomie pour de longues durées qui peuvent être de l'ordre de plusieurs mois, voire des années. Ce besoin à prolonger la durée de vie de ces réseaux de capteurs sans fil a motivé le développement d'algorithmes et de protocoles économes en énergie qui peuvent optimiser la consommation énergétique des nœuds capteurs.

Vu ce besoin d'économiser l'énergie des nœuds, plusieurs techniques de conservation

de cette énergie ont été proposées. L'agrégation des données en est parmi les plus connues. Elle exploite les corrélations qui peuvent caractériser les données prélevées par les nœuds capteurs afin d'éliminer toute redondance potentielle entre ces données, et donc, afin de minimiser la taille et le nombre de messages échangés dans le réseau. Cela contribue à minimiser l'énergie qui est dissipée par ces nœuds durant la collection des données étant donné que la partie importante de cette énergie est consommée durant les communications.

Trois composants sont nécessaires afin de mener cette agrégation de manière efficace [2] : (1) Schémas de routage, qui définissent la façon dont les données sont routées vers le nœud puits en favorisant leur convergence spatiale ; (2) Plans d'agrégation, qui favorisent la convergence temporelle de ces données en définissant les temps d'attente que doit chaque nœud attendre avant l'acheminement et l'agrégation des données reçues ; et (3) fonctions d'agrégation, qui définissent comment les données sont agrégées.

L'essentiel du travail présenté dans cette thèse se focalise sur les schémas de routage. En fait, avec de tels schémas et, par rapport au routage classique, où les données sont routées le long du plus court chemin séparant un nœud source d'un autre de destination, il est important que les décisions de routage prises au niveau des nœuds capteurs favorisent la convergence spatiale des données, et donc favoriser le chevauchement précoce des chemins, afin que ces données puissent être agrégées. Cependant, la réalisation de cette tâche n'est pas une chose aisée, car, pouvoir mener de manière optimale cette agrégation nécessite la construction d'un arbre de Steiner qui est connue pour être NP-Hard.

Outre cette agrégation, la connectivité entre les différents nœuds du réseau est définie par leurs puissances respectives de transmission. Ces nœuds ne sont pas obligés d'utiliser leurs puissances maximales de transmission, mais par contre, ils peuvent les diminuer afin de conserver davantage d'énergie.

Suite à ce qui a été dit, le premier problème visé dans cette thèse consiste à pouvoir :

1. Établir une structure de routage qui connecte l'ensemble de nœuds sources avec le nœud puits. Cette structure doit maximiser le nombre de routes qui se chevauchent afin de maximiser l'agrégation des données. Cette structure correspond à un arbre de Steiner,
2. Assigner à chaque nœud de cette structure une puissance de transmission appropriée,
3. Minimiser la somme des puissances de transmission assignées,
4. Et assurer une assignation de puissance de transmission symétrique qui considère la bidirectionnalité des liens entre les nœuds de la structure formée afin de pouvoir délivrer les données prélevées de manière fiable. Cette considération de ce genre de liens facilite le fonctionnement de certains protocoles de la couche MAC comme

le IEEE 802.11 qui requière l'échange des messages RTS, CTS et ACK entre un nœud émetteur et un autre récepteur.

Pour pouvoir résoudre ce problème, il est important de faire recours à des systèmes auto-organisés. En fait, cette auto-organisation avec laquelle la fonctionnalité globale émerge uniquement à partir de nombreuses interactions entre les différents éléments du système, ayant chacun son propre contrôle, et cela, sans se référencer à aucun contrôle externe ou centralisé, est considérée comme un paradigme de contrôle des systèmes massivement distribuée comme les réseaux de capteurs sans fil. En fait, Ces derniers impliquent un nombre important de dispositifs travaillant ensemble pour l'accomplissement de leur objectif commun. L'administration de ce grand nombre qui nécessite des algorithmes et des mécanismes évolutifs accompagnée de ressources limitées de ces objets requièrent, à leur tour, des algorithmes et des comportements locaux simples exécutés par chacun de ces objets et se basant seulement sur des informations locales. C'est à travers les interactions entre ces objets que doit émerger la solution globale qui reflète la bonne mise en marche de cette coordination.

Vu la pertinence de cette notion d'auto-organisation, nous nous basons, dans cette thèse, sur les algorithmes de colonies de fourmis afin de résoudre ce problème d'agrégation et d'assignation des puissances de transmission. Ces algorithmes ont été inspirés du comportement collectif des fourmis recherchant un chemin entre leur colonie et une source de nourriture. Ils mettent en opération une colonie de fourmis artificielle, et donc un système multi-agents, afin de résoudre une classe particulière de problèmes s'intéressant à la détermination du plus court chemin.

Avec ces systèmes, la coopération et l'auto-organisation sont deux notions basiques qui leurs permettent à résoudre des tâches complexes. En fait, même si une seule fourmi peut trouver un chemin quelconque entre son nid et une source de nourriture, c'est seulement à travers la présence simultanée de plusieurs fourmis avec leur travail collectif que le plus court chemin peut être déterminé. Cette synergie entre fourmis, via leurs comportements simples, qui génère la résolution d'un tel problème, assez complexe, est un bon exemple d'un système auto-organisé.

En se basant également sur ces algorithmes de colonies de fourmis, nous considérons dans cette thèse un deuxième problème qui est pertinent aux réseaux de capteurs sans fil qui requièrent une délivrance en temps réel des données. Cette exigence est importante pour beaucoup d'applications. Par exemple, avec des applications de soins de santé qui sont de nature critique, réagir, en temps opportun, à toute situation d'urgence telles que les crises cardiaques ou les chutes soudaines est très important. Dans ce cas, la délivrance des données des nœuds aux utilisateurs finaux doit se faire rapidement et en temps borné. Ainsi, avec une application de sureté publique qui assistent, par exemple, les employés pratiquant des travaux dangereux comme des mineurs ou des secouristes, la délivrance en temps opportun des données est très importante, car, dans

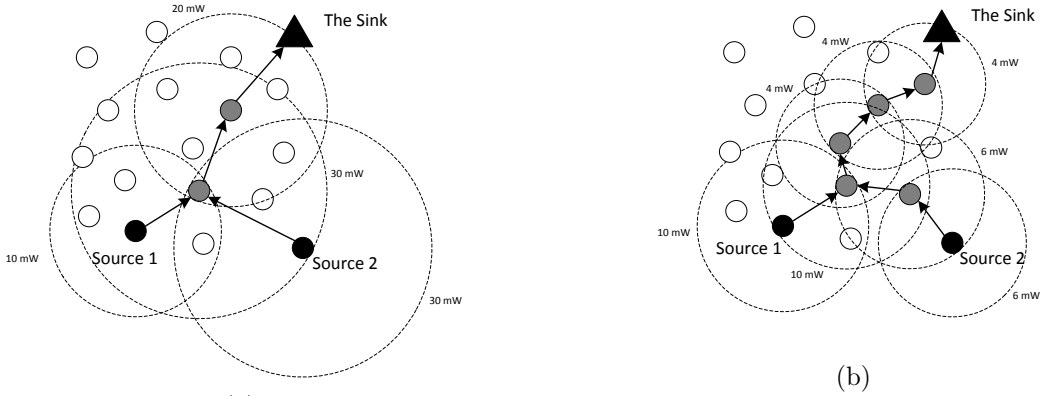


FIGURE 1.1 – Deux structures de routage différentes connectant deux nœuds capteurs sources avec le nœud puits et leurs énergies totales consommées. (1.1a) 90 mW avec une longue route de 3 sauts. (1.1b) 44 mW avec une longue route de 6 sauts.

des situations d'urgence, toute intervention qui se fait en retard risque de coûter des vies. Outre ces deux exemples, beaucoup d'autres applications peuvent être citées.

Sur le plan conceptuel, la considération de ces communications en temps réel requiert l'assurance d'un compromis entre la délivrance en délais bornés des données et la minimisation de l'énergie qui est consommée durant cette délivrance. Par rapport au premier problème où les nœuds ont la liberté de réduire au maximum leurs puissances de transmission, cette minimisation devient contraignante lorsqu'on à faire à des applications qui requièrent des communications en temps-réel. Cela est dû au fait qu'une minimisation agressive des puissances de transmission génère des routes avec un grand nombre de sauts, et par conséquent, un temps de latence important.

Cette constatation est illustrée par la figure (1.1) qui montre deux structures de routages différentes connectant deux nœuds capteurs sources avec le nœud puits. Comme nous pouvons le voir, avec la deuxième structure de routage (figure (1.1b)) nous pouvons conserver plus d'énergie par rapport à la première structure (figure (1.1a)) étant donné que les nœuds capteurs utilisent des puissances de transmission plus petites. Cependant et, comme conséquence, nous allons avoir des chemins avec un grand nombre de sauts, et donc, les données acheminées vont connaitre des délais de transit importants.

Suite à ce que nous venons de citer, nous nous intéressons, dans un deuxième temps, au problème de la collection efficace et en temps réel des données prélevées par un certain nombre de nœuds capteurs sources. Le problème visé ici est similaire au premier problème avec seulement l'ajout de la contrainte d'établissement de chemins à délais bornés.

1.3 Contributions

Suite aux deux problèmes présentés dans la section précédente et outre l'exploration et l'étude des travaux connexes au problème d'agrégation des données dans les réseaux de capteurs sans fil et à l'application des algorithmes de colonies de fourmis à ce problème d'agrégation, nos principales contributions sont en nombre de deux.

Pour le premier problème qui consiste à pouvoir établir des structures de routage qui maximisent l'agrégation des données et minimisent la puissance totale de transmission des nœuds constituant ces structures, nous proposons le protocole PALDA[3] (Power-efficient routing based on Ant-colony-optimization and LMST for in-network Data Aggregation). C'est un protocole réactif où le réseau reste inactif et réagit lorsque des évènements d'intérêt arrivent en construisant la structure de routage nécessaire à la délivrance des données prélevées au nœud puits. Durant cette construction, les nœuds qui détectent le même évènement sont organisés en un même cluster. Ensuite, un arbre qui connecte les différents cluster-heads avec le nœud puits est formé. Le protocole proposé se base sur deux éléments clés. Le premier est pertinent à l'exploitation de l'algorithme LMST[4] afin de construire, au sein de chaque cluster, un arbre ayant un coût énergétique minimal et qui sert à la collection des données des membres de chaque cluster. Le deuxième élément est pertinent à l'utilisation des algorithmes de colonies de fourmis afin de former les routes entre les différents cluster-heads et le nœud puits en déterminant les nœuds relais et leurs puissances de transmission.

Les travaux antérieurs tels que, DA-ACA[5], MANSI[6], ECMANSI[7] et la famille d'algorithmes DAACA[8], qui se basent sur les algorithmes de colonies de fourmis pour l'établissement d'une structure de routage qui correspond à un arbre de Steiner, laissent les agents fourmis explorent le réseau et chercher dans toutes les directions afin de pouvoir se rencontrer avec des nœuds de l'arbre de routage existant, et par conséquent, assurer le chevauchement des routes. Différemment à ces travaux, nous établissons une structure géométrique sous-jacente afin d'orienter la recherche des fourmis seulement vers les parties nécessaires du réseau au lieu de les laisser chercher aveuglement dans tous les sens. Cette façon de faire contribue à accélérer la vitesse de convergence de la solution finale.

Suivant la façon de construction de cette structure géométrique, deux variations de PALDA sont proposées. La première, PALDA-S (Statique) où les routes sont établies suivant l'ordre d'occurrence des évènements et sont utilisées durant la période totale de leurs occurrences. Et la deuxième, PALDA-D (Dynamique) où les routes sont formées selon les distances des cluster-heads du nœud puits et elles sont reconstruites à chaque occurrence d'un nouvel évènement afin d'améliorer la qualité de l'arbre final de routage. Des simulations ont été réalisées afin d'évaluer les performances de notre protocole proposé avec ses deux variations en les comparant avec deux autres protocoles existants

DST[9] et ECMANSI[7]. Les deux variations PALDA-S et PALDA-D avec les deux protocoles de comparaison ont été intégrés au simulateur J-sim. Ces protocoles ont été comparés en considérant divers facteurs : la vitesse de convergence, le nombre de nœuds, le nombre d'évènements et leur durée. Les résultats de simulation ont montré la supériorité de nos propositions par rapport aux deux protocoles de comparaison considérés.

Pour ce qui est du deuxième problème où nous nous intéressons à la délivrance en temps réel et à basse consommation énergétique des données prélevées par certains nœuds capteurs au nœud puits, nous proposons le protocole DPIDA[10] (Delay-bounded and Power-efficient routing for in-network Data Aggregation). Ce protocole DPIDA emprunte certaines idées d'un autre protocole, appelé CSTMAN[11], et introduit d'autres qui améliorent ce dernier en ce qui concerne plusieurs éléments. Autrement dit, nous proposons deux nouvelles stratégies qui définissent la façon dont les chemins des nœuds sources doivent être fusionnés. Ces deux stratégies ont une implication directe sur la qualité de la solution obtenue en terme de puissance de transmission totale de l'arbre final émergé. En outre, nous proposons une nouvelle stratégie afin de restreindre l'espace de recherche des fourmis en les orientant seulement vers la partie nécessaire de cet espace. Cela aide à accélérer la vitesse de convergence de cette solution. Outre ces deux contributions et par rapport à CSTMAN, DPIDA diminue la charge engendrée par les messages de contrôle.

Comme PALDA, des simulations ont été également conduites avec le simulateur J-sim pour valider DPIDA avec ses deux variations en les comparant aux deux protocoles CSTMAN[11] et DEDA[12]. Les résultats de simulation avec des instantanés des arbres d'agrégation construits montrent la capacité de DPIDA à assurer un compromis entre la délivrance, en délais bornés, des données et l'énergie totale consommée. Ainsi, ils montrent les meilleures performances de DPIDA par rapport à CSTMAN et DEDA.

1.4 Organisation de la thèse

Afin de décrire notre façon de réfléchir, cette thèse, présente les principaux éléments académiques intervenant dans nos contributions et aussi de la façon de les mettre en œuvre. Elle est organisée de la façon suivante :

Le chapitre 1 introduit la notion des objets coopératifs et des réseaux de capteurs sans fil et détaille les principales notions liées à ces réseaux. Ces notions incluent les composants d'un nœud capteur, les sources de dissipation d'énergie, les techniques de conservation de cette énergie, les défis de conception et les domaines d'application de ces réseaux.

Le chapitre 2 se focalise entièrement sur la technique d'agrégation des données. Deux grandes parties sont distinguées avec ce chapitre. La première partie illustre le

principe de cette technique d'agrégation et motive son utilisation en exposant quelques exemples applicatifs. Ainsi, les composants essentiels d'un protocole d'agrégation sont brièvement présentés. La deuxième partie détaille les travaux connexes à cette technique d'agrégation en se focalisant sur les travaux qui s'intéressent aux schémas de routage et à celle qui considèrent, en plus de l'établissement de ces schémas, la délivrance à délai borné des données prélevées.

Le chapitre 3 est consacré aux algorithmes de colonies de fourmis et à leur application au problème d'agrégation dans les réseaux de capteurs sans fil. Dans ce chapitre, les deux notions d'auto-organisation et de mise en réseau bio-inspirée sont présentées. L'origine de ces algorithmes, l'analogie des comportements des fourmis avec les réseaux de communication et l'ingénierie de ces comportements pour permettre leur applicabilité aux problèmes techniques en général et aux réseaux de communication en particulier sont détaillés. Ainsi, les principaux travaux qui appliquent ces algorithmes de colonies de fourmis au problème d'agrégation sont présentés.

Le chapitre 4 présente notre premier protocole proposé PALDA. Il présente une formulation du problème traité, le modèle réseau adopté, une description du protocole et la méthodologie de simulation suivie avec les résultats de simulation obtenus et une interprétation de ces résultats.

D'une manière similaire au chapitre 4, le chapitre 5 détaille notre deuxième protocole proposé DPIDA avec les différents résultats de simulation obtenus.

Cette thèse se termine par une conclusion qui récapitule le travail présenté par cette contribution et donne quelques perspectives des différentes propositions présentées.

Chapitre 2

Objets coopératifs et réseaux de capteurs sans fil

2.1 Introduction

Le domaine des objets coopératifs est un domaine très vaste qui se focalise sur la coopération entre un ensemble de dispositifs informatiques dotés de capacités de calcul, de communication et de perception et/ou d'actionnement. Un réseau de capteurs sans fil est un exemple typique de ce domaine où l'objectif principal de l'application n'est atteint qu'avec la coopération d'un ensemble de nœuds capteurs capables de percevoir leur environnement et de communiquer leurs données à d'autres entités spécialisées. Dans ce chapitre, nous présentons dans un premier temps cette notion d'objets coopératifs, ensuite, nous nous focalisons sur les réseaux de capteurs sans fil. Dans la deuxième partie, les diverses notions liées à ces réseaux sont présentées. Particulièrement, ces derniers sont définis et la structure d'un nœud capteur avec ses divers composants sont exposés. Les diverses sources de consommation d'énergie d'un nœud capteur sont énumérées et l'ensemble des techniques de conservation de cette énergie qui existent dans la littérature sont résumées. En outres, les défis qui entravent la conception et le développement de tout algorithme et protocole qui est dédié à ces réseaux sont présentés. Afin d'illustrer l'applicabilité de ces réseaux, nous présentons dans une dernière section leurs principaux domaines d'applications.

2.2 Objets coopératifs

L'évolution que connait le domaine des systèmes embarqués accompagnée des progrès réalisés dans le domaine de l'intégration et de miniaturisation ont conduit à l'émergence d'une nouvelle sorte de dispositifs, de machines et d'objets physiques qui sont dotés de capacités de communication, de calcul et d'interaction avec l'environnement.

Ces objets ou entités varient de simples dispositifs comprenant des capteurs, des unités de calcul, des interfaces de communication et des sources énergétiques jusqu'à d'autres dispositifs plus sophistiqués comme des dispositifs portables, des machines électroménager et même des véhicules. En fait, même si chacun de ces objets peut être individuellement utilisé, c'est à travers la mise en coopération de ces objets que de nouvelles applications innovantes émergent.

Suivant ce contexte, la notion d'objets coopératifs a été proposée. Cette notion a connu sa première émergence en 2006 suite au projet européen *Embedded WiSeNts* [13]. La définition qui a été présentée dans le plan stratégique de ce projet est la suivante [13] :

« Dans le sens abstrait, un objet coopératif est une entité unique ou une collection d'entités qui consistent en un ensemble de : capteurs, contrôleurs (processeurs d'information), actionneurs ou d'objets coopératifs qui communiquent les uns avec les autres et sont en mesure d'accomplir, plus au moins, d'une manière autonome un objectif commun. »

Les capteurs délivrent les informations pertinentes à l'état du monde physique. Les actionneurs modifient l'environnement suite aux commandes données. Les contrôleurs traitent les données délivrées par les capteurs et passent les commandes appropriées aux actionneurs afin d'achever les objectifs désirés. L'inclusion d'autres objets coopératifs comme faisant partie d'un même objet coopératif souligne la possibilité à pouvoir organiser cet ensemble d'objets de manière hiérarchique, et donc, à avoir des structures arbitraires et complexes.

Plus tard, un autre projet européen qui se focalise complètement sur le domaine des objets coopératifs a été lancé, ce projet qui est baptisé CONET (*Cooperating Objects Network of Excellence*) a utilisé dans son premier plan stratégique de recherche (*Research Roadmap on Cooperating Objects*) [1] la même définition. Pour la deuxième édition (*The emerging domain of Cooperating Objects*) [14], elle a été révisée pour mettre l'accent sur les aspects de coopération. Dans cette deuxième édition, le terme d'objets coopératifs a été défini comme suit :

« Les objets coopératifs sont constitués de dispositifs informatiques embarqués dotés de capacités de communication ainsi que des capacités de détection ou d'actionnement. Ces dispositifs peuvent coopérer et s'organiser d'une manière autonome en réseau afin d'accomplir une tâche commune. La vision des objets coopératifs consiste à s'attaquer à la complexité émergente via la coopération et la modularité. À l'égard de cette vision, la capacité de communiquer ainsi que d'interagir avec d'autres objets et/ou l'environnement devient une condition essentielle. Alors que dans de nombreux cas, la coopération est spécifique à l'application, la coopération entre des dispositifs hétérogènes peut être prise en charge par des abstractions partagées. »

Cette dernière définition a été même révisée afin d'améliorer sa clarté comme suit [14] :

« Les objets coopératifs sont des systèmes modulaires qui regroupent un ensemble de dispositifs autonomes et hétérogènes. Ces dispositifs poursuivent un objectif commun via la coopération dans la réalisation des calculs et l'interaction avec l'environnement à travers la perception et l'actionnement. »

Cette dernière définition identifie les caractéristiques clés suivantes des objets coopératifs [14, 15] :

- **Modularité :** cette modularité permet le développement de réseaux évolutifs qui peuvent être adaptés aux nouveaux besoins. En fait, une conception modulaire permet le remplacement d'un dispositif donné par un autre plus performant ou l'ajout de nouveaux dispositifs qui étendent la fonctionnalité du réseau.
- **Autonomie :** l'autonomie souligne la capacité des dispositifs à décider eux-mêmes de leur participation à la réalisation de l'objectif visé. À cet égard, chacun de ces dispositifs peut dédier une fraction de ses ressources ou ses fonctionnalités afin de faire émerger la fonctionnalité globale du système. La décision de coopérer ou non peut se baser sur divers facteurs. Un dispositif peut considérer par exemple son niveau énergétique courant ou ses interactions antérieures avec les autres dispositifs.
- **Hétérogénéité :** les dispositifs constituant un réseau d'objets coopératifs peuvent appartenir à diverses disciplines. Un tel réseau peut mettre en coopération, par exemple, des nœuds capteurs et des robots ayant des caractéristiques totalement différentes, non seulement en termes des capacités de traitement et de stockage, mais aussi sur le plan hardware.
- **Traitement :** comme on vient de le voir, un réseau d'objets coopératifs peut considérer des dispositifs de différentes natures ayant des capacités de traitement différentes. Cependant, ces capacités doivent, au minimum, permettre à un dispositif donné à prendre ses propres décisions et à communiquer avec d'autres dispositifs.
- **Interaction avec l'environnement :** cette interaction se fait à travers les capteurs et les actionneurs qui sont à la disposition des différents dispositifs. Les capteurs permettent la perception de l'état de l'environnement alors que les actionneurs changent cet état.
- **Communication :** même si elle n'est pas citée dans la dernière définition, la communication est une caractéristique très importante qui est indispensable à la coopération. Cette communication entre dispositifs peut se faire de manière explicite via l'échange de messages par voie filaire ou sans fil. Le contenu des

messages échangés se varie. Il peut consister simplement en l'état d'un dispositif donné, comme il peut représenter un plan complet d'actions. Outre cette première façon à communiquer, deux autres techniques qui travaillent par observation à travers les capteurs disponibles peuvent être citées. La première consiste simplement à reconnaitre les actions des autres dispositifs (e.g. les mouvements d'un actionneur). En revanche, les effets des actions effectuées par les autres dispositifs peuvent être perçus (e.g. l'augmentation de la température causée par un chauffage).

- **La poursuite d'un objectif commun :** un réseau d'objets coopératifs est déployé principalement afin de réaliser un certain objectif donné. De manière générale, les dispositifs n'ont aucune idée sur l'objectif dans sa globalité. Mais par contre, chacun d'eux possède des connaissances spécifiques à son champ d'expertise et exécute des tâches spécialisées qui contribuent à la réalisation de cet objectif.
- **Coopération :** avec un réseau d'objets coopératifs, la coopération est intentionnelle est dirigée toujours par un objectif. Donc, sans objectif, et donc, sans tâches, il n'y a aucun besoin à coopérer. En fait, la participation de tous les dispositifs est indispensable à la réalisation de l'objectif visé.

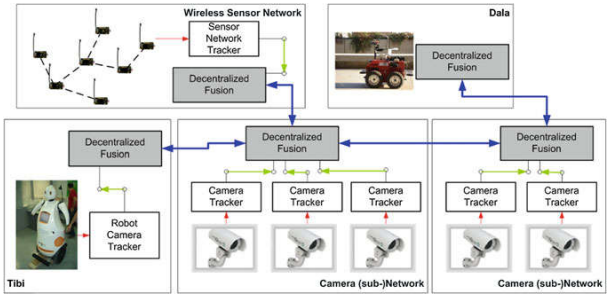
Comme nous pouvons le constater à partir de ces définitions, ces systèmes consistent en un ensemble d'entités ou d'objets qui coopèrent ensemble afin d'achever un objectif commun de perception ou de contrôle. En fait, ces définitions montrent bien l'ampleur de ce domaine d'objets coopératifs et le nombre important d'applications dont il peut envisager. La figure (2.1) montre quatre scénarios applicatifs qui ont été conduits et réalisés par la communauté des objets coopératifs à travers les différents projets s'intéressant à ce domaine [15]. Comme nous pouvons le constater à partir de cette figure, ces scénarios sont totalement différents. Le premier, illustré par la figure (2.1a), montre un réseau de capteur sans fil qui est déployé afin de surveiller des ponts ferroviaires. Ces expériences ont été conduites dans la ville Stockholm de la suède. Un tel réseau regroupe un ensemble de nœuds capteurs, et donc des objets, qui coopèrent afin de détecter toute détérioration potentielle de ces structures qui peut avoir une conséquence significative en termes de pertes de vies humaines.

La figure (2.1b) illustre une description du système de fusion des données qui sont prélevées par les différents capteurs dans le projet URUS [1] [16]. Avec ce projet, le but consistait à concevoir et à développer une architecture qui intègre un ensemble d'objets hétérogènes regroupant des robots, des capteurs intelligents de différentes sortes (e.g. des caméras, des capteurs acoustique, etc.) et des dispositifs intelligents (e.g. PDA). Cet ensemble d'objets sont mis en coopération pour des tâches d'assistance et de guidage

1. Ubiquitous Networking Robotics in Urban Sites

(a) Un réseau de capteurs sans fil qui est déployé pour la surveillance des ponts ferroviaires.

(b) Une description du système de perception dans le projet URUS où un ensemble d'objets coopératifs hétérogènes (nœuds capteurs, des caméras, des robots, etc) coopèrent pour des tâches d'assistance et de guidage des personnes dans des zones urbaines.



(c) Coopération entre des véhicules aériens sans pilote et des nœuds capteurs pour la tâche de maintenance du déploiement des nœuds.

(d) Architecture du système *Bonn-Sens* mettant en coopération des objets de perception (e.g. des smartphones) et d'autres objets collecteurs (e.g. des ordinateurs portables) pour des tâches de sécurité publique.

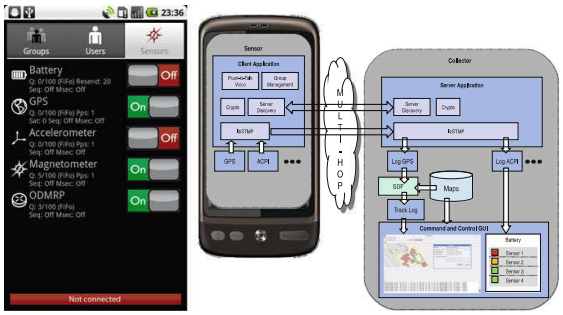


FIGURE 2.1 – Quelques scénarios applicatifs des objets coopératifs.

des personnes dans des zones urbaines. Par exemple, cet ensemble peut guider une personne dans ses déplacements afin de l'orienter vers sa destination souhaitée. Ce scénario illustre bien le bénéfice qu'on peut avoir via la mise en coopération de divers systèmes complémentaires. Les différentes caméras déployés peuvent assurer une vue globale de la zone urbaine couverte. Cependant, le fait qu'ils sont statiques, contraigne leur capacité à surveiller des angles morts. Avoir des robots mobiles qui portent avec eux des caméras peut soulever cette contrainte en les faisant déplacer vers de telles zones. L'ensemble des nœuds capteurs distribués peuvent participer dans les tâches de localisation et d'identification.

La figure (2.1c) illustre un troisième scénario qui a été implémenté dans le projet PLANET [2] [17]. Avec ce scénario, un réseau de capteurs sans fil est déployé pour la surveillance de la pollution dans le parc national de Doñana en Espagne. Ce réseau est censé fonctionner pour une longue durée étant donné que l'accès à ce parc est autorisé pour seulement quelques jours par année. Afin de maintenir le réseau de capteurs sans fil fonctionnel en évitant tout partitionnement potentiel du réseau, des véhicules aériens sans pilote sont déployés afin de remplacer les nœuds capteurs défaillants par d'autres.

Le dernier scénario, qui est illustré par la figure (2.1d), montre l'architecture du système *BonnSens* qui est conçu pour l'assistance des différents employés pratiquant des travaux de nature critique comme les secouristes et les mineurs [15]. Ce système considère l'utilisation d'un ensemble d'objets de perception comme des smartphones qui sont dotés par divers types de capteurs et qui peuvent être portés par chaque usager du système. Ces objets forment un réseau afin de relayer les différentes données perçues vers un autre objet collecteur qui peut être un ordinateur portable.

Ces quatre scénarios illustrent l'ampleur de ce domaine des objets coopératifs et la diversité des applications qui peuvent être imaginées. Pour cela et vu les directives de conception qui peuvent être différentes d'une application à une autre, nous nous intéressons dans cette thèse aux réseaux de capteurs sans fil représentant un exemple typique de ce domaine [18]. Comme nous allons le voir dans la section suivante, ces réseaux consistent en un ensemble d'objets qui peuvent mener individuellement des opérations de perception, d'actionnement, de calcul et de communication. Cependant, l'ultime objectif derrière ces réseaux n'est atteint qu'avec la mise en coopération de tous ces objets.

2.3 Réseaux de capteurs sans fil

Les réseaux de capteurs sans fil sont composés d'une collection de petits dispositifs ayant des capacités de perception, de traitement et de communication. L'idée principale derrière ces réseaux consiste à avoir des dispositifs pouvant prélever des mesures

2. Platform for the Deployment and Operation of Heterogeneous Networked Cooperating Objects.

pertinentes au monde physique, i.e. température et humidité, pour les délivrer ensuite à une station de base pour plus de traitement où des décisions sont prises suite à ces mesures reçues. Donc, les tâches de perception et de communication doivent être réalisées par ce qu'on appelle des *nœuds capteurs* qui collaborent ensemble pour former un réseau ad hoc. La figure (2.2) illustre un tel réseau. Comme nous pouvons le constater à partir de cette figure, un ensemble de nœuds capteurs qui se communiquent ensemble via des interfaces de communication sans fil forment un réseau ad hoc à sauts multiples. Une station de base, appelée aussi nœud puits, et qui fait partie de ce réseau formé, est mise en place afin de récupérer les mesures prélevées par les nœuds capteurs et également pour pouvoir contrôler ces nœuds [19, 20]. Donc, dans un réseau de capteur sans fil, aucune communication n'est faite entre des nœuds arbitraires du réseau. Mais par contre, des nœuds spécifiques émettent leurs données prélevées vers des nœuds puits bien définis.

Outre les opérations de communication, i.e. l'accès au canal sans fil et le routage, qui sont réalisées par tous les nœuds du réseau, les principaux rôles suivants qui dépendent de la tâche de chaque nœud sont à distinguer dans un réseau de capteurs sans fil [19, 20, 21] :

- **Perception :** chaque nœud capteur est doté, pratiquement, d'une multitude de capteurs de différents types responsables du prélèvement des mesures pertinentes aux phénomènes observés.
- **Puits de données :** correspondent aux nœuds qui sont intéressés par les divers évènements qui arrivent. Donc, ils sont intéressés par les données qui sont émises par les nœuds capteurs.
- **Actionnement :** même si la surveillance de l'environnement physique est la tâche qu'on attend le plus avec ce genre de réseaux, il n'est pas rare d'inclure d'autres nœuds, dits nœuds acteurs, comme faisant partie du même réseau, et ayant comme rôle la modification de l'état de leur environnement immédiat en opérant des actions qui sont déclenchées généralement suite aux données prélevées par les nœuds capteurs. Ces nœuds acteurs représentent également des puits de données. Dans la littérature, le nom qui est communément attribué à de tels réseaux qui mettent en coopération des nœuds capteurs et d'autres nœuds acteurs est appelé *réseaux de capteurs et d'acteurs sans fil* [22].
- **Traitement :** chaque nœud du réseau est doté d'éléments de traitement responsables d'effectuer les diverses opérations de communication et de traitements des données prélevées à partir de l'environnement et reçues des autres nœuds.

Pour mieux comprendre les caractéristiques de ces réseaux de capteurs sans fil et les contraintes qui entravent leur conception, il est bien de savoir les éléments qui constituent un nœud capteur. Comme le montre la figure (2.3), un nœud capteur comporte

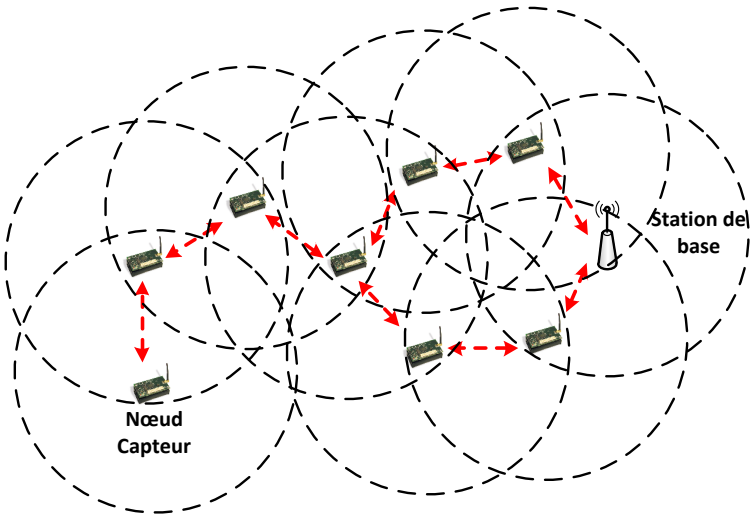


FIGURE 2.2 – Un réseau de capteurs sans fil.

une unité de traitement, i.e. un micro-contrôleur, qui est associé avec une mémoire et, éventuellement, avec une autre mémoire de stockage permanente (e.g. mémoire flash). Un nœud capteur comporte également plusieurs capteurs qui leur permet de prélever des mesures du monde physique et un émetteur-récepteur radio qui lui permet de communiquer ses données.

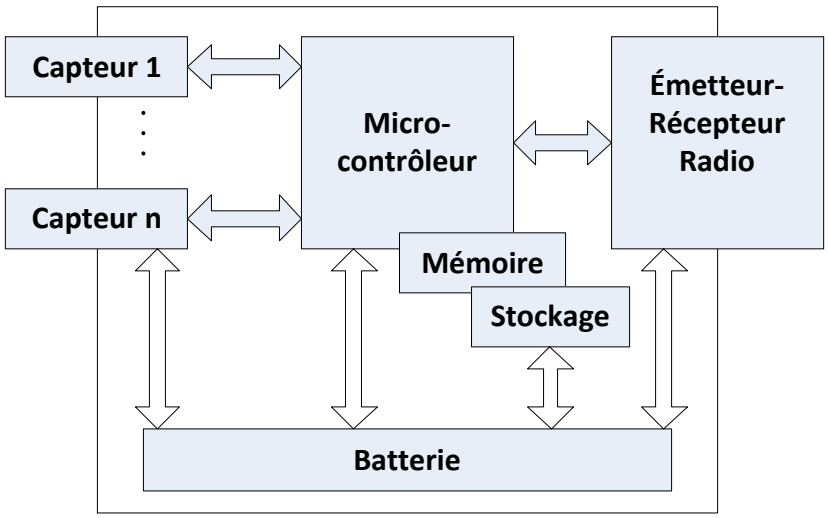


FIGURE 2.3 – Les composants d'un nœud capteur sans fil.

En fait, ces nœuds capteurs sont des systèmes embarqués avec des ressources qui sont généralement limitées. La table (2.1), adaptée de [23], illustre les caractéristiques de quelques nœuds capteurs disponibles sur le marché et qui sont communément utilisés par la communauté scientifique pour des fins d'évaluation et d'expérimentation des algorithmes et des méthodes proposés. Il est clair à partir de ce tableau que ces nœuds capteurs ont des ressources fortement limitées. À cet égard, des calculs intenses ne peuvent pas être réalisés vu les capacités de traitement et de stockage disponibles.

De la même façon, un nœud capteur ne peut émettre qu'avec un débit limité. Outre toutes ces ressources, ces nœuds capteurs sont dotés de batteries qui rendent ces nœuds fonctionnels que pour une durée de temps limitée. Toutes ces contraintes requièrent le développement d'algorithmes, des méthodes et des protocoles qui sont en adéquation avec toutes ces propriétés qui caractérisent ces nœuds de capteurs sans fil.

TABLE 2.1 – Exemples de nœuds capteurs disponibles sur le marché [23].

Nœud Capteur	**Vitesse CPU (MHz)**	**PROM (KB)**	**RAM (KB)**	**Fréquence radio (MHz)**	**Taux de transmission (Kbps)**
mica	6	128	4	868	10/40
mica2	16	128	4	433/868/916	38.4 Kbaud
micaz	16	128	4	2.4 GHz	250
Sun SPOT	16-60	2 MB	256	2.4 GHz	250
BTnode	8	128	64	BT/433-915	Variable

Outre ces caractéristiques hardware, les programmes responsables de la mise en opération de ces nœuds adoptent des architectures évènementielles avec lesquelles le programme est défini par ses réactions aux différents évènements qui peuvent arriver [19]. Avec un nœud capteur, un évènement est déclenché suite à la disponibilité d'une mesure prélevée par l'un de ses capteurs, à la réception d'un message ou à l'écoulement d'un certain temps d'attente (i.e. un timeout). Le système d'exploitation qui est le plus accepté par la communauté scientifique est TinyOS [3]. Il a été développé au sein de l'université de Berkeley pour faire fonctionner ses nœuds capteurs mica. D'autres systèmes d'exploitation peuvent être cités comme LiteOS [24], Contiki [25] et Nut/OS [4].

2.4 Sources de dissipation d'énergie d'un nœud capteur

Comme nous venons de le voir, un nœud capteur est alimenté par une batterie qui le rend opérationnel sur une durée de temps bien limitée. En outre, cette durée doit être de l'ordre de plusieurs mois, voire des années, étant donné que dans la majorité des applications, le remplacement des batteries des nœuds n'est pas faisable à cause du nombre important de ces nœuds et également à cause des contraintes environnementales qui peuvent entraver l'accès à ces nœuds. En fait, la durée de vie du réseau, qui

3. http ://www.tinyos.net/
4. http ://www.ethernut.de/en/software/

représente la durée maximale jusqu'à laquelle ce réseau reste opérationnel, est fonction de la capacité de ces batteries. Dans ce sens, la consommation judicieuse de cette énergie représente le premier prérequis lors de la conception des algorithmes et des protocoles qui sont dédiés à ces réseaux.

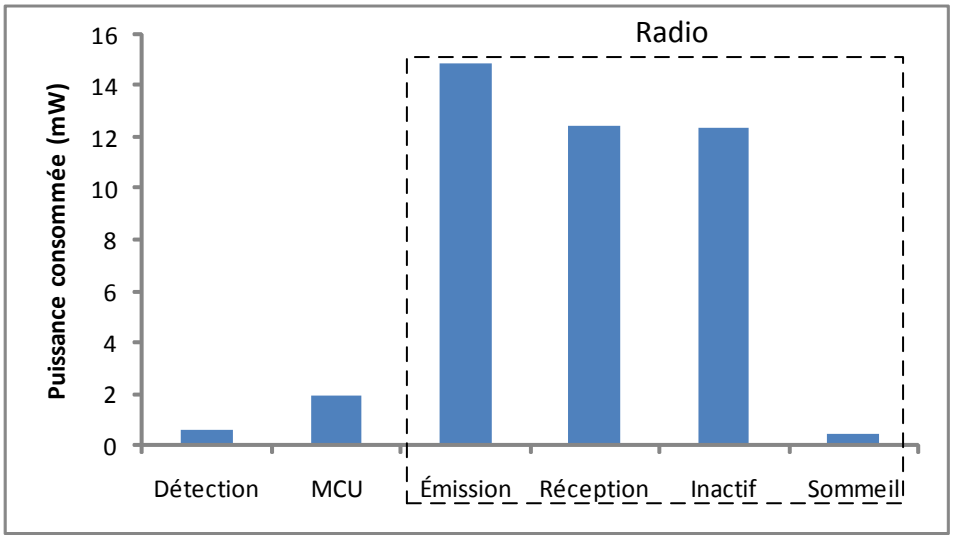


FIGURE 2.4 – La répartition de la dissipation d'énergie d'un nœud capteur [26].

Pour pouvoir concevoir de tels protocoles, il est indispensable de comprendre les sources de dissipation d'énergie d'un nœud capteur. Ce dernier est censé effectuer trois tâches lorsqu'il est déployé : la détection des évènements, le traitement des données prélevées et leur transmission. Suivant ces trois tâches, les sources de dissipation de l'énergie d'un nœud capteur consistent en :

- **Détection :** c'est généralement la source la moins consommatrice d'énergie. L'énergie consommée dépend des capteurs utilisés et la nature de cette détection (e.g. sporadique ou continue) [23]. Par exemple, avec des capteurs passifs comme des capteurs de température et de lumière, cette énergie peut être totalement ignorée par rapport aux autres sources de consommation. En revanche, cette énergie peut être importante dans le cas des capteurs actifs tels que les sonars [21].
- **Traitement des données :** de manière générale, un micro-contrôleur embarqué transite entre multiples états de fonctionnement suivant les tâches à effectuer. Trois états peuvent être communément distingués : actif, inactif et sommeil. L'énergie dissipée varie d'un état à un autre et également du temps passé dans chaque mode. En outre, la transition entre ces états consomme également de l'énergie [21]. Comme le montre la figure (2.4), l'énergie totale qui est consommée par le processeur est moins importante que celle dissipée par la radio. En fait, il est largement admis que l'énergie nécessaire pour transmettre un seul bit peut être utilisée pour effectuer un nombre important d'opérations arithmétiques [27].

Comme cité dans [28], l'énergie consommée pour transmettre un seul bit sur une distance de 100 m est équivalente à celle nécessaire à effecteur 3000 instructions.

- **Communication :** comme un micro-contrôleur, la radio opère dans de multiples modes de fonctionnements : transmission, réception, inactif et sommeil. L'énergie qui est consommée dépend d'un mode à un autre. De manière générale et comme l'illustre la figure (2.4), la radio consomme plus d'énergie lors d'une transmission. Une consommation moins importante lorsqu'il y a une réception des données ou lorsqu'elle est active sans transmission ou réception. Et à la fin, une consommation qui peut être négligeable lorsqu'elle est en mode sommeil. Outre toutes ces consommations, la transition entre ces modes requière également de l'énergie. Une constatation importante qu'il vaut la peine de la citer est que la radio consomme une quantité importante d'énergie lorsqu'elle est inactive même sans l'émission ou la réception des données. Cette énergie est même équivalente à celle dissipée lors de la réception d'un message [29].

Voici les principales sources de dissipation d'énergie d'un nœud capteur. Comme le montre la figure (2.4), la partie importante de l'énergie d'un nœud est consommée par la radio. Cette observation motive le besoin à des protocoles de communication économes en énergie qui utilisent judicieusement les radios des nœuds capteurs. Vu cette nécessité, il est recommandé d'éviter ou de limiter l'effet de l'un des phénomènes suivants qui consomment inutilement de l'énergie [29] :

- **Collisions :** le support de communication sans fil est partagé entre les différents nœuds capteurs. L'accès à ce support est assuré par la couche MAC. Généralement, deux messages émis à partir de deux nœuds distincts peuvent se heurter facilement sans qu'ils soient correctement reçus par leurs destinataires. À ce moment, une retransmission de ces messages est obligatoire ce qui consomme de l'énergie.
- **L'écoute à vide (idle listening) :** comme nous venons de le voir, un nœud capteur consomme une quantité importante d'énergie lorsque sa radio est active sans qu'elle émette ou reçoive des données. Dans ce cas, se basculer en mode sommeil est préférable.
- **L'écoute abusive (overhearing) :** lorsque deux nœuds capteurs communiquent ensemble, il est fort probable que d'autres nœuds reçoivent les données échangées entre ces deux nœuds sans qu'elles soient destinées à eux. Dans ce cas, ces nœuds peuvent perdre beaucoup d'énergie en recevant inutilement ces messages. Cette perte peut être très importante dans le cas où le trafic réseau est intense ou le réseau est dense.
- **L'overmitting :** un nœud capteur peut émettre ses données à un nœud destinataire qui n'est pas prêt à recevoir ces données.

- **La charge induite par les paquets de contrôle :** tout réseau doit être configuré afin qu'il puisse assurer sa fonction. Cette configuration nécessite généralement des échanges de messages de contrôle entre les différents nœuds du réseau. Étant donné que ces échanges consomment beaucoup d'énergie, il est important de mener cette configuration avec le moindre d'échanges de messages de contrôle.

2.5 Techniques de conservation d'énergie

Le long de ces deux dernières décennies, plusieurs techniques et schémas de conservation de l'énergie des nœuds capteurs ont été proposés. La majorité de ces techniques visent à proposer des techniques de communication économes en énergie étant donné que cette communication consomme la plus grande partie d'énergie. Dans cette section, nous présentons brièvement les principales techniques qui ont été proposées. Pour cela, nous reprenons ici la classification proposée dans [30] qui organise ces techniques en cinq grandes classes. Cette classification est schématisée par la figure (2.5). Dans ce qui suit, nous résumons chacune de ces classes.

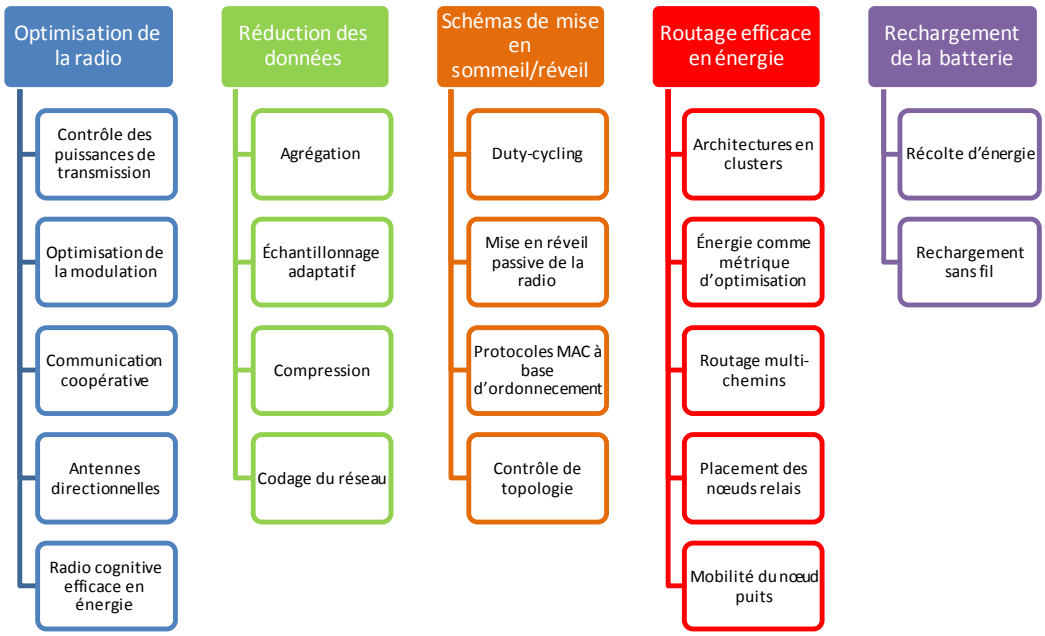


FIGURE 2.5 – Techniques de conservation d'énergie dans un réseau de capteurs sans fil [30].

2.5.1 Optimisation de la radio

Avec cette classe, le but consiste à optimiser le fonctionnement de la radio pour quelle consomme la moindre d'énergie. Cinq sous classes sont à distinguer.

La première vise à mettre en disposition des schémas de modulation économes en énergie [31]. De tels schémas essayent de réguler certains paramètres de modulation comme le temps de transmission et la valence du signal transmis. Avec cette sous classe, nous pouvons citer le travail [32] avec lequel ses auteurs étudient la relation entre le temps de transmission et l'énergie totale qui est consommée par la radio. Généralement, l'énergie consommée par un émetteur-récepteur radio intègre celle de transmission qui dépend de la distance de transmission et celle des circuits qui est indépendante de cette distance. Dans ce dernier travail cité, il a été démontré que la minimisation de l'énergie consommée lors d'une transmission implique la maximisation du temps de transmission dans le cas où la distance à laquelle on veut transmettre est importante. Cependant, cette constatation n'est pas vraie dans le cas où cette distance est courte étant donné que, dans ce cas, l'énergie consommée par les circuits n'est pas négligeable par rapport à celle qui est consommée lors d'une transmission. Dans ce cas, il est nécessaire d'optimiser ce temps afin de minimiser l'énergie totale qui est consommée par la radio.

La deuxième sous-classe concerne l'ensemble des techniques qui exploitent la technique de la communication coopérative. Avec une telle communication, la capacité des autres nœuds à entendre les messages de leurs voisins est exploitée en les autorisant à retransmettre ces messages. Le but ici consiste à pouvoir mettre en œuvre une diversité de transmission spatiale en mettant en coopération les antennes individuelles des divers nœuds qui sont en voisinage l'un par rapport à l'autre [33]. Comme conséquence à cette diversité, la qualité de réception de ces messages est améliorée en minimisant l'effet du phénomène d'évanouissement par propagation multitrajet [34]. Cette bonne réception des messages contribue à la minimisation du nombre de retransmissions qui sont dues à leur mauvaise réception. Cela contribue à son tour à la minimisation de l'énergie qui est consommée à cause de ces retransmissions [31].

Une autre technique qui est pertinente à cette classe d'optimisation consiste à tirer profit des performances accrues que l'on peut atteindre via l'utilisation des antennes directionnelles. En fait, avec ce genre d'antennes et par rapport à celles omnidirectionnelles, peu d'interférences et donc peu de collisions sont constatées. Des portées de communication plus longues pour une même puissance de transmission et des liens de communication plus stables sont assurés étant donné que l'énergie électromagnétique est concentrée vers une seule direction. Cela contribue à l'augmentation de la capacité du réseau comme beaucoup de transmissions peuvent être menées simultanément [27]. Toutes ces propriétés, et en particulier, la probabilité minime d'occurrence des collisions et l'écoute peu abusive des communications des autres nœuds permettent à minimiser l'énergie totale consommée [35].

Outre toutes ces techniques, doter les nœuds capteurs de radios cognitives ou intelligentes est censé améliorer l'efficacité des réseaux de capteurs sans fil [36]. Cette

notion de radio cognitive à été initialement proposée par Mitola et Maguire dans [37] où ils proposent de doter les équipements radio statiques d'une forme d'intelligence logicielle qui leur permet de s'adapter aux conditions de leur environnement en adaptant certains de leur paramètres comme la modulation, les puissances de transmission et les fréquences utilisées. Sur le plan énergétique, la capacité d'un nœud doté d'une radio cognitive à changer ses paramètres opérationnels selon les circonstances du canal de communication permet d'atténuer les effets qui sont dus aux collisions et donc aux retransmissions des messages [36]. Cependant, il est important de noter que le fait de doter une radio de capacités cognitives augmente sa consommation en énergie à cause des fonctionnalités complexes et sophistiquées d'une telle radio [38]. Dans ce sens, la conception d'une radio cognitive efficace en énergie est un défi scientifique que la communauté scientifique essaye de faire face.

La dernière sous-classe qui s'intéresse à l'optimisation de la radio et dont nos contributions sont pertinentes concerne l'ensemble d'approches qui contrôlent les puissances de transmission des nœuds. En fait, cette technique présente les avantages suivants [39, 40] :

1. **la conservation de l'énergie :** le contrôle des puissances de transmission des nœuds est utilisé afin d'optimiser les coûts dus aux communications. En fait, il a été démontré qu'une communication entre une paire de nœuds source-destination qui est menée via un chemin à sauts courts et multiples est plus énergétiquement efficace par rapport à une autre qui est menée via un chemin à longs sauts [39]. Dans ce sens, la réduction des puissances de transmission des nœuds contribue à la favorisation de ce genre de chemins, et par conséquent, contribue à économiser l'énergie des nœuds.
2. **Évitement des collisions :** comme un effet collatéral, la réduction des puissances de transmission des nœuds contribue à la réduction du nombre de collisions. Cela diminue à son tour le nombre de retransmission au niveau de la couche MAC, ce qui aide à minimiser davantage les coûts qui sont dus aux communications.
3. **L'augmentation de la capacité du réseau :** l'utilisation d'un support partagé de communication dans les réseaux sans fil nécessite une attention particulière afin de prévenir les interférences potentielles entre nœuds concurrents, et donc, afin de réduire le nombre des communications qui s'annulent suite aux communications des autres nœuds. Dans ce sens, la diminution des puissances de transmission des nœuds augmente la capacité du réseau en augmentant le nombre de communications pouvant être réalisées simultanément.

Lorsque cette technique est adoptée il est recommandé de suivre les directives de conception suivantes durant la réalisation d'un processus d'ajustement des puissances

de transmission des nœuds [39, 40] :

- **Distribution :** Il est préférable de mener ce processus de manière distribuée en se basant seulement sur des données locales au lieu de le mener de manière centralisée qui requière une image globale du réseau.
- **Bidirectionnalité des liens :** La considération des liens bidirectionnels entre nœuds de la nouvelle topologie générée suite à ce processus est recommandée. Cette considération de ce genre de liens facilite le fonctionnement de certains protocoles de la couche MAC comme le IEEE 802.11 qui requière l'échange des messages RTS, CTS et ACK entre un nœud émetteur et un autre récepteur.
- **Simplicité :** Étant donné que les nœuds capteurs ont des ressources limitées, il est recommandé de minimiser la complexité algorithmique de ce processus afin qu'il puisse être pratiquement mené. De même, la charge engendrée suite au nombre des messages de contrôle nécessaires à son mise en œuvre ne doit pas l'emporter sur le gain que ce processus apporte.
- **Connexité** : Dans le cas de contrôle de topologie où tous les nœuds du réseau sont concernés par ce processus de contrôle des puissances de transmission, il est nécessaire de s'assurer de la connexité de la topologie réduite suite à ce processus. Il est même important que cette nouvelle topologie soit un t-spanner du graphe maximal qui modélise la topologie originale en termes des puissances totales des chemins. Un sous-graphe G' est dit un t-spanner du graphe G s'il existe un nombre réel t tel que la puissance totale du plus court chemin dans G' est au maximum t fois la puissance totale du plus court chemin dans G. Étant donné que cette nouvelle topologie réduite est exploitée pour le calcul des routes entres nœuds, l'assurance de cette propriété assure que ces routes soient presque minimales. En addition, il est préférable que cette nouvelle topologie soit creuse (i.e. ayant peu d'arêtes) afin qu'elle soit bénéfique durant le processus de routage (i.e. la minimisation de la charge engendrée par les messages de contrôle nécessaires à l'établissement des routes et à leur maintenance).

2.5.2 Réduction des données

Avec cette classe d'approches, l'objectif consiste à réduire les coûts qui sont dus aux communications en minimisant la quantité des données qui circulent dans le réseau. Quatre approches peuvent être citées.

La première, qui est celle d'agrégation, autorise les nœuds du réseau à agréger les données reçues le long de leurs chemins vers le nœud puits. En exploitant la corrélation spatiale et temporelle qui caractérise ces données prélevées, un gain important en énergie peut être achevé [41]. Nous reviendrons, en détail, à cette technique d'agrégation des données au chapitre 3.

La deuxième technique consiste à optimiser le processus de prélèvement des données afin d'éviter tout prélèvement inutile qui peut affecter les ressources de communication et de calcul. Cette optimisation est achevée via l'adaptation des taux de prélèvement des nœuds capteurs, et donc le nombre de fois qu'un nœud capteur est censé prélever des données dans un intervalle de temps donné et à quel moment, tout en assurant que les objectifs réseau soient atteints avec une utilisation minimale de l'énergie [42, 43].

Une autre technique qui est similaire à celle de l'agrégation est celle de la compression. Avec cette technique, les données sont codées de manière à ce que leurs tailles soient diminuées. Cela contribue à la minimisation de l'énergie qui est consommée durant leur acheminement étant donné que l'énergie qui est consommée est fonction de la taille des paquets à transmettre.

La dernière technique connexe à cette classe est celle du codage réseau. L'idée de base consiste à ce que les nœuds intermédiaires acheminent une combinaison des paquets entrants, via leur encodage, au lieu de les router, un par un, individuellement avec la possibilité de reconstruire les données originales à leur réception [44].

2.5.3 Schémas de mise en sommeil/réveil

Comme nous avons vu dans la section (2.4), la radio consomme une quantité importante d'énergie lorsqu'elle est dans l'état inactif. L'ensemble d'approches de cette classe visent à adapter l'activité des nœuds de manière à pouvoir les mettre en sommeil et donc à pouvoir conserver de l'énergie.

La mise en réveil cyclique (duty cycling) est le nom qui est communément affecté à ces approches. L'idée consiste à mettre en sommeil les nœuds qui ne connaissent aucune activité. Ces nœuds sont ensuite mis en réveil seulement s'il est nécessaire de recevoir ou d'envoyer des données. Se procéder de cette façon évite l'écoute inutile des messages des autres nœuds, ce qui aide à minimiser l'énergie dissipée suite à cette écoute. La difficulté qui entrave la conception d'un tel processus est le fait qu'avec la plupart des applications, il est difficile de savoir quand les données sont censées arriver aux nœuds et donc de savoir quand il faut mettre les nœuds en réveil. En outre, cette technique de mise en réveil cyclique peut aggraver le temps de latence des messages qui sont transmis vers le nœud puits à cause de l'attente potentielle du réveil d'un nœud qui est supposé router ces messages. Par ailleurs, cette technique peut perdre son avantage si le coût nécessaire à sa mise en œuvre est plus important à celui gagné après sa mise en marche [45].

Cette technique s'accompagne généralement avec une autre technique de contrôle de topologie qui exploite la redondance qui caractérise le réseau où certains nœuds qui assurent la connectivité du réseau et la couverture des points d'intérêt sont les seuls qui sont mis en réveil et donc forment une colonne vertébrale du réseau. Les autres

nœuds sont simplement mis en sommeil [46, 47].

Même si cette technique de mise en réveil cyclique économise de l'énergie, la probabilité pour qu'un nœud capteur donné écoute inutilement les messages des autres n'est pas nulle. Pour pouvoir économiser davantage de l'énergie, d'autres approches dotent les nœuds capteurs, en plus de leurs radios principales, d'autres radios réceptionnistes ayant comme principale tâche la mise en réveil de ces nœuds. Ces derniers sont mis en réveil, à la demande, par d'autres expéditeurs. Ces radios de mise en réveil peuvent être de type actif où une source énergétique permanente est nécessaire. En contre-partie, elles peuvent être passives où elles récoltent de l'énergie à partir du signal de mise en réveil qui vient de l'émetteur. Un exemple de radio de mise en réveil passive et le tag RFID qui s'alimente à partir du signal de mise en réveil qui est émis d'un lecteur RFID [48].

2.5.4 Routage efficace en énergie

La définition des routes appropriées pour la délivrance des données au nœud puits est une tâche importante dans un réseau de capteurs sans fil. Si ces routes ne sont pas choisies convenablement alors une consommation énergétique importante peut être atteinte. Dans ce qui suit, l'ensemble des schémas de routage avec les différentes techniques qui sont utilisées pour économiser de l'énergie sont présentés.

Un premier schéma qui est couramment utilisé est celui de clusterisation. Avec un tel schéma, une hiérarchisation est imposée dans le réseau. L'ensemble des nœuds de ce dernier s'organisent en clusters. Pour chacun, un cluster-head est élu qui coordonne les activités de ses membres et joue le rôle d'interface entre ces derniers et le nœud puits. Le gain en énergie qu'apporte cette technique vient des potentialités qu'offre cette organisation en clusters. Principalement, un schéma de mise en sommeil/réveil peut être mis en place au sein de chaque cluster par chaque cluster-head. Ce dernier peut agréger et fusionner les données de ses membres pour envoyer un seul paquet et par conséquent le nombre de transmissions est diminué. En outre, un équilibrage de charge peut être appliqué via la rotation de ce rôle de leader entre les différents nœuds [49, 50].

Outre cette clusterisation, il est commun de considérer l'énergie comme une métrique pour la définition des routes entre les nœuds capteurs et le nœud puits. Dans ce sens, l'énergie résiduelle des nœuds peut être considérée afin de favoriser les nœuds qui ont une source énergétique abondante pour qu'ils acheminent les données vers le nœud puits. Par ailleurs, les longs sauts peuvent être évités pour choisir, à la place, les voisins qui ne sont pas à de longues distances, et donc, qui requièrent des petites puissances de transmission. Pour plus d'informations concernant les approches qui adoptent une telle stratégie, le lecteur peut se référencer à [51, 52].

De manière générale, les données sont routées le long d'un seul chemin entre un nœud capteur source et le nœud puits. L'acheminement répété qu'effectue les nœuds de ce chemin accélère la dissipation de leurs énergies. Comme conséquence, la durée de vie du réseau est limitée. Pour pallier à ce problème, d'autres approches mènent ce routage à travers de multiple chemins qui séparent une même paire de nœuds source-destination. Avec cette stratégie, l'énergie consommée par l'ensemble des nœuds du réseau est équilibrée étant donné que la charge n'est pas exercée sur un sous-ensemble de nœuds, mais par contre, elle est distribuée sur l'ensemble des nœuds [53].

Une autre technique qui est également utilisée consiste à introduire certains nœuds spécialisés, appelés relais, qui ont des sources énergétiques plus importantes [54]. Ces nœuds peuvent, par exemple, être choisis comme cluster-heads. Cependant, le nombre de relais et leurs placements doivent être définis de façon étudiée de manière à ce que ce nombre soit minimal, à cause de leurs coûts monétaires accrus, et le réseau soit connecté [55].

Une dernière technique qui est également utilisée pour la collection des données consiste à déployer des nœuds puits mobiles qui se déplacent entre les nœuds capteurs et récupèrent leurs données. Cette mobilité du nœud puits décharge complètement ou partiellement les nœuds capteurs de la tâche de relayage des données en permettant à ces nœuds de les communiquer directement ou le long d'un nombre minimal de sauts au nœud puits. De cette façon, le nombre de transmissions qui sont effectuées dans le réseau est largement minimisé. Par ailleurs et par rapport au cas d'un nœud puits statique où les nœuds capteurs, qui sont autour de ce nœud puits, connaissent une consommation énergétique importante par rapport aux autres nœuds, cette consommation est équilibrée sur l'ensemble des nœuds du réseau. Cela contribue à prolonger davantage la durée de vie du réseau [47]. Outre cette efficacité énergétique, cette stratégie est moins exigeante, en termes de connectivité, et assure une fiabilité de transmission accrue, car, les communications sont faites, dans la majorité des cas, à un-saut.

2.5.5 Rechargement de la batterie

Avec cette classe d'approches, le but consiste à recharger les batteries des nœuds capteurs sans aucune intervention humaine. Ce rechargement peut se faire via la récolte de cette énergie à partir de l'environnement immédiat des nœuds ou via un rechargement sans fil.

Un mécanisme de récolte d'énergie exploite diverses sources d'énergie qui caractérisent l'environnement de déploiement des nœuds afin de recharger les batteries de ces nœuds, et donc, afin de leur fournir une alimentation ininterrompue en énergie. Diverses sources d'énergie peuvent être considérées comme l'énergie solaire, l'énergie thermique, l'énergie hydraulique et l'énergie éolienne [56]. Avec une architecture de ré-

colte d'énergie, les nœuds capteurs doivent avoir généralement une idée sur l'évolution de l'énergie de leurs batteries afin qu'ils puissent adapter leurs comportements suite à cette évolution. Des opportunités de récolte d'énergie peuvent ne pas apparaitre continuellement. Pour cela, il reste toujours important de savoir économiser l'énergie des nœuds capteurs via l'implémentation de protocoles économes en énergie [30].

Avec un rechargement sans fil, les divers nœuds du réseau ont la possibilité d'échanger de l'énergie entre eux sans qu'ils soient en contact direct [30].

2.6 Défis de conception

Diverses contraintes entravent la conception et le développement de protocoles et d'applications qui sont dédiés pour les réseaux de capteurs sans fil. La liste suivante résume ces principales contraintes [19] :

- **Ressources énergétique :** comme nous l'avons déjà vu, un nœud capteur est alimenté par une batterie qui doit assurer son fonctionnement pour une longue durée de temps. Cela impose l'utilisation judicieuse de cette énergie.
- **Ressources de traitement :** tous les calculs d'un nœud capteur se font via un micro-contrôleur de capacité limitée. Pour cela, il est important de veiller à ce que la complexité algorithmique des programmes fonctionnant sur ces nœuds ne soit pas importante.
- **Ressources de stockage :** Outre les ressources énergétiques et de calcul limitées d'un nœud capteur, il dispose généralement d'une mémoire à taille réduite qui doit être suffisante au fonctionnement des divers programmes.
- **Bande passante et débit :** un émetteur-récepteur radio qui vient avec un nœud capteur est généralement optimisé pour que sa consommation énergétique soit minimale. Comme conséquence, la bande passante qui est mise à la disposition des applications est limitée.
- **Fiabilité :** il est important d'assurer un certain niveau de fiabilité en ce qui concerne les communications entreprises dans le réseau. Par ailleurs, la tolérance à toute panne potentielle est également requise.
- **Déploiement :** la façon dont les nœuds capteurs sont déployés conditionnent généralement les protocoles qui sont supposés fonctionner sur ces nœuds. Ce déploiement dépend de l'application et de l'environnement dans lequel cette application est censée fonctionner. Suivant cet environnement, ce déploiement peut se faire de manière aléatoire si cet environnement est inaccessible. Dans ce cas, il est nécessaire que le réseau se débrouille avec la distribution résultante et forme les connexions entre ses constituants.

- **Mobilité :** suivant l'application, un certain degré de mobilité peut caractériser le réseau. Cette mobilité peut être spatiale via la mobilité de certain des nœuds du réseau. Elle peut également être temporelle dans le cas où une mise en réveil cyclique des nœuds est appliquée.
- **Adressage :** de manière générale, les nœuds capteurs arrivent sans adresses d'identification. Cette tâche d'adressage doit se faire par des mécanismes réseaux spécialisés. Sinon, les protocoles et algorithmes réseaux doivent être indépendants de tout adressage.
- **Évolutivité :** un réseau de capteurs sans fil comporte généralement un nombre important de nœuds. Dans ce cas, les algorithmes et les protocoles doivent être conçus, de manière à ce qu'ils puissent fonctionner correctement, même dans la présence d'un grand nombre de nœuds.
- **Coûts monétaires :** le coût d'un réseau de capteurs sans fil est fonction du nombre de ses nœuds. Ce coût important entrave la mise en place de tout projet qui vise la prolifération de ces réseaux.
- **Conditions environnementales :** comme nous venons de le voir, un réseau de capteurs sans fil peut être déployé dans un environnement qui peut limiter l'accès aux nœuds de ce réseau. Cela nécessite des capacités d'auto-organisation qui permettent au réseau d'assurer son fonctionnement.
- **Miniaturisation :** la taille et la forme des nœuds capteurs peuvent contraindre l'applicabilité des réseaux de capteurs sans fil à certaines applications.

2.7 Domaines d'application

Les réseaux de capteurs sans fil envisagent un large éventail d'applications. En fait, les avancées spectaculaires, concernant leur miniaturisation et leur intégration dans les divers objets de notre vie quotidienne, ont favorisé davantage cette applicabilité. Dans cette section, nous présentons brièvement leurs principaux domaines d'application. Pour cela, nous reprenons la taxonomie qui a été présentée dans [30]. Elle regroupe cinq classes qui sont présentées dans ce qui suit.

2.7.1 Soins de santé

Les applications de cette catégorie comprennent, entre autres, la télésurveillance des données physiologiques humaines, le suivi et la surveillance des patients et des médecins au sein des hôpitaux. Par exemple, les patients peuvent porter des capteurs qui supervisent leurs paramètres vitaux afin d'identifier toute situation d'urgence pour permettre, à la suite, aux soignants de réagir efficacement [57]. Par ailleurs, autres

applications visent à faire assister et faire surveiller des personnes dépendantes et âgées dans leur vie quotidienne. Nous parlons ici du maintien à domicile des personnes ayant des handicaps moteurs, visuels, auditifs, cognitifs ainsi que des personnes âgées [58, 59].

Ces applications sont de nature critique, depuis que la surveillance des paramètres vitaux de l'humain se font automatiquement. L'hétérogénéité est considérée comme un problème élémentaire car, des dispositifs distincts entre en jeux. Ainsi, la localisation est également importante, puisque dans des moments critiques, aucune faute ou non capacité de localisation d'une personne malade ne doit être tolérée. Cependant, l'intégrité des données doit être assurée, ce qui nécessite une fiable transmission de ces données [13]. Outre toutes ces exigences, réagir en temps opportun à toute situation d'urgence telles que les crises cardiaques ou les chutes soudaines, est très important. Dans ce cas, la délivrance des données des nœuds aux utilisateurs finaux doit se faire rapidement et en temps borné. Une autre exigence concerne la confidentialité des données qui circulent entre les différents usagers de l'application. En fait, les données des patients doivent être privées à tout utilisateur non concerné. La dernière exigence qu'il vaut la peine de la citer concerne la possibilité à assurer la continuité de service lorsque les usagers de l'application sont en mouvement. Pour plus d'exemples concernant cette classe d'applications, le lecteur peut se référencer à [60, 61].

2.7.2 Environnement et agriculture

Les applications environnementales présentent un cadre applicatif favorable pour les réseaux de capteurs sans fil. Beaucoup d'applications peuvent être citées. Il y a certaines qui s'intéressent à la surveillance de l'habitat des animaux comme ce qui a été le cas avec les trois projets bien connus Dusck Island [62, 63], ZebraNet [64] et PLANET [65]. D'autres s'intéressent à l'érosion des côtes [66], d'autres à la surveillance de la pollution de l'aire [67] et ainsi de suite. Outre ces applications, le domaine d'agriculture a également attiré beaucoup d'attention. Avec ce cadre applicatif, les nœuds capteurs sont dispersés dans le champ pour surveiller certains paramètres pertinents, tels que la température atmosphérique, l'humidité du sol, les heures d'ensoleillement et l'humidité des feuilles. Toutes ces informations collectées sont ensuite utilisées afin d'améliorer les produits finaux et donc maximiser le bénéfice que les agriculteurs peuvent avoir. Pour plus d'informations et d'exemples concernant ce domaine d'agriculture, le lecteur peut se référencer à [68].

Avec ce genre d'applications, les zones supervisées sont d'une vaste largeur et cette supervision peut durer des années. La localisation des évènements est l'une des principales issues pour cette catégorie. Encore, les inévitables défis de la nature comme les terrains difficiles et les changements atmosphériques, impliquent l'obligation d'utiliser des réseaux sans infrastructure ainsi que des systèmes très robustes. En outre, comme

les nœuds sont libres et sans surveillance, le système doit être d'une faible consommation d'énergie et tolérant aux pannes. Enfin, les réseaux déployés ont la tendance à comporter un grand nombre de nœuds qui peut aller jusqu'à mille nœuds suivant la superficie de la zone surveillée. Pour cette raison, avoir des protocoles qui fonctionnent correctement avec ce grand nombre de nœuds est nécessaire.

2.7.3 Sécurité publique et systèmes militaires

Les réseaux de capteurs sans fil peuvent apporter une grande aide afin d'assurer la sécurité de l'humain lors des opérations critiques, d'urgences ou militaires. Les applications de cette classe peuvent être catégorisées en deux sous-classes : intervention active et supervision passive.

Avec une application à intervention active, les usagers du système portent avec eux des nœuds capteurs durant leurs activités. Ces nœuds permettent à d'autres superviseurs distants à être conscients des situations de ces usagers et donc à assurer leur sécurité et également à avoir une idée sur leur environnement. Ces applications sont applicables généralement aux travaux qui présentent des dangers potentiels à leurs pratiquants comme les secouristes [69], les soldats [70] et les mineurs [71].

Pour les applications à supervision passive, des nœuds capteurs statiques sont déployés dans une région d'intérêt pour une longue supervision. Plusieurs applications peuvent être citées. Par exemple, ces nœuds peuvent être déployés pour suivre certaines cibles [72] ou pour donner aux secouristes une idée cohérente du l'état du champs d'intervention (e.g. la distribution de la température) [73].

Les applications de ce domaine sont caractérisées par leurs propres exigences. Premièrement, la délivrance en temps opportun des données est très importante, car, dans des situations d'urgence, toute intervention qui se fait en retard risque de coûter des vies. En addition, les messages d'urgence doivent être acheminés, en priorité, par rapport à d'autres données régulières. Dans ce cas, un mécanisme de différenciation de services est indispensable. Outre ces deux exigences, l'assurance de l'intégrité des données est élémentaire afin d'éviter la délivrance de toute donnée corrompue qui peut mettre, en erreur, les usagers de l'application durant leurs missions. Par ailleurs, ces usagers dans le cas des applications, à intervention active, sont en perpétuel mouvement. Par conséquent, ces applications doivent supporter la mobilité des objets et dispositifs déployés. En outre, les applications de sureté publique et militaire sont généralement utilisées dans des environnements hostiles. Pour cela, il est important que le réseau soit résistant à toute panne potentielle et à des liens de communication de faible qualité via la mise en place de mécanismes de tolérance aux pannes appropriées.

2.7.4 Industrie

Les réseaux de capteurs sans fil sont largement utilisés en industrie. Le but consiste à automatiser tout système de surveillance et de contrôle en éliminant l'obligation de la présence de l'être humain dans des places qui peuvent lui coûter la vie. Par ailleurs, l'utilisation de la technologie sans fil à la place des câbles contribue à la minimisation des coûts monétaires qui sont dus à l'installation de ces câbles et des isolants qui les protègent [74]. Ces réseaux de capteurs sans fil peuvent être utilisés afin de surveiller l'état des équipements et l'identification de ceux qui ne fonctionnent pas correctement, d'assurer le bon déroulement des processus de fabrication, d'automatiser l'acquisition des données à partir des capteurs distants, et donc, à minimiser l'intervention humaine, d'assurer la sécurité des employées, la détection de toute anomalie comme les fuites de liquide / gaz, etc.

Les applications industrielles requièrent généralement une délivrance des données qui soient réalisée dans des délais bornés. En fait, à cause de la nature des produits qui sont traités en industrie et qui peuvent être très dangereux, la détection précoce de toute anomalie durant leur utilisation peut prévenir l'occurrence de tout accident potentiel. Encore, ces réseaux sont subis à diverses perturbations qui peuvent endommager leurs nœuds et entraver leur fonctionnement à cause des conditions hostiles dans lesquelles les nœuds capteurs opèrent. Cela nécessite des réseaux robustes avec des protocoles de communication tolérants aux pannes. Par ailleurs, ces réseaux doivent avoir un coût monétaire minimal et être constitués de nœuds capteurs compacts. Pour plus de détails concernant ces exigences connexes à ces applications industrielles, le lecteur peut se référencer à [74].

2.7.5 Systèmes de transport

Le but de ce type d'applications est d'offrir aux utilisateurs des conditions de transport sûres et confortables. L'intégration des réseaux de capteurs sans fil avec ces systèmes de transport assure la disponibilité, en temps réel, de précieuses données qui sont mises à la disposition de divers services gouvernementaux ou commerciaux. En fait, le déploiement des nœuds capteurs au niveau des routes et des intersections permet d'avoir une idée sur le trafic routier. Cette information peut être utilisée ensuite afin d'ajuster les signaux qui régulent ce trafic ou le nombre de postes de péage et de voies ouvertes [75], de guider les conducteurs durant leur trajet pour éviter les embouteillages [76], etc. Les nœuds capteurs peuvent également être déployés au niveau des parkings afin d'informer les conducteurs de toute place de stationnement libre [77]. Pour plus d'exemples, le lecteur peut se référencer à [78].

La communication ad hoc, la mobilité et la localisation sont des issues clés pour ces applications. Ainsi, des systèmes comme celui du contrôle du trafic routier requièrent

des informations, en temps réel, afin de pouvoir gérer efficacement la circulation des véhicules. En addition, avoir un support pour la différentiation de services est très important. En fait, le canal réseau est partagé entre diverses applications de différentes natures comme celles qui contrôle le trafic routier, d'autres qui assurent la sureté routière, etc. Dans ce cas, les informations critiques et de contrôle du trafic doivent être acheminées en priorité par rapport à d'autres informations qui sont liées à d'autres services. Une autre exigence concerne l'intégrité des données et la sécurité du réseau de communication. Le réseau doit être protégé contre toute tentative de piratage qui peut corrompre les données pour falsifier les informations qui concernent le trafic ou l'état des routes.

2.8 Conclusion

Dans ce chapitre introductif, nous nous sommes intéressés au domaine des objets coopératifs, et en particulier, à celui des réseaux de capteurs sans fil qui représente un exemple typique de ce domaine. Nous avons présenté les principales notions connexes à ces réseaux ; Les diverses sources de dissipation d'énergie d'un nœud capteur ; Les différentes techniques de conservation de cette énergie ; les contraintes qui contraignent le développement de solutions de communication pour ces réseaux et leurs domaines d'application. L'une des techniques de conservation d'énergie que nous avons citée et qui est pertinente à la classe d'approches qui visent à réduire la quantité des données qui circulent dans le réseau est celle d'agrégation. Cette dernière technique présente plusieurs avantages et exploite la corrélation spatiale et temporelle qui peut caractériser les données prélevées par les nœuds capteurs afin d'économiser l'énergie des nœuds. Afin d'illustrer le bénéfice qu'apporte cette stratégie, nous consacrons le suivant chapitre à cette technique en présentant les principales notions qui y sont liées et les travaux de la littérature qui y sont connexes.

Chapitre 3

Agrégation des données

3.1 Introduction

L'agrégation des données est considérée comme l'une des principales techniques pouvant être exploitées afin d'optimiser la consommation énergétique dans un réseau de capteurs sans fil. En exploitant les corrélations qui peuvent caractériser les données prélevées par les nœuds capteurs, cette technique d'agrégation vise à éliminer toute redondance potentielle, entre données, afin de minimiser la taille et le nombre de messages échangés dans le réseau. Cela contribue à minimiser l'énergie qui est dissipée par ces nœuds durant la collection des données, étant donné, que la partie importante de cette énergie est consommée durant les communications. Dans ce chapitre, nous exposons les éléments clés de cette technique avec une exploration détaillée des travaux connexes à cette dernière. Nous nous focalisons particulièrement sur les structures de routage qui sont exploitées pour mener l'agrégation des données et sur la façon dont elles sont créées. Nous commençons par une première section introductive afin d'illustrer et motiver cette technique d'agrégation. Nous présentons ensuite quelques exemples applicatifs ainsi que l'impact de cette technique sur les performances du réseau. Les composants essentiels d'un protocole d'agrégation sont ensuite exposés. Le reste du chapitre se focalise sur les structures de routage et la façon dont elles sont créées tout en détaillant différents travaux qui s'intéressent à cette construction. D'autres approches qui s'attaquent, en plus à ce problème d'agrégation, au problème de la délivrance en temps réel des données prélevées sont aussi exposées dans une section à part.

3.2 Motivation et Principe

Les WSNs sont conçus principalement pour surveiller certains phénomènes environnementaux via le déploiement d'un nombre important de nœuds capteurs dispersés à travers une certaine zone géographique. Comme nous l'avons déjà vu, ces nœuds cap-

teurs sont caractérisés par des ressources limitées, particulièrement, en termes d'énergie. Par conséquent, toute solution de conception qui est dédiée pour ces réseaux doit tenir compte de cette particularité. Étant donné que la partie importante de l'énergie d'un nœud capteur est consommée lors des communications (i.e. transmission et réception des messages) lorsqu'on la compare avec celle qui est consommée lors des calculs. Il est donc plus rationnel de privilégier les calculs aux communications.

D'un autre côté, les phénomènes qui sont observés par les nœuds capteurs sont corrélés dans l'espace et dans le temps. En addition, ces nœuds sont généralement déployés avec une haute densité. Par conséquent, les données qui sont émises par ces nœuds sont relativement corrélées. Deux nœuds capteurs peuvent même capter la même donnée s'ils sont très proches l'un de l'autre. En contre-partie et bien que leurs données émises soient différentes lorsqu'ils sont éloignés l'un de l'autre, ces données peuvent avoir le même type [23].

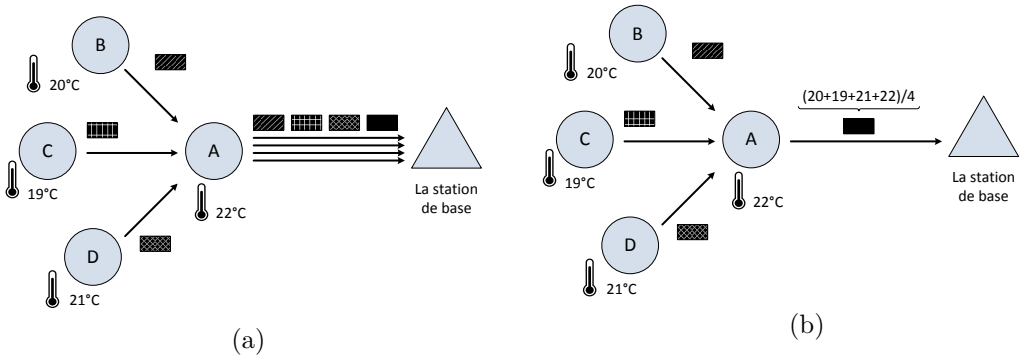


FIGURE 3.1 – Exemple de calcul de la température moyenne.(3.1a) Aucune agrégation de données avec un total de 7 transmissions ; (3.1b) Avec agrégation de données avec un total de 4 transmissions.

En exploitant cette similarité entre données captées, les nœuds qui sont situés à des endroits proches peuvent agréger leurs données prélevées et envoyer un seul paquet pouvant résumer l'essentiel de ces données. En outre, les paquets qui proviennent de lieux éloignés peuvent être concaténer ensemble pour envoyer un seul paquet. Cela constitue le principe d'agrégation de données où l'idée de base consiste à fusionner les différentes données captées et éliminer toute redondance potentielle, en exploitant la corrélation qui peut exister entre elles, lorsqu'elles sont acheminées vers le nœud puits au lieu d'envoyer individuellement chacune d'elles. Cela aide à diminuer le nombre de messages échangés entre les nœuds du réseau, ce qui permet à minimiser l'énergie consommée par ces nœuds [79]. Ce principe est illustré par la figure (3.1) où le nœud puits veut calculer la température moyenne dans une région donnée. Une première solution consiste à recevoir toutes les valeurs à partir des nœuds capteurs A, B, C et D pour calculer ensuite cette moyenne. Cela génère un total de 7 transmissions. Au lieu de procéder de cette façon et comme deuxième solution, il est plus optimal de

calculer cette moyenne au sein du réseau lorsque les données sont acheminées vers le nœud puits. Avec cette solution, le nœud A récupère les valeurs de température de ses voisins B, C et D, calcule la moyenne en utilisant aussi sa valeur captée et envoie le résultat au nœud puits. Cela permet de diminuer le nombre de transmissions total à un nombre de 4 au lieu de 7.

Dans cette thèse, nous adoptons la définition proposée dans [80] et nous considérons le processus d'agrégation de données comme étant : *le processus global de la collection et du routage des données prélevées à travers un réseau à sauts multiples, et également du traitement de ces données au niveau des nœuds intermédiaires avec l'ultime objectif de minimisation des différentes ressources du réseau, particulièrement les ressources énergétiques, et par conséquent, l'augmentation de la durée du vie du réseau.*

Par rapport au nœud puits, l'agrégation des données contribue considérablement à la minimisation de la charge qui peut être exercée sur ce nœud dans le cas où tous les nœuds capteurs émettent leurs données vers lui [81]. Elle fait diminuer même la charge qui est exercée sur les capteurs qui sont autour de ce nœud puits, ce qui aide à éviter leur panne énergétique précoce, et par conséquent, contribue à l'augmentation de la durée de vie du réseau [23].

3.3 Exemples applicatifs

Afin de montrer cette importante notion d'agrégation de données, voici quelques exemples applicatifs montrant l'exploitation de cette technique.

Prenant l'exemple d'une application qui vise à surveiller les niveaux des rayonnements autour d'une installation (e.g. usine) nucléaire. Avec une telle application, les différents nœuds capteurs peuvent agréger leurs valeurs captées lorsqu'elles sont acheminer vers le nœud puits pour considérer seulement la valeur maximale perçue étant donné que cette valeur est cruciale pour la sureté de l'installation et de l'environnement qui l'entoure [82].

Les auteurs dans [83] utilisent cette technique d'agrégation de données afin d'optimiser les ressources énergétiques des nœuds capteurs contribuant à une opération d'inventaire. Ces auteurs visent la classe d'applications ayant comme objectif l'estimation de la population d'une certaine espèce ou certain type d'objets. Leur idée de base consiste à mener l'opération de sommation des nombres des objets perçus tout près des sources de données et envoyer ensuite des rapports agrégés accompagnés des informations de la région sous-jacente au lieu d'envoyer des rapports individuels pour chaque objet perçu.

Dans le domaine d'agriculture, les auteurs dans [84] évaluent un système d'arrosage automatique tirant profit de la technologie des WSNs. Ils ont comparé entre deux méthodes de collection de données de température et d'humidité du sol. Une

première méthode où les communications entre les nœuds capteurs et le nœud puits sont faites directement. Et une deuxième où les données sont agrégées au niveau de certains nœuds assumant des rôles de leadership en considérant seulement la moyenne des valeurs perçues. Les nouvelles données agrégées sont ensuite envoyées vers le nœud puits. Les résultats de leur évaluation montrent l'apport important de cette technique d'agrégation dans la conservation de l'énergie des nœuds capteurs.

Dans le cadre des applications visant à poursuivre des cibles d'intérêts, VigilNet [85] est l'une de ces applications qui a été réellement expérimentée. L'agrégation des données constitue l'un de ses mécanismes utilisés pour minimiser l'énergie qui est dissipée lors du suivi d'un évènement et le signalement de son occurrence au nœud puits. L'idée consiste à grouper l'ensemble des nœuds capteurs détectant l'occurrence d'un même évènement (e.g. la présence d'un véhicule) en un même groupe qui est ajusté suivant les mouvements de l'évènement surveillé. Un leadership qui représente ce groupe collecte les rapports provenant des autres membres pour en déduire la position de l'évènement et sa vitesse et envoyer un rapport sommaire ensuite au nœud puits. Les expérimentations conduites ont montré que cette agrégation contribue à diminuer le nombre des rapports individuels signaler au nœud puits, et donc à optimiser le nombre des messages échangés. Elles ont montré également que cette agrégation peut diminuer le nombre de fausses alertes si le degré [1] d'agrégation et bien ajusté.

Outre ces applications, l'agrégation des données peut également être exploitée afin de mettre en place des systèmes, économes en énergie, et qui visent à assurer une image globale de l'état des ressources des nœuds capteurs du réseau tel que leurs états énergétiques. eScan [86] est un exemple de tels systèmes. Son objectif principal consiste à schématiser la distribution de l'énergie résiduelle des différents nœuds du réseau. Ce système est très utile pour tout administrateur d'un réseau de capteurs sans fil. Il permet la localisation de tout nœud ayant un niveau énergétique critique afin d'intervenir au bon moment pour assurer la continuité du fonctionnement du réseau. eScan tire pleinement profit de la technique d'agrégation et l'applique afin de combiner des représentations locales au sein du réseau au lieu de les combiner centralement au niveau du nœud puits.

3.4 Impacts de l'agrégation des données

Dans ce qui suit, voici les principaux éléments qui sont influencés par l'agrégation des données [2, 87, 81, 19] :

1. **Efficacité énergétique :** comme nous l'avons déjà mentionné, l'agrégation des données a comme principal objectif l'optimisation de la consommation énergé-

1. le nombre de rapports individuels des nœuds membres agrégés au niveau du leadership.

tique des nœuds capteurs via la minimisation du nombre de messages échangés entre ces nœuds.

2. **Le temps de latence des données :** en fait, la présence simultanée de multiples messages au niveau d'un nœud capteur donné est indispensable afin que ce nœud puisse mener l'opération d'agrégation. Pour cette raison, ce nœud doit introduire un certain *temps d'attente* dans lequel il attend l'arrivée des données qui lui sont destinées avant de procéder à leur agrégation. Cependant, mettre les données en attente au niveau de chaque nœud agrégateur augmente considérablement leur temps de latence. Comme on peut le voir dans la figure 3.2, tirée de [19], si chaque nœud attend une durée Δt avant d'agréger et envoyer ses données, on va avoir un délai d'agrégation total de $n\Delta t$ pour n nœuds agrégateurs. Cela contraigne défavorablement les applications temps réel. À partir de là, savoir ajuster le temps d'attente de chaque nœud agrégateur est important afin d'assurer un compromis entre le degré d'agrégation et le temps de latence des données prélevées.

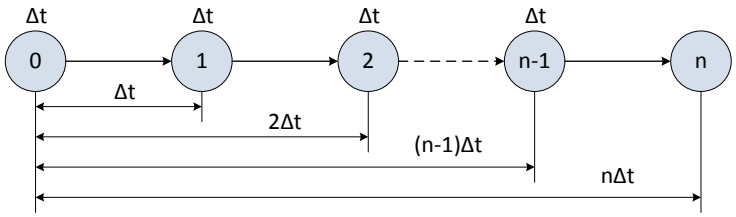


FIGURE 3.2 – Le délai d'agrégation total introduit par n nœuds agrégateurs [19].

3. **Le degré de collisions :** l'agrégation aide à minimiser l'occurrence des collisions étant donné qu'elle contribue à la minimisation du nombre de messages échangés dans le réseau.
4. **Précision des données :** même si l'agrégation aide à conserver l'énergie des nœuds, elle influence défavorablement la précision des résultats reçus par le nœud puits. Cette précision peut être mesurée en considérant le nombre de nœuds capteurs contribuant à ces résultats. Par exemple, la moyenne de température calculée en utilisant les mesures de deux capteurs est moins fiable par rapport à celle qui peut être calculée par les mesures de dix capteurs. De manière similaire au temps de latence des données, les temps d'attente attribués au différents nœuds capteurs ont également un impact sur cette précision.
5. **La fraicheur des données :** généralement, la collection des données à partir des nœuds capteurs se fait par rounds successifs. La fraicheur de ces données est mesurée en calculant la différence entre le round de génération de ces données et le round de leur réception. Cette fraicheur est assurée si les données générées en un même round sont agrégées ensemble. Suivant l'application, cette fraicheur

peut ou non pas être importante. Par exemple, la fraicheur n'a pas d'importance pour une application où l'objectif est de détecter seulement l'occurrence d'un incendie. Cependant, être constamment conscient des contours de cet incendie requiert l'assurance de la fraicheur des données prélevées. Cette fraicheur dépend grandement des temps d'attentes utilisés par les nœuds agrégateurs.

6. **La bande passante du réseau :** les nœuds capteurs sont connus pour leurs bandes passantes restreintes. Pour cela, la diminution du nombre de messages échangés via l'agrégation aide à optimiser l'utilisation de ces bandes passantes.

3.5 Composants d'un protocole d'agrégation de données

Certains éléments sont indispensables pour que toute agrégation de données peut être menée de manière efficace. Premièrement, les données captées, et lors de leur acheminement, doivent être dirigées de manière à pouvoir se rencontrer au niveau des mêmes nœuds afin qu'elles puissent être agrégées. Cela impose le besoin à des *schémas de routage* qui font converger spatialement ces données vers les mêmes nœuds qui seront considérés ensuite comme points d'agrégation. Deuxièmement, même si ces données peuvent passer par les mêmes nœuds capteurs, elles doivent y être présentent simultanément. Cette contrainte oblige chacun de ces nœuds d'attendre un certain temps d'attente avant de procéder à l'agrégation des données et le relayage des données résultantes vers le nœud puits. Donc, avoir *un plan* qui définit le temps d'attente de chaque nœud participant à l'opération de routage est indispensable pour réussir l'agrégation des données. Troisièmement, une fois que les données sont présentes simultanément au niveau des mêmes nœuds, elle sont agrégées suivant une certaine *fonction* qui définit comment cette agrégation est effectuée. Ces trois éléments : schémas de routage, la planification du processus d'agrégation et la fonction d'agrégation constituent les composants essentiels d'un tout protocole d'agrégation de données [2]. La majorité des travaux qui s'intéressent à la technique d'agrégation se focalisent généralement sur l'un de ces composants en assumant l'existence des deux autres. Pour notre travail qui est présenté dans cette thèse, et en ce qui concerne nos contributions, nous nous concentrons essentiellement sur les schémas de routage en supposant une planification et une fonction d'agrégation simples. Pour cette raison, nous décrivons brièvement dans cette section chacun de ces composants et nous détaillons ensuite, dans la section 3.6, le principe et certains travaux connexes aux schémas de routage.

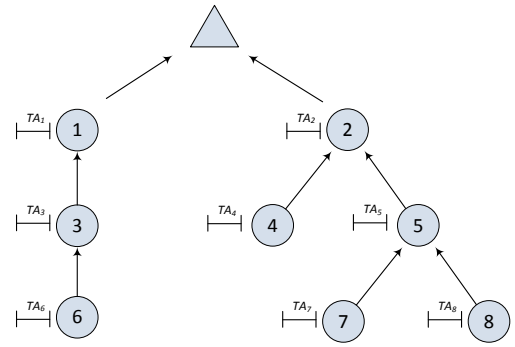
3.5.1 Schéma de routage

Ce composant définit la façon dont les données prélevées sont routées vers le nœud puits. Ce routage peut être fait avec ou sans la considération d'une structure réseau particulière. Dans le cas d'un schéma de routage structuré, une structure en arbre ou en clusters est généralement établie pour être ensuite exploitée durant l'agrégation des données. Cette construction peut se faire de manière proactive où elle est définie au préalable et maintenue périodiquement. Cette façon de faire est plus appropriée aux applications où la collection des données se fait de manière périodique. La construction peut se faire également de manière réactive afin de conserver l'énergie des nœuds capteurs durant les périodes d'inactivité du réseau [88]. Les applications cibles par ces schémas réactifs sont généralement des applications évènementielles. Un schéma de routage structuré assure un haut degré d'agrégation étant donné que la structure établie définit un certain ordre en ce qui concerne la manière de transmission des données [89]. Par exemple, si on prend l'exemple d'une structure en arbre, les données sont transmises des nœuds feuilles et remontent d'un nœud de l'arbre à un autre jusqu'à arriver au nœud racine. Cependant, se baser sur une structure pour mener les opérations de routage et d'agrégation requiert sa maintenance continue qui nécessite une charge importante dans la présence d'une grande dynamique dans le réseau. Pour cette raison, d'autres approches élaborent des schémas de routage sans la considération d'aucune structure particulière. L'idée de base pour ces travaux consiste à concevoir des mécanismes qui favorisent la convergence spatiale et temporelle des données prélevées.

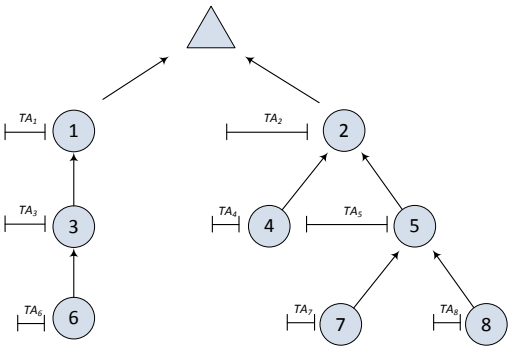
3.5.2 La planification du processus d'agrégation

De manière évidente, un nœud capteur ne peut effectuer aucune agrégation si les données qui arrivent de ses nœuds, en aval, ne sont pas présentent simultanément à son niveau. Pour cela, assigner un certain *temps d'attente* à chaque nœud capteur intermédiaire qui participe au relayage des données et à leur agrégation est indispensable afin d'assurer un haut degré d'agrégation. Ce temps d'attente définit la période du temps que doit attendre chaque nœud agrégateur avant d'agréger ses données reçues et envoyer les données résultantes à son nœud en amont. La définition de la durée de cette période est très cruciale et comme nous l'avons déjà vu dans la section 3.4, elle influence certains éléments dont leur importance dépend de l'application et le scénario considéré. Suivant la nature de ce temps d'attente, les schémas de planification du processus d'agrégation peuvent suivre deux approches [2] :

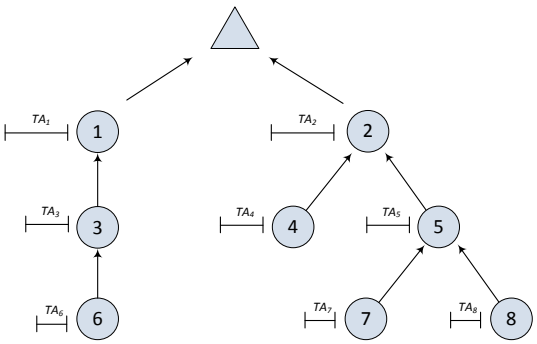
1. **Une planification avec temps d'attente non périodisé :** avec une telle planification, le temps d'attente qui est assigné à chaque nœud est continu et de longueur prédéterminée. Durant la période d'attente, le nœud attend les données qui arrivent des autres nœuds. Une fois cette période expire, il agrège et relaie



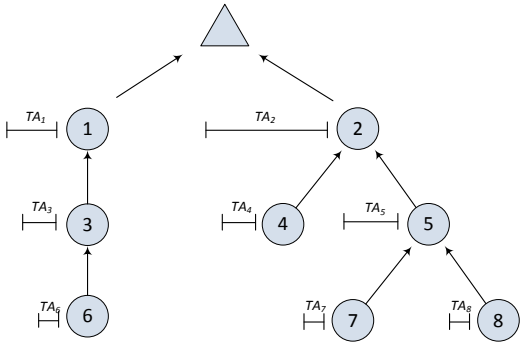
(a) Temps attente fixe.



(b) Temps d'attente ajusté selon le nombre des fils des noeuds.



(c) Temps d'attente ajusté selon la position des nœuds.



(d) Temps d'attente ajusté avec une considération conjointe de la position et le nombre des prédécesseurs des nœuds.

FIGURE 3.3 – Planification avec temps d'attente non périodisé. (*TA* correspond à Temps d'Attente)

les nouvelles données. Les approches se basant sur une telle planification sont employées généralement au niveau de la couche réseau en adoptant un protocole d'accès au canal à base de contention (e.g . solutions de la famille CSMA). Suivant la façon dont les différents intervalles du temps sont assignés aux différents nœuds capteurs, les approches dans cette catégorie peuvent davantage être classées en deux grandes classes :

(a) **Approches avec temps d'attente fixe :** les approches de cette catégorie assignent un même temps d'attente à tous les nœuds agrégateurs durant lequel ils attendent, agrègent toutes les données reçues et envoient les données agrégées vers leurs prochains sauts. La figure (3.3a) illustre une telle planification. Directed-diffusion [90] est un exemple d'approche adoptant cette stratégie. L'inconvénient avec de telles approches est que la fraicheur des données est grandement affectée étant donné qu'il y a une grande probabilité que des données qui sont générées à des rounds différents soient agrégées ensemble [81].

(b) **Approches avec temps d'attente ajusté :** par rapport à la première catégorie, les approches dans cette catégorie définissent les durées des périodes attribuées suivant certains facteurs. Principalement, trois facteurs peuvent être considérés [2]. Comme le montre la figure (3.3c), les temps d'attente peuvent être assignés suivant la position du nœud dans l'arbre d'agrégation où leurs durées sont attribuées de manière inversement proportionnelle au nombre de sauts séparant les nœuds capteurs du nœud puits. Cascading timers [81] est un schéma de planification qui se base sur ce principe. Ces temps d'attentes peuvent être également assignés selon le nombre de fils de chaque nœud comme ce qui est le cas de ATCA (Aggregation Time Control Algorithm) [91]. Dans ce cas et comme l'illustre la figure (3.3b), les nœuds ayant un même nombre de nœuds fils vont avoir un même temps d'attente. Dernièrement, une approche hybride peut être adoptée en attribuant ces temps d'attente selon la position et le nombre des prédécesseurs des nœuds comme le cas de ATC (Adaptif Timing Control) [92]. La figure (3.3d) illustre une telle stratégie. Les approches dans cette catégorie sont meilleures par rapport aux approches de la première catégorie. Cependant, elles présentent certains inconvénients qui diffèrent suivant le mécanisme employé. Par exemple, avec une planification qui se base sur la position des nœuds, deux nœuds ayant un même niveau dans l'arbre de routage peuvent émettre leurs données en même temps, ce qui mène à d'éventuelles collisions. En outre, un agrégateur ayant un nombre important de prédécesseurs peut relayer ses données avant de recevoir toutes les données qui arrivent de ses nœuds fils. Cela affecte la fraicheur des données. Cette fraicheur est également affectée avec une planification qui se base sur le nombre de nœuds fils. Ce cas arrive lorsqu'un agrégateur donné a moins de fils par rapport à ce que ses fils en possède.

2. **Une planification avec temps d'attente périodisé :** contrairement à la première approche, le temps d'attente qui est attribué à chaque nœud est cette fois-ci subdivisé en un ensemble de créneaux (i.e. time slots). Chaque nœud utilise des créneaux distincts, qui sont suffisamment longs, pour envoyer et recevoir les données. Par rapport à la première méthode de planification, l'objectif ici consiste à savoir distribuer les différents créneaux entre les nœuds du réseau. La figure (3.4) illustre cette façon à planifier. Comme on peut le constater, quatre créneaux sont employés. Chaque nœud utilise certains créneaux pour recevoir des données (i.e. créneaux rouges) et d'autres créneaux pour transmettre (i.e. créneaux bleus). Les créneaux non exploités (i.e. créneaux noirs) donnent l'opportunité à ce nœud d'éteindre son émetteur-récepteur afin de conserver son énergie. Par ailleurs, une approche adoptant une telle planification assume l'existence d'un

service tiers de synchronisation et utilise un protocole d'accès au canal basé sur un accès multiple à répartition dans le temps (i.e. basé TDMA). Les solutions dans cette classe considèrent une conception inter-couches avec laquelle la couche réseau et la couche MAC sont entrelacées. Ainsi, elles procèdent en exécutant les deux tâches suivantes : (*i*) la construction de la structure de routage (e.g. arbre) ; (*ii*) l'assignation des créneaux aux nœuds du réseau. Par rapport à une planification avec temps d'attente non périodisé, cette manière à planifier est plus bénéfique. Particulièrement, elle prévient l'occurrence des collisions étant donné que les nœuds sont synchronisés. Cela aide à minimiser le temps de latence et à conserver l'énergie des nœuds. Comme l'illustre la figure (3.4), les nœuds 7 et 8 utilisent différents créneaux pour envoyer leurs données afin de prévenir d'éventuelles collisions au niveau du nœud 5. Cette classe regroupe un nombre important d'approches. NCA (Nearly Constant Approximation for data aggregation scheduling) [93] et JSPC (Joint Scheduling and Power Control) [94] sont deux exemples d'approches adoptant une telle planification.

Pour plus de détails concernant les techniques de planification, le lecteur peut se référencer à [2].

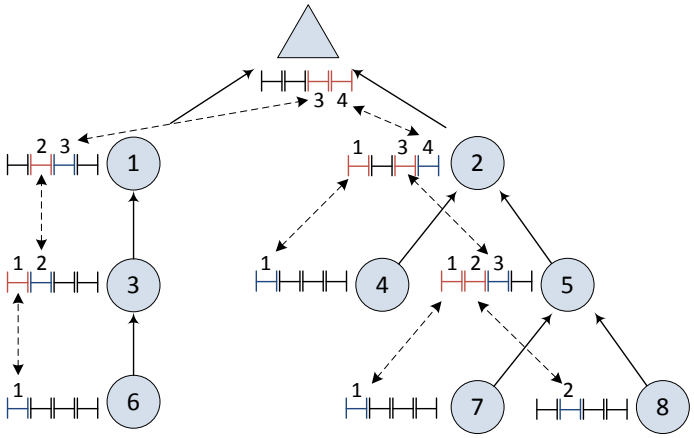


FIGURE 3.4 – Planification avec temps d'attente périodisé.

3.5.3 La fonction d'agrégation

Une fonction d'agrégation assure la combinaison des données brutes pour produire un résumé pouvant représenter ces données. Une telle fonction et son apport dans la réduction de la taille totale des données dépend principalement des propriétés des données et de leurs représentations ainsi que de la nature de l'application en main [95]. Néanmoins, certains critères peuvent être utilisés afin de les classer :

1. **Avec ou sans perte :** une fonction d'agrégation peut être conçue de manière à pouvoir restituer ou pas les données originales à partir des données agrégées.

Une fonction d'agrégation sans perte préserve toutes les données importantes et élimine toute redondance potentielle afin de diminuer la taille des données résultantes. Par exemple, prenons l'exemple d'un nœud capteur reçevant deux notifications pertinentes à deux évènements distincts qui sont temporellement corrélés (e.g. avec une différence de quelques secondes concernant leurs temps d'occurrence). En sachant que chaque notification comporte un champs d'horodatage, ce nœud capteur peut éliminer la redondance des parties heure et minute dans ces horodatages en adoptant une représentation appropriée des données agrégées. En contre-partie, une fonction d'agrégation avec perte élimine certains détails et génère des données qui sont moins précises. le système eScan mentionné dans la section 3.3 applique une fonction d'agrégation avec perte [80, 95].

2. **Parfaite ou partielle :** suivant le nombre de sorties d'une fonction d'agrégation, on peut distinguer entre une fonction d'agrégation parfaite qui résulte une seule sortie et une fonction d'agrégation partielle qui considère plus qu'une sortie. Des exemples de fonctions parfaites regroupent les fonctions MIN, MAX, SUM et COUNT [2].
3. **Sensible ou pas aux duplicatas :** dans un WSN, un nœud capteur intermédiaire peut recevoir de multiples copies de la même information. Suivant l'influence des duplicatas sur le résultat renvoyé par une fonction d'agrégation, on parle de fonction d'agrégation qui est sensible ou pas aux duplicatas. Une fonction d'agrégation qui renvoie un résultat qui dépend du nombre de considération d'une même donnée est dite sensible au duplicatas. Contrairement, on parle d'une fonction qui n'est pas sensible aux duplicatas. Comme exemples, La fonction AVG est sensible aux duplicatas alors que la fonction MIN ne l'est pas [80].

Lors de la conception de toute fonction d'agrégation, il est important de tenir en compte des caractéristiques propres aux nœuds capteurs, en particulier, leurs ressources énergétiques et leurs capacités de calcul restreintes.

3.6 Schémas de routage pour agrégation de données

Les schémas de routage jouent un rôle important dans un processus d'agrégation de données. Ils définissent la manière avec laquelle les données prélevées sont routées vers le nœud puits. Avec de tels schémas et par rapport au routage classique où les données sont routées le long du plus court chemin séparant un nœud source d'un autre de destination, il est important que les décisions de routage prises au niveau des nœuds capteurs favorisent la convergence spatiale des données afin que ces dernières puissent être agrégées. Généralement, un protocole d'agrégation peut adopter un schéma de

routage qui se base sur une structure particulière qui est pré-définie au-dessus de la topologie originale du réseau. On parle dans ce cas d'approche structurée. En contrepartie, ce routage peut être mené sans la considération d'une structure particulière en se basant seulement sur des règles locales. On parle dans ce cas d'approche non-structurée. Dans ce qui suit, nous décrivons brièvement certains travaux connexes à chaque catégorie d'approches.

3.6.1 Approches structurées

Les approches adoptant un schéma de routage structuré organisent généralement les nœuds du réseau suivant une certaine hiérarchie. Elles peuvent être classées en deux grandes classes. Celles qui se basent sur une structure en arbre et d'autres qui adoptent une structure en clusters.

3.6.1.1 Approches se basant sur une structure en arbre

Les approches dans cette catégorie établissent préalablement un arbre qui est enraciné au niveau du nœud puits et couvre tous les nœuds sources. Ensuite, elles exploitent cet arbre pour la collection et l'agrégation des données prélevées. Durant le routage, chaque nœud qui reçoit deux ou plusieurs paquets de ses nœuds fils les agrège avec ses propres données et achemine un seul paquet vers son nœud parent. Ce processus se répète jusqu'à ce que les données arrivent au nœud puits. La figure (3.5) illustre un arbre d'agrégation de données.

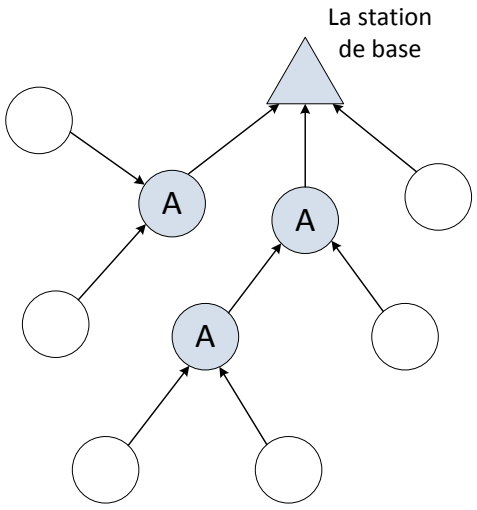


FIGURE 3.5 – Schéma de routage se basant sur une structure en arbre. L'agrégation des données est effectuée au niveau des différentes jonctions.

L'approche la plus communément appliquée lorsqu'une structure en arbre est adoptée consiste à établir un arbre de plus court chemin (shortest path tree) [79]. TAG [96] est un exemple d'approches se basant sur un tel arbre. L'établissement de ce dernier

s'initie par le nœud puits qui est choisi pour être la racine de l'arbre. Il diffuse un message dans lequel il spécifie son identifiant et son niveau (i.e. sa distance du nœud racine) qui est zéro. Tout nœud reçevant ce message met à jour son niveau, s'il est nécessaire, en incrémentant de 1 le niveau contenu dans le message. Ce nœud considère également l'émetteur du message comme nœud parent. Ensuite et dans le cas où il redéfinit son niveau, il remplace le niveau et l'identifiant contenus dans le message par son propre niveau et son propre identifiant et rediffuse le message. Les nœuds reçevant ce dernier répètent les mêmes opérations jusqu'à ce que tous les nœuds définissent leurs niveaux et leurs nœuds parents.

Le protocole Directed Diffusion [90, 95] est un autre protocole qui fait usage d'un arbre de routage. Suivant ce protocole, le nœud puits inonde initialement le réseau par un premier message d'intérêt qui décrit la tâche de perception à attribuer aux nœuds du réseau. Ce premier message a pour but, la découverte de tout nœud potentiel pouvant répondre à cette tâche. Durant l'inondation du message d'intérêt, chaque nœud qui achemine un tel message crée une entrée qui correspond à cet intérêt dans son cache et définit un gradient vers le nœud puits via le voisin qui lui a passé le message. L'inondation de ce dernier se continue ensuite de manière périodique. Lorsqu'un nœud capteur donné détecte un certain phénomène qui correspond à l'une des entrées dans son cache, il commence à acheminer les données prélevées vers le nœud puits suivant les gradients qui ont été préalablement établis. Lorsque le nœud puits commence à recevoir ces données, il procède au renforcement d'un certain chemin lui séparant de ce nœud capteur source. Ce processus de renforcement des chemins entre les différents nœuds sources et le nœud puits émerge un arbre de routage qui est exploité pour l'agrégation des données.

En fait, avec Directed Diffusion, l'agrégation des données se fait de manière opportuniste. En d'autres termes, les données qui proviennent des différents nœuds sources sont agrégées lorsque les chemins de ces sources se chevauchent ensemble. Cependant, cette façon à faire ne permet pas de tirer pleinement profit des avantages de l'agrégation étant donné que cette agrégation peut être réalisée loin des nœuds sources, ce qui ne minimise pas considérablement le nombre de transmissions dans le réseau. Pour cette raison, favoriser le chevauchement précoce des chemins conduit à de meilleurs résultats. Cependant, la réalisation de cette tâche n'est pas une chose aisée, car, pouvoir mener de manière optimale cette agrégation nécessite la construction d'un arbre de Steiner qui est connue pour être NP-Hard [79]. Pour cela, plusieurs heuristiques ont été proposées afin de construire des solutions sous-optimales. CNS (Center at the Nearest Source) [97] est l'une de ces heuristiques. Elle détermine le plus proche nœud source du nœud puits comme point d'agrégation. Les autres sources envoient leurs données vers ce nœud qui les agrège pour envoyer ensuite le paquet résultant au nœud puits. GIT (Greedy Incremental Tree) [97] est une autre heuristique qui fusionne séquentiellement

les chemins des nœuds sources. Dans un premier temps, le plus court chemin séparant le plus proche nœud source du nœud puits est établi. Ensuite, le prochain plus proche nœud source est connecté avec ce chemin. Ce processus est répété jusqu'à ce que les chemins de tous les nœuds sources restants soient établis.

D'autres heuristiques qui visent à définir un arbre de Steiner entre nœuds sont présentées dans la section suivante.

Les deux algorithmes CNS et GIT visent à définir des structures de routage qui maximisent l'agrégation des données (en maximisant le chevauchement des routes) et ayant un nombre de sauts minimal. Le but ici consiste à minimiser au maximum le nombre de transmissions nécessaires à la collection et la délivrance des données prélevées au nœud puits. Cependant, les nœuds qui participent à l'acheminement de ces données ne sont pas obligés d'utiliser leurs puissances maximales de transmission mais, au contraire, ils ont le potentiel de les minimiser afin de conserver davantage d'énergie. En exploitant cette potentialité à contrôler les puissances de transmission des nœuds, certaines approches visent à établir des arbres d'agrégation à haute efficacité énergétique. Le gain en énergie, suite à l'utilisation de ce genre d'arbres, revient non seulement à l'agrégation, mais aussi, à l'utilisation de chemins à basse consommation énergétique durant le relayage des données.

R&M [98] est un protocole distribué qui vise à établir ce genre d'arbres. Il se base sur l'observation qui montre qu'une communication entre deux nœuds u et v peut être plus énergétiquement efficace si elle est menée via un troisième nœud relais w au lieu d'être menée directement entre u et v. Suite à cette observation, l'objectif du protocole R&M est d'identifier ce genre de communications et éliminer tout lien inefficace dans la topologie originale du réseau. Cela mène à une nouvelle topologie réduite pouvant être utilisée pour l'établissement de l'arbre de routage. Deux notions sont basiques au fonctionnement de R&M : *région relais* et *graphe d'enceinte*. La région relais pour une paire de nœuds $u - w$, et comme le montre la figure (3.6a), est l'ensemble des points (i.e. positions des nœuds) du plan pour lesquels il est plus énergétiquement efficace de communiquer avec eux via w au lieu de communiquer avec eux directement. Cette notion de région relais est utilisée afin de définir la région d'enceinte d'un nœud donné, par exemple u. Cette notion est illustrée par la figure (3.6b) où le nœud u découvre trois voisins v_1, v_2 et v_3. Lorsque le nœud u calcule la région relais avec chacun de ces voisins, une certaine région autour de u au-delà de laquelle, il n'est pas énergétiquement efficace de communiquer directement avec d'autres nœuds est définie. Cette région délimitée qui est autour de u est dite région d'enceinte. Maintenant, tout nouveau nœud qui est découvert par u et qui se situe dans l'une des région relais des nœuds précédemment découverts peut être simplement ignoré. Donc, le nœud u considère seulement les nœuds qui se situent dans sa région d'enceinte. Ces nœuds sont considérés alors comme les voisins de u dans la nouvelle topologie construite qui est dite graphe d'enceinte. Après

la définition de cette nouvelle topologie, l'arbre de routage qui est enraciné au niveau du nœud puits est construit au-dessus de cette nouvelle topologie en utilisant l'algorithme classique de Bellman–Ford. Pour mener cet algorithme, les liens entre nœuds sont valués par les coûts énergétiques nécessaires à la communication entre ces nœuds.

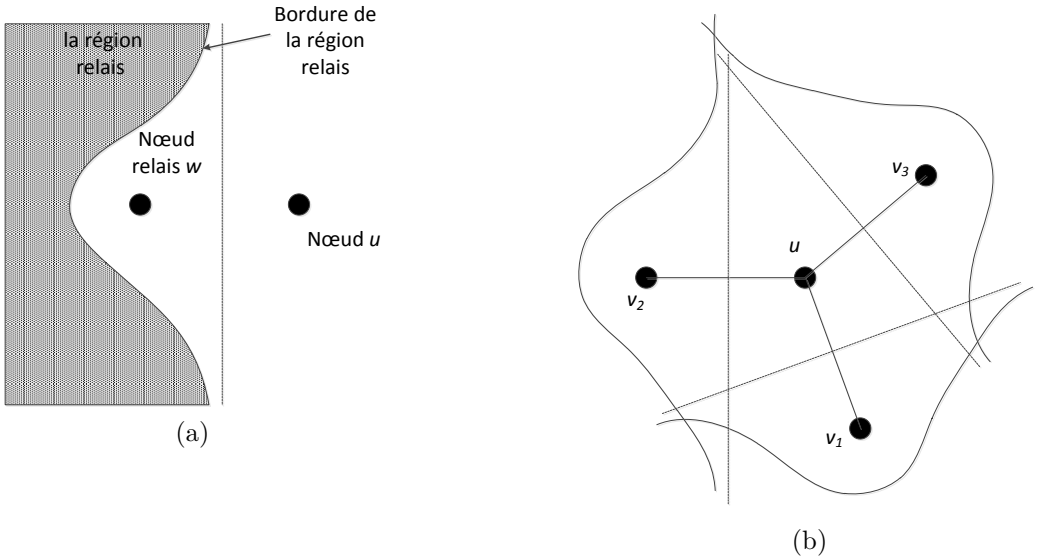
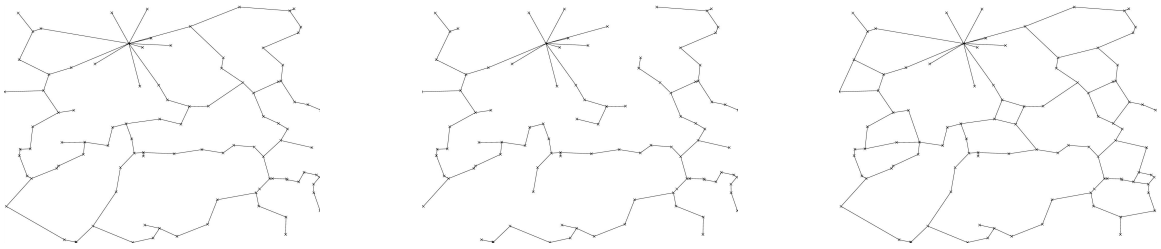


FIGURE 3.6 – Les deux notions élémentaires de R&M : (3.6a) la région relais de la paire de nœuds $u - w$. (3.6b) La région d'enceinte du nœud u.

PEDAP [99] est un autre protocole qui mène ce processus de construction de l'arbre de manière centralisée. En se basant sur les positions des nœuds et en utilisant l'algorithme de Prim, le nœud puits établit l'arbre couvrant minimal, en se considérant comme racine, dans le graphe maximal qui est valué par le coût énergétique nécessaire à la communication entre deux nœuds voisins. Une fois établi, le nœud puits informe chaque nœud capteur de son nœud parent et ses nœuds fils dans le nouveau arbre calculé ainsi que son créneau d'envoi. Les mêmes auteurs proposent dans le même article une deuxième version appelée PEDAP-PA. L'idée de cette version est d'incorporer le niveau énergétique de chaque nœud capteur dans la fonction de coût utilisée pour établir l'arbre couvrant minimal. En se basant sur cette nouvelle fonction, cet arbre est recalculé périodiquement afin de s'adapter aux changements dans l'état du réseau (c.à.d. changement dans les niveaux énergétiques des nœuds ainsi que leurs morts). De cette façon, la durée de vie du réseau est augmentée.

Même si PEDAP et PEDAP-PA donnent de bons résultats, leur inconvénient majeur est le caractère centralisé qui les caractérise. Pour cette raison, les mêmes auteurs proposent une autre version distribuée, appelée L-PEDAP [100], qui peut être menée en considérant seulement les informations de voisinage des nœuds. Cette nouvelle version se base sur des algorithmes tels que LMST (Local Minimum Spanning Tree) [4] et RNG (Relative Neighborhood Graph) [101] qui peuvent établir une approximation de



(a) (b) (c)

FIGURE 3.7 – Différence visuelle entre les topologies : (3.7a) LMST, (3.7b) MST et (3.7c) RNG.

l'arbre couvrant de poids minimal de manière distribuée. L-PEDAP procède en deux phases : *la construction de la topologie creuse* et *l'établissement de l'arbre de routage au-dessus de cette nouvelle topologie*. Durant la première phase, une nouvelle topologie creuse est construite en se basant sur l'algorithme RNG ou LMST. Pour l'algorithme RNG, un lien $e_{u,v}$ entre les deux nœuds u et v est inséré dans la nouvelle topologie seulement s'il n'y a aucun nœud w de sorte que la distance maximale entre celle qui est entre u et w et celle qui est entre v et w est inférieure à celle qui est entre u et v. En outre, avec l'algorithme LMST, la nouvelle topologie est construite comme suit : au début, chaque nœud détermine ses voisins à un saut et calcule localement son arbre couvrant minimal. Ensuite, il considère comme voisins, seulement les nœuds qui sont à un saut dans ce nouvel arbre construit. La nouvelle topologie générée suite à ce processus est un graphe orienté. Deux procédés peuvent être appliqués afin de la convertir en un graphe non orienté. Le premier consiste à considérer tout arc entre u et v dans la nouvelle topologie seulement si les arcs $e_{u,v}$ et $e_{v,u}$ en font partie, respectivement, des arbres locaux de u et v ($LMST^{-}$). Avec le deuxième procédé, un nouvel arc est inclus dans la topologie finale si seulement l'un des deux arcs $e_{u,v}$ et $e_{v,u}$ existe ($LMST^{+}$). La figure (3.7), tirée de [100], illustre les topologies résultantes d'une même topologie après l'application des algorithmes LMST, MST et RNG.

Après l'établissement de la nouvelle topologie, le nœud puits initie l'opération de construction de l'arbre de routage via la diffusion d'un message de découverte de routes à son voisinage dans la nouvelle topologie. Ce message est rediffusé ensuite d'un nœud à un autre afin que chaque nœud définisse son nœud parent et ses nœuds fils. L-PEDAP propose trois méthodes qui peuvent être exécutées par chaque nœud afin de choisir son nœud parent : (1) choisir le nœud à partir duquel le message de découverte des routes est reçu pour la première fois, (2) choisir le voisin qui assure un nombre de sauts minimal vers le nœud puits, et (3) choisir le voisin qui assure le chemin le plus énergétiquement efficace vers le nœud puits.

Outre ces approches, il y a d'autres schémas qui organisent les nœuds du réseau sous la forme d'une seule ou plusieurs chaines disjointes de nœuds. PEGASIS [102]

est le plus connu des protocoles adoptant une telle approche. Avec ce protocole, une chaine qui relie tous les nœuds du réseau est utilisée pour router et agréger les données prélevées. Cette construction peut se faire soit de manière centralisée par le nœud puits, soit de manière distribuée au niveau de chaque nœud. Les différents nœuds capteurs prennent la responsabilité de leadership à tour de rôle afin d'équilibrer la consommation énergétique. À chaque fois, le leader est le seul autorisé à se communiquer avec le nœud puits. Chaque nœud de la chaine qui reçoit des données de son voisin les agrège avec ses propres données et passe un seul paquet vers le prochain nœud dans la chaine. Ce processus se répète jusqu'à ce que le paquet arrive au leader courant. À cet instant, ce leader inclut ses données et envoie le paquet résultant vers le nœud puits.

3.6.1.2 Approches se basant sur une structure en clusters

Avec de telles approches, les nœuds sont organisés en clusters. Pour chacun d'eux un nœud spécial, appelé cluster-head, est élu pour assumer les tâches d'agrégation et la notification du nœud puits du résultat. La figure (3.8) illustre ce principe.

LEACH [103] est le plus connu des protocoles adoptant une structure en clusters. Il fonctionne en rounds successifs. Chaque round commence par une phase d'annonce dans laquelle les cluster-heads sont élus. Chaque nœud capteur décide individuellement de la potentialité de devenir un cluster-head en se basant sur une certaine valeur prédéfinie représentant une mesure du nombre des cluster-heads à créer et un certain seuil qui est calculé localement par chaque nœud. Après cette élection, chaque cluster-head annonce son rôle à son voisinage en utilisant sa puissance maximale de transmission. À la réception de cette annonce, chaque nœud capteur usuel détermine le cluster-head avec lequel il sera associé en choisissant celui qui requiert la puissance minimale de transmission. Une fois ce choix fait, ce nœud informe le cluster-head choisi de sa désirabilité à rejoindre son cluster. À cet instant et suivant les messages reçus, ce cluster-head établit les informations nécessaires concernant son cluster. Une fois les clusters formés, les nœuds exploitent cette structure pour le relayage des données prélevées vers le nœud puits.

Une structure qui se base sur des clusters peut être établie de manière proactive où elle est définie au préalable est maintenue périodiquement comme ce qui est le cas de LEACH. Ce genre de protocoles proactifs sont dédiés aux applications où la collection des données se fait de manière continue. D'un autre côté, cette clusterisation peut être menée de manière réactive afin d'économiser l'énergie durant les périodes d'inactivités du réseau [88]. InFRA [88, 104] est un protocole réactif avec lequel la construction de la structure de routage est déclenchée suite à l'occurrence d'évènements. Le principe consiste à regrouper les nœuds capteurs qui détectent un même évènement en un même cluster. Parmi cet ensemble, un cluster-head est élu. Ensuite, un arbre de routage qui relie les différents cluster-heads existants est formé afin de relayer les

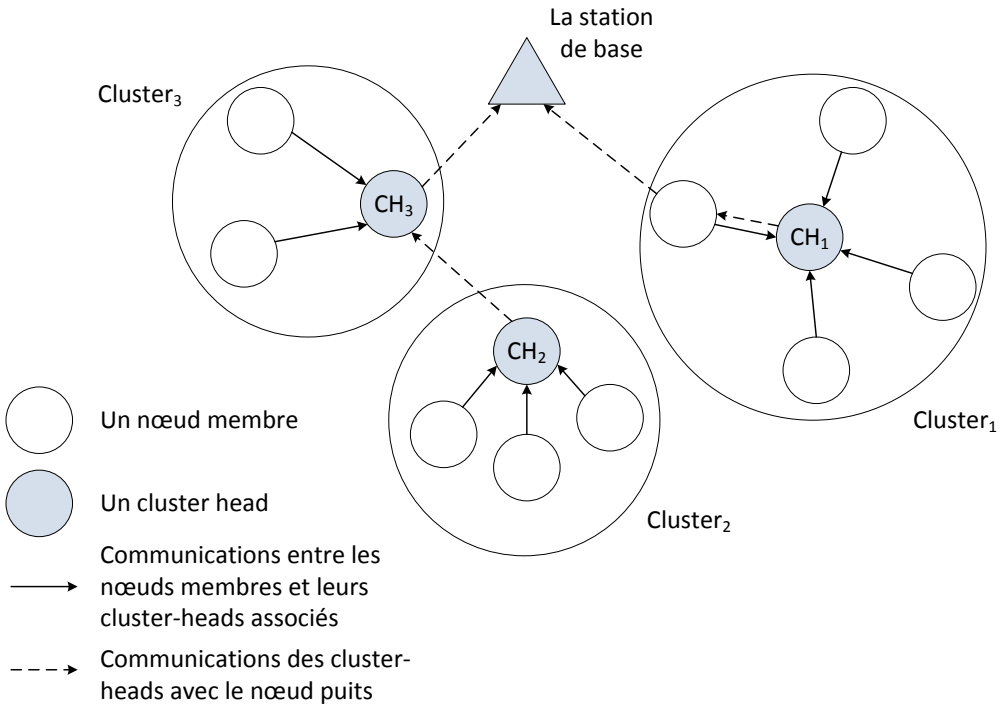


FIGURE 3.8 – Schéma de routage se basant sur une structure en clusters. L'agrégation des données est effectuée au niveau des cluster-heads.

données prélevées vers le nœud puits. Durant cette construction, InFRA procède à un processus d'assignation des rôles en supposant les rôles suivants :

1. **Puits :** nœud qui est intéressé par les évènements qui arrivent,
2. **Collaborateur :** un nœud capteur qui détecte un évènement mais qui n'assume pas le rôle du cluster-head,
3. **Coordinateur :** un nœud capteur détectant l'occurrence d'un évènement et assumant le rôle du cluster-head,
4. **Relais :** un nœud faisant partie de l'arbre de routage qui connectent les cluster-heads avec le nœud puits.

Ces rôles sont assignés durant le fonctionnement de InFRA qui suppose deux phases principales : la formation des clusters et la formation des routes entre ces clusters et le nœud puits.

La première phase concerne seulement les nœuds sources (i.e. les nœuds qui détectent des évènements). Ces nœuds procèdent à un processus d'élection du nœud coordinateur. Plusieurs métriques peuvent être appliquées pour élire ce nœud. Cependant, InFRA considère le nœud ayant le plus petit identifiant pour assumer ce rôle. Tous les autres nœuds détectant le même évènement deviennent des collaborateurs. Durant la formation du cluster, un arbre de plus court chemin qui est enraciné au niveau du coordinateur et qui couvre tous les collaborateurs est construit afin de pouvoir router les données à l'intérieur du cluster.

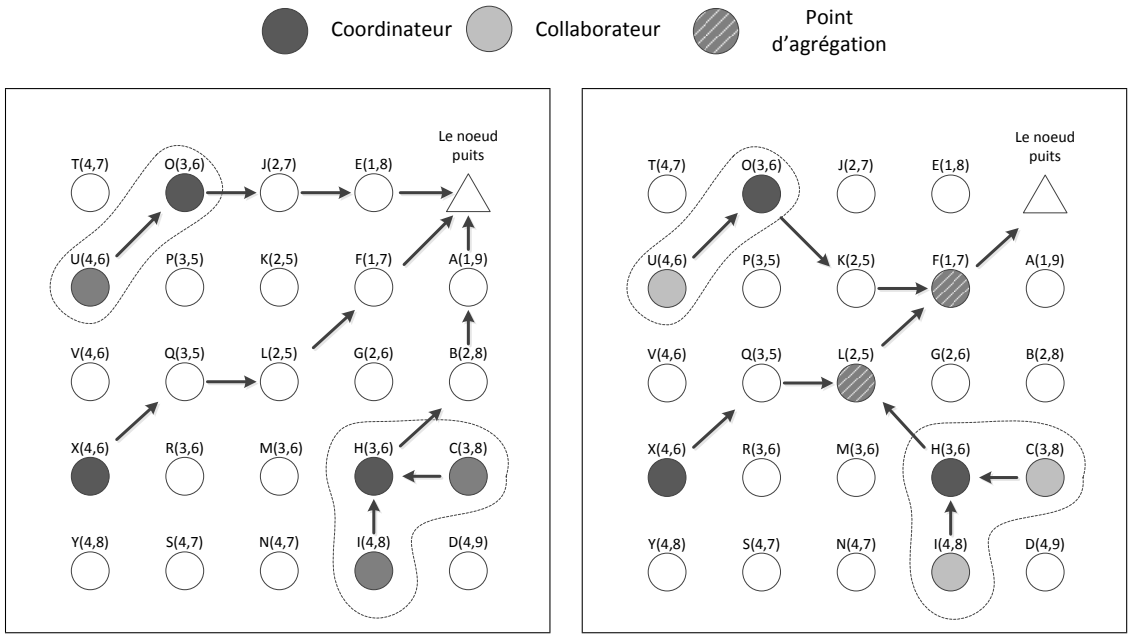
Une fois les coordinateurs élus, la deuxième phase qui consiste à établir l'arbre de routage qui relie ces coordinateurs avec le nœud puits commence. Durant cette formation, InFRA ne vise pas seulement à établir un simple arbre, mais par contre, le but est d'établir un arbre qui maximise le chevauchement des routes, et par conséquent, qui maximise l'agrégation des données. En fait, le problème revient à construire un arbre de Steiner qui est enraciné au niveau du nœud puits et qui couvre tous les nœuds coordinateurs. Pour cela, InFRA propose une nouvelle heuristique afin de construire cet arbre de Steiner. Le principe consiste à choisir le voisin approprié à chaque saut durant le relayage des données. Le voisin approprié qui est choisi par un nœud donné est soit celui qui est le plus proche du nœud puits, soit celui qui minimise la distance vers les autres coordinateurs dans le cas d'égalité entre plusieurs voisins vis-à-vis de leurs distances du nœud puits.

Pour pouvoir mettre en opération ce processus, certaines données sont nécessaires. Pour cette raison, chaque coordinateur et après son élection inonde le réseau par des messages de contrôle. Chaque nœud recevant ce message calcule sa distance de ce coordinateur. Une fois que tous les coordinateurs ont inondé leurs messages, chaque nœud peut calculer maintenant la distance agrégée vers tous les coordinateurs existants. Comme nous pouvons le voir dans la figure (3.9b), le nœud N est à 2 sauts du coordinateur X, à 1 saut du coordinateur H et à 4 sauts du coordinateur O. Ce qui donne une somme de 7. À cet instant chaque nœud peut choisir le voisin approprié lors de l'acheminement des données.

La figure (3.9) montre l'arbre de plus court chemin et l'arbre qui résulte suite à l'heuristique adoptée par InFRA. Comme nous pouvons constater, les routes avec l'arbre de plus court chemin peuvent ne jamais se chevaucher alors qu'avec InFRA c'est totalement différent.

InFRA considère deux types d'agrégation : intra-cluster et inter-cluster. La première, Intra-cluster, est menée à l'intérieur du cluster. Elle peut être menée par un nœud collaborateur et par le nœud coordinateur. Un nœud collaborateur a la possibilité d'agréger les données lui arrivant de ses fils avec ses propres données lorsqu'il est considéré comme un intra-cluster relais. Un nœud coordinateur agrège les données qui arrivent de ses membres. Pour ce qui de l'agrégation inter-cluster, elle est menée au niveau des points d'agrégation dans l'arbre de routage formé entre les nœuds coordinateurs et le nœud puits.

Le deuxième protocole réactif considéré ici dans cette section est le protocole DRINA [79, 105]. Son principe est similaire à InFRA. Cependant, il y a plusieurs différences qui rend DRINA meilleur. La principale différence consiste en la manière avec laquelle les routes qui connectent les nœuds coordinateurs avec le nœud puits sont formées. Cette façon à faire qui nécessite moins de paquets de contrôle et qui donne un arbre de routage qui est constitué de moins de nœuds de Steiner, et par conséquent, qui assure

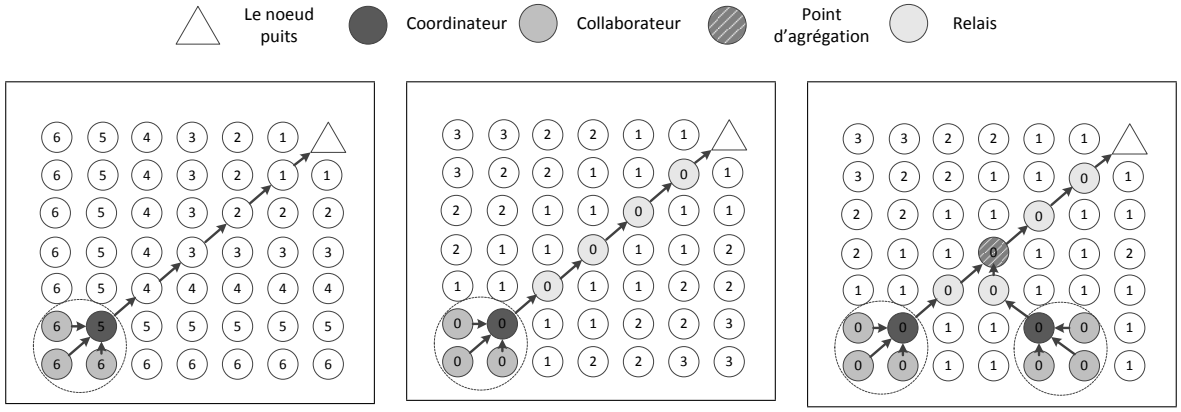


(a) Arbre de plus court chemin. (b) Formation des routes avec InFRA.

FIGURE 3.9 – Formation de l'arbre de routage qui connecte les coordinateurs avec le nœud puits. La notation *a(b,c)* qui est utilisée pour libeller les nœuds signifie que le nœud a est de b sauts du nœud puits et que la somme des distances lui séparent des coordinateurs existants est c.

moins de transmissions.

DRINA fonctionne en trois phases. Dans la première phase, un arbre, appelé arbre des sauts, et qui connecte tous les nœuds du réseau avec le nœud puits est créé. Durant cette phase, tous les nœuds définissent leurs distances (en termes du nombre de sauts) du nœud puits. Après ce processus, le réseau reste inactif jusqu'à l'occurrence d'évènements. Lorsqu'un évènement arrive, la deuxième phase qui consiste à former les clusters commence. Comme InFRA, les nœuds capteurs qui détectent le même évènement sont regroupés en un même cluster. Cependant, le noeud qui est élu comme coordinateur est celui qui assure une distance minimale du nœud puits. Une fois les clusters formés, la troisième phase est déclenchée. Durant cette phase, les routes entre les nœuds coordinateurs et le nœud puits sont formées. Cette formation des routes dépend de l'ordre d'occurrence des évènements. Le coordinateur responsable du premier évènement qui arrive envoie un message d'établissement de route vers son prochain saut. Dans ce cas du premier évènement, le prochain saut est le voisin qui assure le plus court chemin séparant ce coordinateur du nœud puits. Ce message est acheminé d'un nœud à un autre jusqu'à ce qu'il arrive au nœud puits. La figure (3.10a) illustre le chemin construit suite à l'occurrence du premier évènement. Afin de pouvoir créer les routes des futurs évènements et durant le relayage du message d'établissement de route, chaque nouveau relais diffuse un message de configuration du réseau qui est inondé sur l'ensemble du



(a) Formation de la route vers un seul évènement.

(b) Mise à jour de l'arbre des sauts.

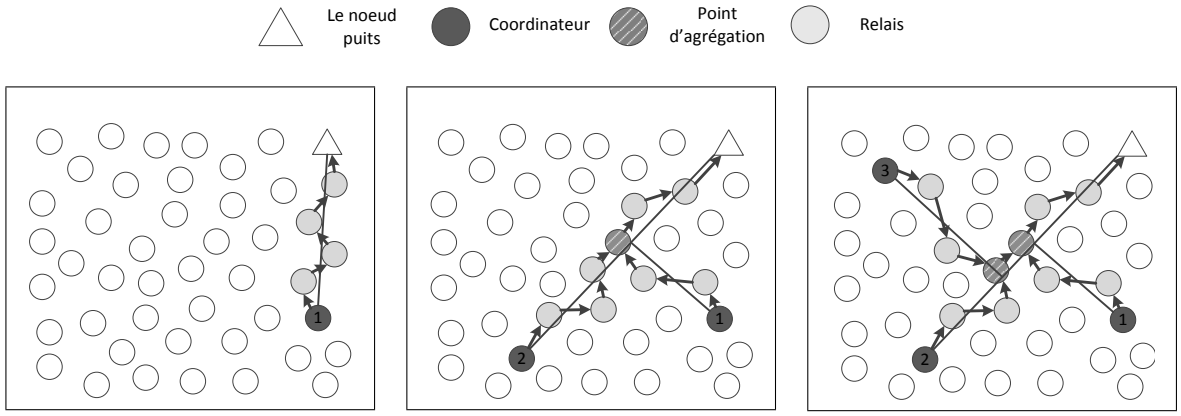
(c) Arbre de routage crée avec deux évènements.

FIGURE 3.10 – Principe de DRINA vis-à-vis de la formation des routes entre les coordinateurs et le nœud puits.

réseau. Le but ici consiste à mettre à jour l'arbre des sauts créé au début, pour que chaque nœud soit conscient du plus court chemin le séparant de la route du premier évènement. La figure (3.10b) illustre cette mise à jour. Maintenant et à l'occurrence du deuxième évènement et comme le montre la figure (3.10c), le coordinateur élu sélectionne, durant la formation de sa route, le prochain saut qui assure le plus court chemin vers le plus proche nœud faisant partie de la route du premier coordinateur. Durant cette définition de la deuxième route, l'arbre des sauts est, à nouveau, mis à jour. Ce processus ce répète avec chaque coordinateur nouvellement créé. De cette façon, chaque route nouvellement crée est connectée avec le plus proche nœud qui fait partie de l'arbre de routage connectant les coordinateurs existants avec le nœud puits.

Comme InFRA et DRINA, DST [9] est un autre protocole qui est dédié aux réseaux de capteurs sans fil évènementiels. Il vise à définir une structure de routage qui relie les nœuds capteurs sources avec le nœud puits et qui assure un nombre minimal de transmissions. Cependant et par rapport aux deux protocoles InFRA et DRINA, il tire pleinement profit des coordonnées des nœuds capteurs et établit une certaine structure géométrique sous-jacente qui est utilisée pour établir les routes des différents coordinateurs. Cette façon à faire contribue considérablement à minimiser les messages de contrôle qui sont utilisés pour établir la structure de routage.

DST procède en quatre phases. Dans la première phase, le réseau est configuré. Donc, les nœuds capteurs enregistrent les coordonnées du nœud puits et prennent conscience de leurs nœuds voisins et leurs coordonnées. La deuxième phase est déclenchée suite à l'occurrence d'évènements dans laquelle les clusters sont formés et les nœuds coordinateurs sont élus. Par rapport aux deux protocoles précédents, la différence consiste en la métrique qui est adoptée pour élire chaque coordinateur. Avec DST, ce dernier est celui qui a la distance euclidienne la plus minimale le séparant du



(a) Établissement de la route du coordinateur du premier évènement.

(b) Établissement de la route du coordinateur du deuxième évènement et la mise à jour de celle du premier évènement.

(c) Établissement de la route du coordinateur du troisième évènement.

FIGURE 3.11 – Principe d'établissement des routes entre coordinateurs avec DST.

nœud puits parmi tous les nœuds détectant le même évènement. Une fois qu'un nœud coordinateur donné est élu, la troisième phase commence. Dans cette phase, chaque coordinateur informe le nœud puits de sa position. Le nœud puits lui envoie en retour les coordonnées de tous les coordinateurs existants. Il informe également ces derniers des coordonnées de ce nouveau coordinateur. Donc, à chaque instant, chaque coordinateur est conscient des autres coordinateurs existants et de leurs coordonnées. Le but derrière cette démarche est de donner aux coordinateurs l'opportunité d'établir une structure géométrique qui est exploitée pour router les données entre eux et le nœud puits. Cette structure géométrique émerge suite au processus suivant qui représente la phase finale de DST. L'idée consiste à laisser le coordinateur le plus éloigné du nœud puits établir son segment de droite vers le nœud puits. Ensuite, le deuxième plus éloigné coordinateur forme son segment de droite vers le plus proche point du premier segment. Ensuite, vient le tour du troisième plus éloigné coordinateur, ensuite, le quatrième, et ainsi de suite. Chacun de ses nouveaux coordinateurs forme son segment vers le plus proche point des segments préalablement établis. Après la formation des segments, les nœuds capteurs qui sont proches de ses segments sont considérés comme nœuds relais. En outre, les nœuds où les routes se chevauchent sont considérés comme points d'agrégation. La figure (3.11) illustre ce principe de formation de routes de DST. Avec cette figure, chaque évènement est supposé détecté par un seul nœud capteur. Donc, aucune clusterisation n'est supposée. En plus, les numéros qui sont à l'intérieur des nœuds coordinateurs représentent l'ordre d'occurrence des évènements avec lesquels ces coordinateurs sont associés.

L'avantage de DST par rapport à InFRA et DRINA et outre que la minimisation du nombre des messages de contrôle utilisés, est sa capacité à équilibrer la consommation

énergétique entre les nœuds capteurs du réseau. Cela est dû au fait que les routes formées entre coordinateurs ne dépendent pas de l'ordre d'occurrence des évènements. Donc, ils ne sont pas statiques durant toute la période d'occurrence des évènements. Cela n'est pas le cas avec InFRA et DRINA où les routes formées dépendent de l'ordre d'occurrence des évènements et restent statiques tant que les évènements sont toujours actifs.

Il est à noter que d'autres stratégies peuvent être appliquées pour définir la façon avec laquelle les segments de droites des différents coordinateurs sont formés. Par exemple, au lieu d'établir, en premier, les segments de droites des coordinateurs les plus éloignés du nœud puits, il est possible de commencer par les segments de droites des coordinateurs les plus proches. En outre, la meilleure combinaison, c.à.d. celle qui assure la distance euclidienne totale la plus courte, peut être considérée [106].

Les trois protocoles InFRA, DRINA et DST visent à définir des structures de routage qui maximisent l'agrégation des données et ayant un nombre minimal de sauts. À part cet objectif, il y a d'autres approches qui visent à minimiser le coût total des liens de la structure en associant à chaque lien un certain coût qui reflète certains objectifs de conception. WDARS [107] est un exemple de telles approches. II partage plusieurs idées de conception avec DRINA. Cependant, le choix du meilleur prochain saut durant la formation des clusters et durant la formation des routes dépend de trois fonctions innovantes qui considèrent les distances (en termes du nombre de sauts) des nœuds capteurs du nœud puits, leurs distances de l'arbre des sauts, leurs énergies résiduelles et la taille de leurs tampons. L'objectif ici consiste à pouvoir choisir les points d'agrégation les plus appropriés, d'équilibrer la consommation énergétique et à augmenter le débit du réseau.

3.6.2 Approches non-structurées

Les approches qui se basent sur la formation des structures bien déterminées tirent pleinement avantage de l'agrégation étant donné que l'ordre avec lequel les données sont acheminées vers le nœud puits est bien défini. Ces approches et particulièrement celles qui établissent ces structures proactivement sont bien adéquates aux applications avec lesquelles tous les nœuds du réseau participent dans la collection des données. Cependant, de telles approches peuvent ne pas être bénéfiques lorsque des applications évènementielles sont considérées. En fait, avec de telles applications, le réseau est caractérisé par une grande dynamique et les nœuds capteurs sources changent continuellement suivant les évènements qui arrivent. Dans ce cas, deux façons peuvent être adoptées afin de construire la structure de routage :

1. **De manière réactive :** Elle peut être construite de manière réactive à l'occurrence d'évènement et maintenue continuellement afin de suivre cet évènement.

Cependant, la charge engendrée suite à cette construction et cette maintenance peut l'emporter sur le bénéfice engendré par l'agrégation.

2. **De manière proactive :** avec cette façon, l'énergie des nœuds est consommée inutilement durant les période d'inactivité du réseau. En outre, on peut ne pas profiter pleinement de l'agrégation étant donné qu'il y a une grande possibilité que des nœuds capteurs adjacents et qui captent le même évènement utilisent des chemins totalement disjoints pour envoyer leurs données [89].

Motivés par ces contraintes et dans le contexte des réseaux de capteurs sans fil évènementiels, certains auteurs ont proposé d'autres approches qui assurent une agrégation efficace sans la construction et la maintenance explicite d'une structure particulière. Le principe de base consiste à concevoir des techniques faisant usage d'informations locales afin de favoriser la convergence spatiale et temporelle des données. DAA+RW [89] est le premier protocole qui est conçu dans ce sens. Il exploite le mode anycast de communication afin de favoriser la convergence spatiale. Avec ce mode, la couche réseau donne à la couche MAC la liberté de choisir le prochain saut approprié parmi un ensemble de voisins candidats. Pour cela, chaque nœud qui a des données à envoyer diffuse un message RTS à son voisinage. Tout nœud reçevant ce message est candidat pour être le prochain saut. Cependant, ces récepteurs ont différentes priorités pour répondre à ce message RTS. Un récepteur qui a des données de même type que celle de l'émetteur et qui est plus proche du nœud puits par rapport à cet émetteur est prioritaire. Vient en deuxième priorité, le récepteur qui a des données similaire mais qui est plus loin du nœud puits. Et en dernière priorité vient le récepteur qui n'a pas de données de même type mais qui assure un avancement vers le nœud puits. Donc, le voisin le plus prioritaire va être le premier à répondre par un message CTS et les autres voisins vont annuler leurs réponses à l'écoute de ce message CTS. À ce moment, ce voisin est choisi pour être le prochain saut. Pour ce qui est de la convergence temporelle, DAA+RW fait usage d'une simple technique en laissant chaque nœud attendre une certaine durée aléatoire avant d'envoyer son paquet. Cependant, cette technique peut mener à des mauvaises performances si les nœuds qui sont plus proches du nœuds puits utilisent des périodes très courtes. D'autres approches, comme SFEB [108], qui améliorent ce protocole ont été proposés.

DASDR [109] est un autre protocole qui ne suppose aucune structure. Il se focalise seulement sur la convergence spatiale. Son idée consiste à orienter les paquets vers les prochains sauts qui ont des tampons avec une large occupation afin d'augmenter la probabilité d'agrégation.

3.7 Agrégation de données et communication en temps-réel

Avec la prolifération des réseaux de capteurs sans fil et l'émergence d'applications sensibles aux délais, il devient nécessaire de considérer cet aspect de communication en temps réel lors de la conception des protocoles qui sont dédiés à ces réseaux. Comme nous l'avons vu dans la section (3.4), même si l'agrégation des données contribue considérablement à la minimisation de la consommation énergétique, elle peut augmenter le délai de transmission de bout-en-bout des données, et donc, elle peut ne pas être applicable dans le cas des applications temps-réel. En fait, cela est dû aux temps d'attentes dont chaque nœud agrégateur doit attendre afin de pouvoir agréger ses données reçues. Lorsque cette communication en temps-réel est à considérer, le problème d'agrégation peut être formulé comme un problème d'arbre de Steiner sous-contrainte [87]. Ce dernier qui est connu pour être NP-hard [110] doit assurer les deux points suivants :

1. Chaque chemin séparant chaque nœud capteur source du nœud puits doit être à délai borné.
2. L'arbre de Steiner final généré doit assurer une consommation énergétique minimale.

Donc, il est nécessaire d'assurer un certain compromis entre l'énergie totale dissipée et la délivrance en temps réel des données prélevées tout en assurant le chevauchement des routes pour que les données puissent être agrégées. Avec ou sans la considération de l'agrégation des données, plusieurs travaux se sont intéressés à ce problème de communication en temps réel dans les réseaux de capteurs sans fil. SPEED [111] est l'un des premiers protocoles qui a été conçu dans ce sens. Il présente beaucoup de caractéristiques qui le rendent approprié à ce genre de réseaux. Outre la garantie d'une communication avec un délai de bout-en-bout prédictible via l'assurance d'une certaine vitesse de délivrance prédéterminée à travers le réseau, il requiert peu de messages de contrôle et maintient seulement une table de voisinage afin de définir ses voisins et leurs coordonnées. SPEED se base sur un routage géographique en restreignant les prochains sauts potentiels au niveau d'un nœud donné seulement aux voisins qui peuvent assurer un avancement vers le nœud de destination (i.e. les voisins qui sont plus proches au nœud de destination par rapport au nœud courant). Cet ensemble est davantage restreint afin de prendre seulement les voisins qui assurent une vitesse de relayage qui est au-dessus d'un certain seuil donné. Il se peut qu'aucun voisin qui assure cette vitesse ne soit trouvé à cause d'une congestion potentielle. Dans ce cas, il est nécessaire d'éliminer certains paquets et diminuer la charge qui est exercée sur les nœuds en amont afin de maintenir une vitesse qui est au-dessus de ce seuil. En outre, les nœuds qui sont en aval peuvent recevoir l'ordre à détourner leurs paquets reçus vers d'autres chemins et

prévenir par conséquent toute future élimination de paquets.

En se basant sur SPEED, d'autres protocoles qui ajoutent davantage d'options ont été proposés. Par exemple, avec MMSPEED [112], et au lieu d'assurer une seule vitesse de relayage de données dans le réseau, une multitude de vitesses sont garanties et différents niveaux de fiabilité sont assurés. Contrairement à MMSPEED où la conservation de l'énergie des nœuds n'est pas un souci de conception étant donné que les applications cibles sont celles qui ont une courte durée de fonctionnement, RPAR [113] est un autre protocole qui est conçu de manière à assurer, non plus, la délivrance en temps-réel des données, mais par contre, de manière à minimiser, autant que possible, l'énergie qui est dissipée suite à cette délivrance. RPAR assure cela à travers un schéma de contrôle des puissances de transmission des nœuds. Avec ce schéma, chaque nœud élève sa puissance de transmission afin d'assurer une vitesse de relayage pouvant garantir la délivrance en temps opportun des données. En contre-partie, cette puissance est diminuée lorsqu'une vitesse de relayage satisfaisante est garantie.

JiTS [114] est un autre protocole qui vise ce genre d'applications temps-réel. L'idée principale consiste à retarder les paquets au niveau des nœuds intermédiaires en les laissant attendre un certain temps d'attente sans que leur délivrance avant leurs dates limites ne soit affectée. Cette façon à faire contribue à minimiser la chance d'occurrence des collisions, et surtout, elle donne l'opportunité à agréger les paquets qui sont en route vers le nœuds puits. En se basant sur cette idée et sur le principe du protocole DAA+RW (présenté dans la section (3.6.2)), les auteurs de [115] ont proposé RAG. Ce dernier est un protocole d'agrégation de données qui ne maintient aucune structure de routage particulière. Il vise à maximiser le gain en agrégation tout en assurant la délivrance en temps réel de ces données agrégées. Par rapport à DAA+RW, les temps d'attente au niveau des nœuds intermédiaires ne sont pas définis aléatoirement, mais par contre, ils sont déterminés suite à un schéma qui est similaire à celui de JiTS. En outre, les nœuds voisins qui répondent par des messages CTS doivent être éligibles à être des prochains sauts. En d'autres termes, ils doivent assurer une vitesse de relayage qui peut garantir la délivrance en temps opportun des données agrégées.

Outre les travaux connexes aux réseaux de capteurs sans fil, les travaux qui s'intéressent aux réseaux de capteurs et d'acteurs sans fil (WSANs, Wireless Sensors and Actor Networks) sont de bons exemples de travaux qui visent à assurer un compromis entre l'énergie qui est dissipée par les nœuds capteurs durant la collection des données et leur délivrance en temps réel aux nœuds acteurs [22]. Avec ces réseaux, les nœuds capteurs effectuent la tâche de perception et les nœuds acteurs prennent les actions appropriées sur la base des données perçues. En fait, lorsqu'on a à faire à une application qui est destinée à être mise en œuvre par un WSAN, certains critères de conception sont hautement recommandés. Le premier est pertinent au caractère temps réel de l'application. Il reflète le délai, toléré par l'application, qui sépare le temps du prélèvement

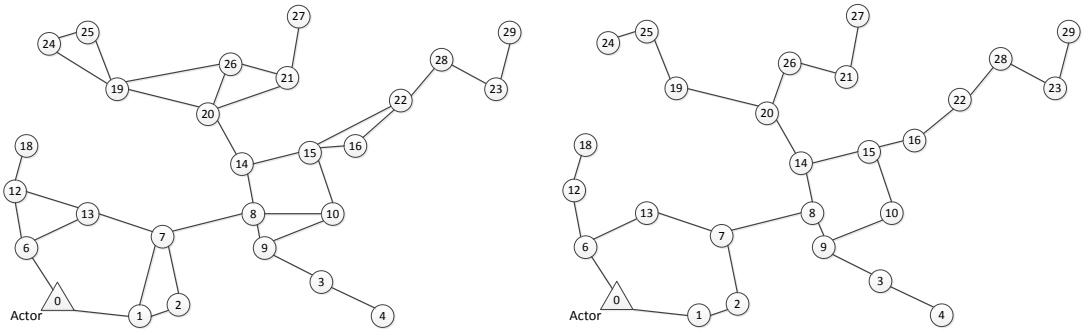
de l'évènement et celui où l'action est entreprise par le nœud acteur une fois cet évènement est récupéré. Il est forcement nécessaire de respecter ce délai, qui diffère d'une application à une autre, afin que l'action réalisée ait un sens. Le deuxième critère est pertinent aux ressources restreintes des objets intervenant dans un WSAN. Parmi ces ressources, l'énergie est la plus contraignante, ce qui nécessite sa bonne gestion.

DEPR [116] est l'un des premiers protocoles qui a été conçu pour ces réseaux. Il permet le partitionnement implicite des nœuds capteurs situant dans la zone où l'évènement a eu lieu en de multiple clusters. Chacun de ces derniers constitue un arbre appelé da-tree (data aggregation tree), créé de manière réactive à l'occurrence d'évènement, et qui est enraciné au niveau du nœud acteur (considéré comme un cluster-head) le plus proche. Au cours de ce partitionnement, l'établissement des chemins vers ce nœud acteur se font en assurant la délivrance opportune des évènements prélevés tout en essayant d'assurer une consommation énergétique minimale. Pour assurer cela, les nœuds capteurs changent leurs comportements suivant une certaine rétroaction de la part du nœud acteur qui est définie par une valeur du fiabilité observée. Cette valeur est définie comme étant le ratio entre le nombre des paquets reçus à temps et le nombre total des paquets générés dans un intervalle donné. Suivant cette rétroaction et lorsque la valeur de fiabilité observée est au-dessous d'un certain seuil donné, chaque capteur choisit le voisin le plus proche géographiquement de la destination finale. Cela aide à minimiser le nombre de sauts qui mènent vers cette destination, ce qui minimise le temps de latence des données. Cette minimisation se fait au détriment de la consommation énergétique. Dans le cas contraire, lorsque la valeur de fiabilité observée est au-dessus du seuil prédéterminé, ce qui reflète un excès de fiabilité, chaque nœud capteur délivre ses données vers son voisin le plus proche afin d'utiliser une petite puissance de transmission. Il est à noter qu'une agrégation opportuniste est adoptée où les données sont agrégées lorsqu'elles se rencontrent au niveau des mêmes nœuds capteurs.

Dans [117, 118], les auteurs proposent une approche qui permet d'associer chaque nœud capteur avec le nœud acteur le plus proche de lui. Cette association permet la formation préalable de clusters autour des nœuds acteurs existants (considérés comme cluster-head). Leur approche consiste en trois phases. La première, d'apprentissage, exécutée au déploiement initial, et assurée par un protocole baptisé ADP (Actuator-discovery Protocol), donne l'opportunité à chaque nœud capteur de procéder à un attachement à un nœud acteur approprié. La deuxième phase de coordination permet à chaque nœud capteur de router les évènements qui se produisent au nœud acteur auquel il est attaché, et cela, en utilisant le chemin établi lors de la première phase. Dernièrement, la troisième phase maintient la clusterisation initiale et permet le rétablissement des chemins qui mènent vers chaque nœud acteur lorsqu'ils deviennent obsolètes. En ce qui concerne la QoS assurée, l'approche proposée garantit un court temps de latence en considérant le plus court chemin, en termes du nombre de sauts,

qui sépare chaque nœud capteur de son nœud acteur correspondant. Pour ce qui est de l'efficacité énergétique, l'approche favorise la modification des chemins d'acheminement des données au cours du fonctionnement du réseau sur la base d'état énergétique des différents nœuds situant sur différents chemins. En outre, elle permet aux nœuds acteurs, par une forme d'une supervision centrale, de contrôler les duty cycling de leurs nœuds capteurs correspondants.

CCR [119] est un autre protocole qui est dédié à ces réseaux de capteurs et d'acteurs sans fil. Avec ce protocole, les nœuds capteurs s'auto-organisent pour former un ensemble de clusters. Ensuite et, suite aux messages de contrôle qui sont inondés par chaque nœud acteur, chaque cluster-head choisit, comme nœud acteur de destination, celui qui peut couvrir la zone géographique de son cluster. Dans le cas où plusieurs acteurs couvrent cette zone, il sélectionne celui qui assure le plus court chemin. CCR tire profit de l'agrégation de données en autorisant les cluster-heads à agréger les données qui arrivent de leurs membres. Cependant, Cette agrégation doit être menée sans que la délivrance avant la date limite des données soit violée.



(a) Topologie originale. (b) LMST.

FIGURE 3.12 – Un réseau de capteurs sans fil avec un seul nœud acteur et modélisé avec deux topologies distinctes.

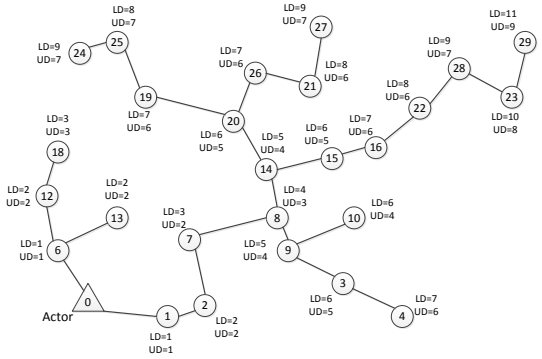
DEDA [12] s'intéresse au problème de construction d'un arbre d'agrégation avec des chemins bornés en délais et une consommation énergétique minimale. DEDA établit, dans un premier temps, et en se basant sur l'algorithme LMST, une nouvelle topologie creuse du graphe original du réseau (figure (3.12)). Ensuite, un arbre qui est enraciné au niveau du nœud acteur et qui couvre tous les nœuds capteurs avec lesquels cet acteur est supposé connecté est construit au-dessus de la nouvelle topologie. Cet arbre est utilisé, sans modification, si la date limite imposée n'est pas violée. Dans le cas contraire, cet arbre est ajusté en remplaçant certains chemins de la topologie creuse par d'autres liens du graphe original. Cet ajustement se fait sur la base d'une certaine valeur qui est calculée au niveau de chaque nœud et qui reflète l'avancement désiré sur la topologie creuse afin que la date limite soit respectée. Cette valeur est calculée par la formule suivante : $DEP(DEsiredProgress) = \lceil \frac{LD(w)}{DL-MED} \rceil \times l(w)$. $LD(w)$ représente

la distance qui sépare le nœud w du nœud acteur au niveau de la topologie creuse. DL est la date limite qui est spécifiée par l'utilisateur. MED est le plus grand délai déjà expérimenté par tous les rapports reçus à partir des voisins de w. $l(w)$ est défini suivant l'intensité du trafic réseau. Il est égal à 1 si cette intensité n'est pas élevée. Sinon, il correspond au degré du nœud w. Étant donné que le temps de latence des données est affecté par le nombre des nœuds qui concurrent pour l'accès au canal sans fil, DEDA route les données prélevées le long du chemin dont la somme des degrés de ses nœuds est minimale si le trafic réseau est important. Sinon, le chemin qui assure un nombre de sauts minimal est sélectionné.

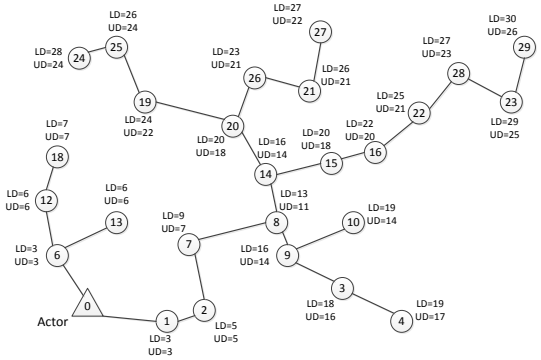
La figure (3.13) illustre le principe du protocole DEDA. Les deux figures (3.13a) et (3.13c) représentent, respectivement, les arbres initial et final construits avec une borne temporelle $DL = 7$ (nombre de sauts à ne pas dépasser) lorsque la distance entre deux nœuds correspond au nombre de sauts du chemin entre eux. Comme le montrent ces deux figures, à partir d'un arbre initial qui est construit au dessus la topologie creuse créée (LMST), un nouvel arbre est ensuite généré suite aux ajustements effectués par les nœuds capteurs afin que la date limite requise ne soit pas dépassée. Durant l'établissement de l'arbre initial qui est construit suite à la requête du nœud acteur, chaque nœud capteur enregistre sa distance de ce nœud au niveau de la topologie creuse (LD) et celle qui est au niveau de la topologie originale (UD). En se basant sur ces valeurs, les nœuds feuilles de l'arbre initial déclenchent le processus d'ajustement de cet arbre. Examinons le nœud 22 dans la figure (3.13a) ayant le nœud 16 comme nœud parent. Durant le processus d'ajustement, le nœud 28 informe le nœud 22 qu'il est choisi comme nœud parent. Il accompagne cette annonce avec la distance qui sera parcourue par les rapports futurs qui seront routés par ce nœud 28. Dans ce cas, cette distance est égale à 1. En fait, le nœud 28 ignore la distance qui sera parcourue par les paquets générés à partir des nœuds 23 et 29, car, ces paquets ne seront jamais délivrés à temps, vu que les distances qui séparent ces deux nœuds du nœud acteur dans la topologie originale sont respectivement, 8 et 9. Une fois que le nœud 22 reçoit l'annonce du nœud 28, il change son nœud parent et choisit le nœud 15 à la place, car, ce dernier assure un avancement de 2 sauts dans la topologie creuse (LD(22)-LD(15)=2) qui correspond à sa valeur $DEP = \lceil \frac{8}{7-1} \rceil \times 1 = 2$.

Dans le cas d'un trafic réseau intense, un processus similaire est adopté. La seule différence concerne la distance entre deux nœuds qui correspond dans ce cas à la somme des degrés des nœuds du chemin entre ces deux nœuds. Un exemple de construction et d'ajustement de l'arbre d'agrégation est illustré par les deux figures (3.13b) et (3.13d) qui représentent, respectivement, les arbres initial et final construits avec une borne temporelle $DL = 21$ (somme des degrés à ne pas dépasser).

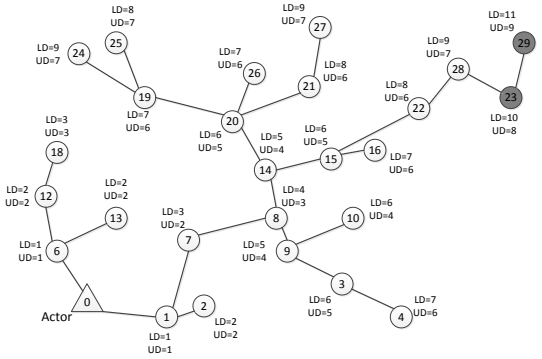
Jusqu'à ici, la plupart des approches vues visent à construire des structures de routage qui assurent une délivrance en temps réel des données avec une consommation



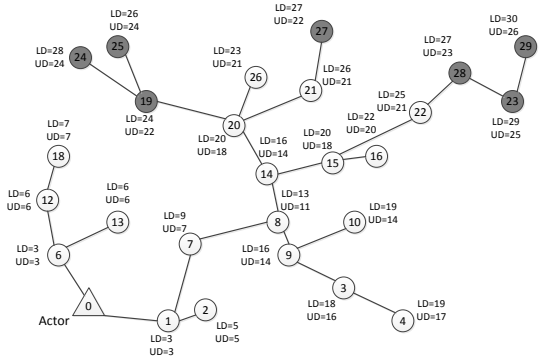
(a) Arbre initial (hop count-based).



(b) Arbre initial (degree sum-based).



(c) Arbre final avec DL=7 (hop count-based).



(d) Arbre final avec DL=21 (degree sum-based).

FIGURE 3.13 – Principe du protocole DEDA.

énergétique minimale. Outre ces approches, il y a d'autres propositions qui supposent l'existence préalable de cette structure et leurs contributions se résument à la façon dont les temps d'attente sont distribués sur les nœuds de la structure prédéfinie sans que la date limite ne soit violée. Le problème ici correspond au problème de planification du processus d'agrégation et le lecteur peut se référencer à [10] pour un état de l'art récent et détaillé de cette classe d'approches.

3.8 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la technique d'agrégation des données comme une technique importante de conservation de l'énergie des nœuds capteurs. Nous avons fait un tour d'horizon sur cette technique avec une focalisation particulière sur les schémas de routage qui sont exploités pour mener cette agrégation. Une exploration détaillée sur les travaux connexes à ces schémas a été présentée. En outre, d'autres approches qui s'intéressent au problème de la délivrance en délai borné des données agrégées ont été aussi exposées. Après ce tour d'horizon, nous présentons dans

le chapitre suivant les algorithmes de colonies de fourmis, notre solution multi-agents à appliquer à ce problème d'établissement de la structure de routage pour agrégation de données, et leur utilisation dans le contexte des réseaux de capteurs sans fil.

Chapitre 4

Algorithmes de colonies de fourmis

4.1 Introduction

Les algorithmes de colonies de fourmis forment une classe des métaheurisqtiques qui mettent en opération une colonie de fourmis artificielles afin de trouver de bons résultats à des problèmes d'optimisation difficiles. Ils ont été inspirés du comportement collectif des fourmis recherchant un chemin entre leur colonie et une source de nourriture. Tous les mécanismes caractérisant ce processus d'exploitation des ressources alimentaires par les fourmis ont été remaniés pour donner naissance à un cadre général dédié à la conception des systèmes multi-agents (SMAs) distribués, robustes et adaptatifs pouvant être appliqués à la résolution d'une classe particulière de problèmes s'intéressant à la détermination du plus court chemin [120]. Ces nouveaux algorithmes développés ont suscité depuis leur première application au problème de voyageur de commerce une grande attention parmi les scientifiques. En effet, beaucoup de problèmes d'optimisation peuvent être modélisés sous forme de graphes et formulés sous la forme d'un problème du plus court chemin [121]. Cela est accompagné de la nature distribuée et adaptative de l'approche a rendu ces algorithmes de colonies de fourmis applicables à un large éventail d'applications, ce qui a contribué à leur popularité. Dans ce chapitre, nous détaillons l'origine de ces algorithmes, l'analogie des comportements des fourmis avec les réseaux de communication et l'ingénierie de ces comportements pour permettre leur application aux problèmes techniques en général et aux réseaux de communication en particulier. Ce chapitre détaille aussi un certain ensemble de travaux connexes à l'application de ces algorithmes aux problèmes d'agrégation de données et de routage en temps réel dans les réseaux de capteurs sans fil. Mais avant tout cela, nous présentons au début de ce chapitre la notion d'auto-organisation qui caractérise ces colonies de fourmis et qui est considérée comme le principal paradigme de contrôle des futures réseaux de communication. En outre, le domaine émergeant de la mise en réseau bio-inspirée est également présenté.

4.2 L'auto-organisation comme paradigme de contrôle des systèmes massivement distribués

Les objets coopératifs et avec les différents domaines d'application qui peuvent envisager favorisent l'inclusion d'un nombre important de dispositifs travaillant ensemble afin d'assurer un objectif commun. Ceci impose le besoin à des algorithmes et des mécanismes scalables pouvant évoluer correctement suivant le nombre de nœuds à considérer, et cela, d'une manière à maintenir le bon fonctionnement de l'application. Cette nécessité accompagnée de ressources limitées de ces objets requière à son tour des algorithmes et des comportements locaux simples exécutés par chacun de ces objets et se basant seulement sur des informations locales. Ces objets et par le biais de leurs interactions résultent l'émergence de la solution globale qui reflète la bonne mise en marche de cette coordination.

Ce processus dans lequel la fonctionnalité globale émerge uniquement à partir de nombreuses interactions entre les différents éléments du réseau ayant chacun son propre contrôle, et cela, sans se référencer à aucun contrôle externe ou centralisé est connu sous le terme *auto-organisation* [19]. La figure (4.1), tirée à partir de [19], illustre les principales propriétés d'un système auto-organisé. Comme nous pouvons le constater, un ensemble de systèmes individuels sont déployés dans certain environnement avec lequel ils peuvent interagir. Chacun de ces systèmes maintient certaines informations locales pertinentes à son environnement et à son voisinage immédiat. En se basant sur des règles simples, ils participent dans la réalisation d'un objectif global sans se référencer à une entité externe pouvant être consciente de cet objectif. Dans le contexte des réseaux informatiques, l'auto-organisation peut être vue comme étant les interactions entre les différents nœuds du réseau qui mènent vers des effets globaux et visibles comme le transport des messages d'un nœud source à un autre de destination. L'objectif ici consiste à réduire le recours à un état global du réseau en se basant seulement sur des états locaux afin de réaliser les objectifs souhaités.

De manière générale, le comportement émergeant qu'un système auto-organisé manifeste est rendu possible, principalement, suite aux trois méthodes basiques suivantes [19, 122, 123] :

1. **Rétroactions positive et négative :** une rétroaction peut être définie comme étant l'action en retour d'un effet sur sa cause. Typiquement, les sorties du système et donc ses actions influencent, en retour, son comportement future. Ce genre de rétroactions sont principalement utilisées afin qu'un système donné peut mettre à jour son état local suivant l'état courant de son environnement. En fait, c'est à travers les rétroactions du système sur ses propres actions que ce système peut s'adapter aux différents changements qui arrivent dans son environnement

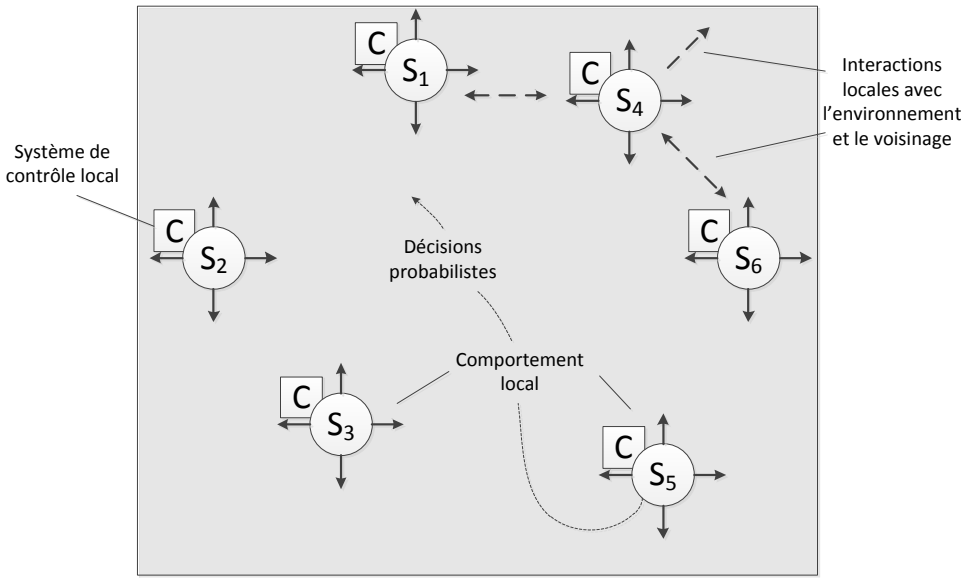


FIGURE 4.1 – Les propriétés d'un système auto-organisé.

immédiat. Deux types de rétroactions sont à distinguer. La première, une rétroaction positive, qui favorise les changements qui arrivent dans le système, et donc, qui fournit des capacités d'amplification. En se basant sur ce genre de rétroactions, le système peut répondre en temps opportun et de manière efficace à ces changements. Cependant, une rétroaction positive excessive tend à affecter la stabilité du système. C'est pour cette raison qu'une rétroaction négative est nécessaire. Cette dernière est utilisée principalement afin de contrôler cette rétroaction positive via l'inhibition de cette amplification. Une rétroaction négative peut être créée via le développement de certaines règles qui tendent à garder l'état du système dans un intervalle bien déterminé. Sinon, des contraintes physiques peuvent simplement et implicitement jouer ce rôle. L'utilisation conjointe d'une rétroaction positive avec une autre négative doit être bien coordonnée. Sinon, le système va tendre à ne rien faire, à osciller entre multiple états ou à sur-réagir. Dans un réseau de capteurs sans fil, ces rétroactions sont assurées par l'observation et l'évaluation des différents paramètres du système. Cela peut être réalisé via l'exploitation des capteurs propres aux nœuds pour observer et mesurer les conditions environnementales. En outres, les informations échangées entre nœuds voisins à la détection de certaines changements peuvent également être utilisées. En fait, c'est à travers ces informations de voisinage accompagnées des informations pertinentes aux états locaux des nœuds que ces derniers peuvent contrôler leurs comportements et s'adapter aux différents changements.

2. **Interaction entre voisins et avec l'environnement :** comme on vient de le voir, dans un système auto-organisé, l'organisation et donc l'objectif désiré

émerge suite aux différentes interactions entre ses composants. Cette interaction peut être entreprise de manière directe via des signaux comme le cas d'une communication unicast dans un réseau de communication. En outre, elle peut également être menée de manière indirecte via l'environnement en affectant ce dernier via le dépôt d'empreintes ou simplement en modifiant ses conditions. Les autres systèmes détectent ces changements et réagissent en retour. Cette façon à se communiquer à travers la modification de l'environnement est communément connue sous le nom *stigmergie*. Dans un réseau de capteurs sans fil, ces interactions sont primordiale au bon fonctionnement du réseau. À titre d'exemple, un nœud capteur doit communiquer avec ses voisins immédiat (i.e. ceux qui sont dans sa portée de communication) via l'échange des messages RTS/CTS afin de pouvoir gérer l'accès au canal sans fil.

3. **Techniques probabilistes :** ces techniques sont présentes dans tous les systèmes auto-organisés. Elles sont employées principalement afin d'organiser le comportement local d'un système individuel ou à simplement définir les paramètres connexes à d'autres algorithmes déterministes. L'exploitation de ces techniques dans un réseau de communication peut être très bénéfique. En fait, elles permettent de mettre en marche certaines fonctions sans le recours à des informations particulières de l'état du réseau. Cela contribue, à son tour, à ne pas s'inquiéter de toute opération de maintenance de cet état. Cependant et en contre-partie, le comportement du réseau tend à être moins prévisible. Plusieurs approches se basant sur ces techniques et appliquées aux réseaux de capteurs peuvent être citées. Par exemple, le routage probabiliste tel que le gossiping fait partie de ces approches. En plus, la sélection du chemin approprié dans un routage à multi-chemins ou le retardement des retransmissions à l'arrivée d'une collision au niveau de la couche MAC sont de bons exemples.

De manière générale, ces techniques doivent être utilisées de manière conjointe pour que le comportement désiré du système soit atteint. Par exemple, dans un système donné, certaines méthodes probabilistes peuvent assurer une bonne configuration initiale, les boucles de rétroactions permettent son adaptation appropriée et l'interaction entre ses composants contribue à l'échange d'information d'état et de contrôle.

Même si cette auto-organisation présente divers avantages, certaines limites sont à noter. Parmi ces limites, l'imprévisibilité ascendante du comportement du système à l'augmentation de ses constituants est la plus importante. En fait, le contrôle d'un système donné devient difficile lorsque le nombre de ses composants croit. À titre d'exemple, les solutions classiques d'administration des réseaux n'évoluent pas bien car l'acquisition des informations nécessaires à leur opération et qui sont pertinentes à l'état du réseau devient difficile.

4.3 La mise en réseau bio-inspirée

Depuis toujours, la nature a constitué une vraie source d'inspiration à partir de laquelle la science technique et ingénierie en imite afin de résoudre des problèmes ayant une certaine analogie avec les systèmes naturels et les processus et stratégies qui les régissent. L'auto-organisation que nous venons d'exposer est l'un des exemples de cette inspiration où ses principes ont été principalement constatés dans la nature pour être, à la suite, étudiés et appliqués à des problèmes techniques.

Dans ces dernières décennies, les réseaux informatiques ont connu beaucoup d'avancées, de façon à ce que, la vision de Mark Weiser de la future génération de l'informatique commence à être concrétisée. En fait, cette future génération d'informatique, dite informatique ubiquitaire, envisage des réseaux de communication omniprésents qui connectent presque tous les objets qui nous entourent sans se limiter au cadre traditionnel [124]. Ces réseaux doivent être en mesure d'assurer une multitude de services n'importe où et n'importe quand avec un accès facile et transparent. Même si cette mutation de l'informatique contribue à l'amélioration de notre qualité de vie, elle émerge beaucoup de contraintes et défis techniques qui doivent être résolus avant sa réalisation concrète. En termes de communication, les protocoles et services déployés au-dessus des réseaux doivent être en mesure de supporter la mise en réseau d'un large nombre de dispositifs hétérogènes qui génère un trafic de données très important. En outre, ces réseaux seront caractérisés par une grande dynamique en termes de la mobilité de leurs nœuds, du type du trafic exigé, de la demande en bande passante ainsi que d'autres. Cette nature dynamique requière la capacité du réseau à s'adapter à tout changement potentiel des conditions de son environnement. Outre cette dynamicité, ces réseaux seront constitués de nœuds ayant des ressources limitées qui peuvent être déployés de manière ad hoc sans le recours à une infrastructure particulière et sans se baser sur aucune entité centrale. Ils doivent être également tolérants à toute panne potentielle.

Les systèmes naturelles et biologiques montrent certains comportements intéressants pour faire face à des problèmes qui sont en analogie avec ces nouveaux problèmes émergents liés à cette future génération de l'informatique et des réseaux de communication. En observant de près ces systèmes, nous sommes impressionnés des diverses potentialités qu'ils présentent. En fait, ils peuvent s'adapter facilement à la dynamique de leurs environnements comme ce qui peut être constaté dans les systèmes immunitaires des mammifères qui peuvent détecter les variations des circonstances de leurs environnements ou les déviations par rapport aux patterns attendus et réagir en conséquence. Des systèmes immunitaires artificiels qui imitent de leurs équivalents naturels ont été déjà proposés et appliqués à divers problèmes. Dans le contexte des réseaux de communication et à titre d'exemples, l'algorithme DNRS [125] exploite ces systèmes afin de sélectionner les nœuds capteurs appropriés responsables de l'émission des don-

nées perçues au nœud puits ainsi que leurs taux d'émission. Le travail proposé dans [126] se base également sur ces systèmes et propose une approche de détection des mauvais comportements des nœuds dans un réseaux ad hoc mobile.

Par ailleurs, la dissémination des informations dans des réseaux à grande échelle peut être inspirée de la façon dont une épidémie se propage [127, 128]. Des systèmes massivement distribués peuvent être synchronisés en se basant sur les principes de synchronisation des lampyridés [129, 130]. En plus, des techniques de communication pour des réseaux exhibant un certain niveau d'hétérogénéité peuvent être développées en s'inspirant du phénomène d'homéostasie où l'état du système (e.g. le corps humain) est maintenu à un niveau stable à travers la collaboration de divers sous-systèmes hétérogènes [131].

En outre, certains systèmes biologiques sont capables d'accomplir des tâches complexes en se basant seulement sur peu de règles simples qui n'exigent aucune intelligence centrale. Par exemple, l'intelligence collective des insectes sociaux pour faire survivre et croître leurs colonies émerge seulement à partir des comportements locaux de leurs individus indépendamment de toute règle globale. Cette intelligence en essaim constitue une vraie source d'inspiration pour le développement d'algorithmes de routage efficaces dans des réseaux de communication. L'analogie entre ces derniers et cette intelligence collective des insectes sociaux est détaillée dans la prochaine section en étudiant le comportement des fourmis et leur adaptation au problèmes techniques tels que les réseaux de communication.

Ce que nous venons de citer représente quelques exemples d'un domaine vaste et récent qui peut contribuer énormément aux développement des futures réseaux de communication. En fait, la nature reste une source d'inspiration importante afin d'aboutir à des techniques de communication innovantes. Cependant, malgré tous les travaux de recherche conduits dans ce sens, ce domaine de mise en réseau bio-inspirée n'est pas assez mature et davantage de recherches restent à mener.

4.4 Les fourmis dans la nature

Les algorithmes de colonies de fourmis sont nés suite à l'observation de la façon dont les fourmis procèdent durant l'exploitation des ressources alimentaires [132]. Une fourmi, en se déplaçant, marque son chemin par une substance chimique volatile appelée phéromone. Cette dernière est utilisée ensuite par les futures fourmis, étant donné qu'elles peuvent la sentir, et les chemins ayant une forte intensité de cette substance vont avoir une grande probabilité d'être choisis. Ce principe de dépôt et de sensation de phéromone constitue le principe de base qui permet à ces fourmis de se déplacer entre leur nid et la source de nourriture.

La chose impressionnante dans leur processus de recherche est qu'elles peuvent

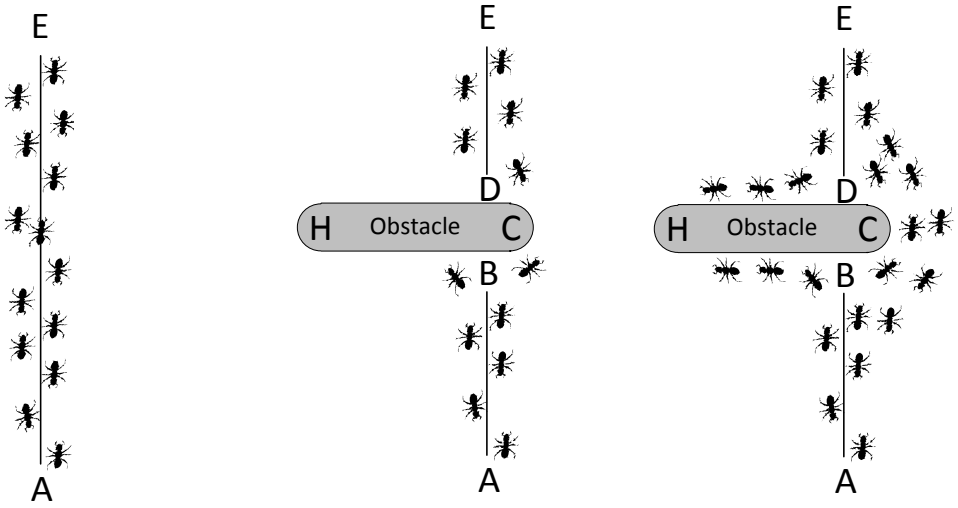


FIGURE 4.2 – Comportement des fourmis dans la présence d'obstacle. (4.2a) Les fourmis trouvent et empruntent un chemin entre les deux points A et E. (4.2b) À la présence d'obstacle, les fourmis ont la même probabilité d'emprunter l'un des deux chemins autour de cet obstacle. (4.2c) Le plus court chemin est sollicité par plus de fourmis par rapport au deuxième chemin.

trouver le plus court chemin en se basant seulement sur des comportements simples et des communications indirectes médiées par l'environnement (stigmergie). Pour illustrer ce phénomène, nous reprendrons l'exemple de Dorigo dans son article [132] qui a donné naissance au premier algorithme imitant des comportements des fourmis et visant à résoudre le problème du voyageur de commerce. Considérons l'exemple de la figure (4.2) où un ensemble de fourmis sont en déplacement entre deux points A (e.g. la source de nourriture) et E (e.g. leur nid). À la présence d'obstacle comme le montre la figure (4.2b), les fourmis qui sont au niveau du point B et qui arrivent de A doivent décider de la direction à emprunter (i.e. à gauche ou à droite de l'obstacle). La même chose pour les fourmis qui sont au niveau du point D et qui arrivent de E. À ce moment-là et comme aucun dépôt de phéromone a été réalisé précédemment sur les deux chemins, la première fourmi qui est au niveau de B a la même probabilité à tourner à gauche ou à droite. Cependant, étant donné que le portion BCD est plus courte que la portion BHD, la première fourmi qui emprunte BCD arrive à D avant la première fourmi qui emprunte BHD. Comme résultat, le chemin BCD va connaitre une forte intensité de phéromone à cause de l'ensemble des fourmis qui, par chance, ont décidé d'emprunter le chemin BCD pour aller à E ou d'emprunter le chemin DCB pour aller à A. Comme conséquence à cette forte intensité de phéromone, le chemin BCD va attirer davantage de fourmis qui renforcent à leur tour cette intensité jusqu'à ce que presque toutes les fourmis empruntent ce plus court chemin. Par ailleurs, le long chemin finira par attirer

moins de fourmis étant donné que la phéromone s'évapore au fil du temps.

En fait, même si une seule fourmi peut trouver un chemin quelconque entre son nid et une source de nourriture, c'est seulement à travers la présence simultanée de plusieurs fourmis avec leur travail collectif que le plus court chemin peut être déterminé. Cette synergie entre fourmis via leurs comportements simples qui résulte la résolution d'un tel problème, assez complexe, est un bon exemple de système auto-organisé. On peut constater ici que ce choix du meilleur chemin est rendu possible suite à une rétroaction positive où le dépôt initial de phéromone attire d'autres fourmis qui déposent, à leur tour, davantage de phéromone afin d'attirer davantage de fourmis. En outre, une rétroaction négative qui est assurée par l'évaporation de la phéromone contribue à la réussite de ce phénomène en rendant les mauvais chemins moins attirants. Un autre facteur important qui influence la façon dont les fourmis réussissent dans cette tâche d'exploitation des ressources alimentaires consiste en leur recherche probabiliste [133]. En réalité, les fourmis explorent leur environnement aléatoirement afin de trouver de la nourriture. Cette exploration est ensuite biaisée par l'intensité de la phéromone déposée le long des différents chemins entre la source de nourriture et le nid des fourmis. Une petite déviation durant cette exploration peut avoir une grande influence sur les futures évènements. Ce comportement aléatoire permet d'un côté, l'émergence de nouvelles solutions, et d'un autre côté, il guide ce processus de recherche pour s'adapter aux futures changements de l'environnement. Dernièrement, cette capacité des fourmis à déterminer le plus court chemin n'est rendu possible que via le phénomène de stigmergie, le moyen de communication des fourmis via le dépôt et la sensation de la phéromone dans leur environnement.

Cette force à faire émerger de bonnes solutions à partir des comportements auto-organisés et coopératifs d'agents (fourmis) simples a motivé les scientifiques à imiter ce phénomène afin de résoudre des problèmes ayant une certaine analogie avec ces systèmes naturels. Parmi les domaines qui ont une grande analogie avec ces systèmes et comme nous allons voir dans ce qui suit est celui du routage dans les réseaux informatiques.

4.5 Algorithmes de colonies de fourmis et problème du voyageur de commerce

Avant de présenter le principe général d'un algorithme de colonies de fourmis et son application au problème du routage dans les réseaux de communication, il est intéressant de présenter la première implémentation d'un algorithme de colonies de fourmis connu sous le nom *Ant System (AS)* [132]. Cet algorithme a été appliqué au problème du voyageur de commerce qui consiste à trouver le plus court chemin reliant

n villes données tout en veillant à-ce-que chaque ville ne devant être visitée qu'une seule fois.

4.5.1 L'algorithme *Ant System (AS)*

Le problème est modélisé sous forme d'un graphe $G = (V, E)$ où V représente l'ensemble des sommets qui correspondent aux différentes villes, avec $|V| = n$, et E représente l'ensemble d'arêtes qui correspondent aux trajets séparent ces villes. Chaque arête $e_{i,j}$ est associé avec une valeur $d_{i,j}$ qui correspond à la distance Euclidienne entre les deux villes i et j. La résolution de ce problème est assurée par le déploiement d'une colonie de fourmis artificielle sur ce graphe. Chaque fourmi k de cette colonie démarre d'une ville aléatoire et parcourt le graphe, en se déplaçant d'une ville à une autre, jusqu'à ce qu'elle fait un tour complet T^k. Si on considère que cette colonie comporte m fourmis, alors, lorsque ces m fourmis accomplissent un tour, on dit qu'une itération t de l'algorithme a été effectuée. Pour réussir cette recherche, chaque fourmi de cette colonie est modélisée sous forme d'un agent simple caractérisé par les comportements suivants :

1. Une fourmi k qui est au niveau d'une ville donnée i, choisit la prochaine ville, j, à laquelle elle est censée aller, et comme le montre la figure (4.3), sur la base d'une certaine probabilité $p^k_{i,j}$qui dépend d'une certaine valeur $\eta_{i,j} = \frac{1}{d_{i,j}}$, dite visibilité, et une autre valeur $\tau_{i,j}$ représentant l'intensité de la phéromone sur l'arête $e_{i,j}$;
2. Afin d'éviter la visite d'une même ville plus qu'une fois, chaque fourmi k est dotée d'une petite mémoire afin d'enregistrer la liste des villes déjà visitées durant l'itération courante. Cette liste définit les mouvements possibles, J^k_i, de cette fourmi lorsqu'elle est au niveau d'une ville donnée i.
3. Lorsque cette fourmi k accomplit un tour, elle dépose une certaine quantité de phéromone sur chaque arête $e_{i,j}$ visitée qui dépend de la longueur du tour parcouru.

La règle de déplacement d'une fourmi k qui est au niveau d'une ville i est donnée par la formule suivante :

$$p^k_{i,j}(t) = \begin{cases} \frac{[\tau_{i,j}(t)]^\alpha \cdot [\eta_{i,j}]^\beta}{\sum_{l \in J^k_i} \left[\tau_{i,l}(t)\right]^\alpha \cdot \left[\eta_{i,l}\right]^\beta} & si\ j \in J^k_i \\ 0 & sinon \end{cases} \tag{4.1}$$

Dans l'équation (4.1), α et β sont deux paramètres qui contrôlent, respectivement, l'importance relative à l'intensité de la phéromone, $\tau_{i,j}(t)$, et la visibilité, $\eta_{i,j}$. Suivant cette règle, le choix de la prochaine ville à visiter dépend d'un compromis entre la valeur de visibilité qui favorise la sélection des villes proches de i, et de l'intensité de

phéromone sur chaque arête séparent i d'une ville voisine qui reflète l'intensité du trafic, et donc la désirabilité que connait chaque arête. Il est à noter que la sélection des valeurs de α et β est cruciale. En fait, il est important d'assurer un certain compromis entre ces deux valeurs, jouant sur les comportements de diversification et d'intensification, afin d'éviter la convergence rapide vers un tour quelconque.

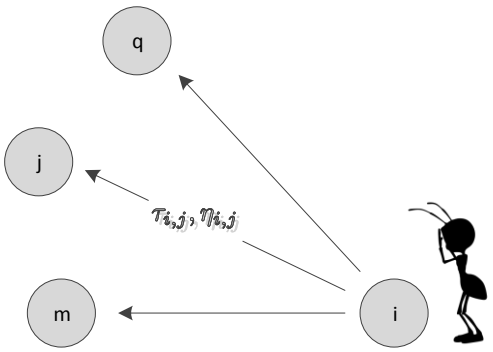


FIGURE 4.3 – Une fourmi qui est au niveau de la ville i choisit sa prochaine ville en fonction de l'intensité de phéromone et la visibilité caractérisant chaque arête connectant i avec une autre ville j non visitée.

À la fin d'une itération, chaque fourmi k calcule la quantité de phéromone $\Delta\tau^k_{i,j}(t)$ qui sera utilisée pour mettre à jour l'intensité de phéromone sur chaque arête $e_{i,j}$. Cette quantité est calculée comme suit :

$$\Delta\tau^k_{i,j}(t) = \begin{cases} \frac{Q}{L^k(t)} & si\ e_{i,j} \in T^k(t) \\ 0 & sinon \end{cases} \tag{4.2}$$

Dans cette équation (4.2), $L^k(t)$ correspond à la longueur du tour parcouru par la fourmi k à l'itération t. Et Q est une constante.

Pour que l'algorithme ne converge pas vers des solutions sous-optimales, il est nécessaire de permettre au système d'oublier les mauvaises solutions via un processus d'évaporation des pistes de phéromone. Pour réaliser cela, le dépôt de la phéromone est contrebalancé par une décroissance constante de l'intensité de phéromone sur chaque arête. De ce fait, la règle de mise à jour de l'intensité de phéromone est la suivante :

$$\tau_{i,j}(t+1) = (1-\rho)\cdot\tau_{i,j}(t) + \Delta\tau_{i,j}(t) \tag{4.3}$$

Où $\Delta\tau_{i,j}(t) = \sum_{k=1}^{m}\Delta\tau^k_{i,j}(t)$. Dans cette équation (4.3), ρ correspond à la fraction de phéromone qui est évaporée. Le pseudo-code 4.1 , adapté de [59], résume le principe de l'algorithme AS.

Après cette première implémentation d'un algorithme de colonies de fourmis, d'autres algorithmes qui améliorent davantage les performances de *AS* ont été proposés. Parmi ces algorithmes, *Min-Max Ant System* et *Ant-Colony-System* que nous présentons dans

Algorithme 4.1 Principe de l'algorithme AS.

1: **Pour** $t = 1, ..., t_{max}$
2: **Pour** chaque fourmi $k = 1, ..., m$
3: choisir une ville au hasard
4: **Pour** chaque ville non visitée i
5: choisir une ville j suivant la règle (4.1)
6: **Fin Pour**
7: **Fin Pour**
8: mettre à jour l'intensité de phéromone sur chaque arête visitée suivant la règle (4.3)
9: **Fin Pour**

la suite de cette section sont les plus connus.

4.5.2 L'algorithme *Min-Max AS*

Cette variation de l'algorithme AS, notée *MMAS,* introduit quatre modifications [134, 135] :

1. Seule la fourmi qui trouve le meilleur chemin durant l'itération courante de recherche ou depuis le début de cette recherche est autorisée à déposer de la phéromone.
2. Étant donné que cette façon à faire va tendre vers une situation de stagnation (i.e. toutes les fourmis sont attirées par le même chemin), les intensités possibles de phéromone sont bornées dans l'intervalle $[\tau_{min},\ \tau_{max}]$.
3. La phéromone sur chaque arc $e_{i,j}$ est initialisé à la limite supérieur τ_{max}. Cela, accompagné d'un faible taux d'évaporation, favorise une grande exploration de tours au début de la recherche.
4. Afin d'augmenter la probabilité de sélection des chemins qui ont une faible probabilité d'être choisis, la phéromone est réinitialisée d'un temps à un autre. Cette réinitialisation s'effectue principalement lorsque le système se rapproche d'une stagnation ou lorsque aucune amélioration concernant le meilleur chemin déjà trouvé n'est constatée.

Les résultats d'expérimentation ont montré que l'algorithme *MMAS* passe par une longue phase d'exploration étant donné que la phéromone est initialisée par t_{max} accompagné d'un faible taux d'évaporation. Durant cette phase initiale, les solutions obtenues sont très mauvaises par rapport à celles trouvées par *AS*. Cependant, après un certain nombre d'itérations, l'algorithme va tendre vers d'autres meilleures solutions et produit une solution finale qui est meilleure par rapport à celle trouvée par l'algorithme *AS*.

4.5.3 L'algorithme *Ant-Colony-System*

Ant-Colony-System (ACS) introduit trois modifications [136, 137] :

1. Une nouvelle règle de déplacement est proposée.
2. Les arêtes connexes, au meilleur chemin trouvé depuis le début de la recherche, sont les seules concernées par le dépôt et l'évaporation de la phéromone.
3. À chaque déplacement d'une fourmi d'une ville i à une autre j, elle enlève une certaine quantité de phéromone sur l'arête $e_{i,j}$ afin de favoriser la sélection des autres chemins.

Concernant la règle de déplacement, une fourmi k qui est au niveau d'une ville i, choisit sa prochaine ville j sur la base d'un paramètre aléatoire q et un autre paramètre q_0. Ce dernier est un paramètre qui est compris entre 0 et 1 ($0 \leq q \leq 1$). Si $q \leq q_0$, alors le système tend vers une intensification. Sinon (i.e. $q > q_0$), le système va tendre vers une exploration. La règle de transition adoptée est comme suit :

$$j = \begin{cases} \arg\max_{l \in J_i^k} [\tau_{i,l}] \cdot [\eta_{i,l}]^\beta & si\ q \leq q_0 \\ J & sinon \end{cases} \tag{4.4}$$

Dans cette équation (4.4), J est la ville qui est sélectionnée suivant les probabilités calculées par la formule (4.1) (avec $\alpha = 1$). Comme nous pouvons constater, l'algorithme *ACS* tire plus davantage de l'expérience que connait les fourmis durant leur recherche.

Concernant la mise à jour de la phéromone, *ACS* utilise deux règles : une globale et une autre locale. Pour ce qui est de la règle globale, la fourmi qui trouve le meilleur chemin depuis le début de la recherche est la seule autorisée à mettre à jour de la phéromone après chaque itération. Elle la met à jour de la façon suivante :

$$\tau_{i,j}(t+1) = (1-\rho) \cdot \tau_{i,j}(t) + \rho \cdot \Delta\tau_{i,j}^{bs}, \qquad \forall e_{i,j} \in T^{bs} \tag{4.5}$$

Dans cette équation (4.5), $\Delta\tau_{i,j}^{bs} = 1/L^{bs}$ où L^{bs} représente la longueur du meilleur chemin T^{bs} trouvé depuis le début de la recherche.

Outre cette mise à jour globale, chaque fourmi qui se déplace d'une ville i à une autre j diminue l'intensité de la phéromone sur l'arête $e_{i,j}$ suivant cette règle de mise à jour locale :

$$\tau_{i,j}(t+1) = (1-\xi) \cdot \tau_{i,j}(t) + \xi \cdot \tau_0 \tag{4.6}$$

Dans cette équation (4.6), t_0 et ξ sont deux paramètres. La valeur de t_0 correspond à la valeur initiale avec laquelle l'intensité de la phéromone est initialisée. Le but derrière cette règle est de favoriser la diversification par la prise en compte des chemins peu ou

non explorés.

Les résultats d'expérimentation ont montré que l'algorithme *ACS,* et par rapport à *AS* et *MMAS,* arrive à trouver de bons résultats dans peu de temps. La solution finale trouvée est meilleure par rapport à celle trouvée par *AS* mais pas à celle trouvée par *MMAS*. Cependant, cette meilleure solution produite par *MMAS* émerge après un grand nombre d'itérations durant lesquelles *ACS* a déjà pu trouver une solution acceptable.

4.6 Formalisme général d'un algorithme de colonies de fourmis

Il est bien de connaitre le formalisme général d'un algorithme de colonies de fourmis afin de pouvoir l'appliquer à un problème donné. Pour cela, nous présentons dans cette section une telle formalisation qui comprend une représentation du problème, un comportement de base d'une fourmi et une organisation générale d'un algorithme de colonies de fourmis [120, 138].

4.6.1 Représentation du problème

Dans un algorithme de colonies de fourmis, une fourmi représente une procédure de construction stochastique de solution via l'ajout de composants à une solution partielle. Donc, un algorithme de colonies de fourmis peut être appliqué à tout problème auquel une heuristique constructive peut être définie.

Les algorithmes de colonies de fourmis constituent une métaheuristique qui peut être appliquée aux problèmes d'optimisation combinatoire. Un tel problème peut être défini par le triplet : (C, Ω, J) où C est un ensemble fini de composants, $\{c_0, c_1, ..., c_n\}$, utilisé pour définir toute solution candidate parmi un ensemble S de solutions. Ω représente l'ensemble des relations mathématiques définissant les contraintes appliquées aux éléments de l'ensemble C afin de produire une solution faisable $\sigma \in S$. Et J est une fonction objective qui assigne à chacune de ces solutions un coût. La résolution de ce problème consiste à trouver l'optimum globale satisfaisant ces contraintes. Si nous faisons l'extrapolation sur le problème du routage dans les réseaux de communication, l'ensemble C va correspondre à l'ensemble de tous les nœuds du réseau. L'ensemble S englobe tous les chemins possibles connectant chaque paire (s, d) de nœuds source et destination. Et Ω définit, par exemple, l'obligation de non présence des boucles. Le coût d'un chemin peut correspondre par exemple à la somme des coûts (e.g. la durée de transmission) associés aux transmissions entre toute paire de nœuds adjacents i et j faisant partie de ce chemin.

Suivant cette formulation, les fourmis parcourent le graphe $G = (C, L)$, appelé *graphe de construction*, afin de trouver une solution au problème posé. Dans ce graphe G, C représente l'ensemble de composants et L correspond aux connexions entre ces composants. Les contraintes du problème sont implémentées directement dans les règles de déplacements des fourmis (e.g. en empêchant les déplacements qui violent les contraintes).

4.6.2 Comportement basique d'une fourmi

Une fourmi k dans un algorithme de colonies de fourmis se comporte de la façon suivante :

1. Elle se déplace sur le graphe $G = (C, L)$ afin de trouver des solutions optimales au problème posé.
2. Elle possède une mémoire M^k qu'elle l'utilise pour être consciente du trajet déjà parcouru. Cette mémoire peut être utilisée pour : (1) construire des solutions faisables (i.e. des solutions obéissant aux contraintes) ; (2) calculer la valeur heuristique η ; (3) évaluer la solution trouvée ; et (4) pouvoir retracer le chemin inverse.
3. Elle démarre à partir d'un état initial et termine sa recherche suivant une ou plusieurs conditions de terminaison.
4. Lorsqu'elle est au niveau d'un nœud i, elle sélectionne son prochain nœud à visiter j parmi l'ensemble N_i^k représentant les voisins du nœud i dans le graphe G. Par ailleurs, il est nécessaire qu'elle s'assure qu'elle est autorisée à continuer sa recherche (i.e. il n'y a aucune condition d'arrêt qu'elle l'oblige à s'arrêter), et que j n'a pas déjà été visité (en se basant sur sa mémoire).
5. Elle se déplace d'un nœud i à un autre j sur la base d'une règle de décision probabiliste fonction des : (1) intensités de phéromone et des valeurs heuristiques qui peuvent être associées soit avec les voisins de i : N_i^k (on note : τ_j et η_j) ou avec les arêtes qui connectent le nœud i avec ses voisins (on note : $\tau_{i,j}$ et $\eta_{i,j}$) ; (2) sa mémoire M^k ; et (3) les contraintes du problème.
6. Il est possible qu'elle mette à jour l'intensité de phéromone lorsqu'elle se déplace d'un nœud à un autre durant son trajet aller (comme le cas de l'algorithme ACS).
7. Une fois qu'elle finalise la construction d'une solution, elle prend le chemin inverse et met à jour l'intensité de phéromone sur les éléments correspondant au nœuds ou arêtes de cette solution construite.

4.6.3 Organisation générale d'un algorithme de colonies de fourmis

D'une manière générale, un algorithme de colonies de fourmis incorpore les trois actions suivantes :

1. **La construction de solutions et leur évaluation :** un algorithme de colonies de fourmis est basé sur la recherche et la construction répétée des solutions pertinentes au problème considéré par des agents fourmis. Ces solutions sont à chaque fois évaluées et les résultats sont exploités ensuite afin de mettre à jour certaines variables jouant le même rôle que joue la phéromone dans une société de fourmis.
2. **La mise à jour de la phéromone :** durant ce processus de mise à jour, les agents fourmis renforcent davantage et de manière proportionnelle à la qualité de la solution trouvée les variables de phéromone pertinentes aux entités faisant partie de cette solution trouvée. Cela biaise à chaque fois ce processus de recherche de solutions vers la partie de l'espace de recherche pouvant contenir la solution optimale ou sous-optimale. En contre-partie, ces variables de phéromone sont à chaque fois diminuées de manière similaire à l'évaporation de la phéromone dans le cadre biologique. Cette diminution évite une convergence rapide vers des parties sous-optimales de l'espace de recherche ce qui favorise une exploration diversifiée de cet espace.
3. **D'autres actions optionnelles :** qui se réfèrent aux activités spécifiques qui n'ont aucune relation avec le cadre biologique de l'algorithme. Par exemple, pour décider du nombre de fourmis autorisées à mettre à jour les variables de phéromone.

4.7 Algorithmes de colonies de fourmis et problème du routage dans les réseaux de communication

Comme nous venons de le voir, les fourmis et via leur travail collectif peuvent résoudre des problèmes de routage. Afin de faire survivre leur colonie, les fourmis doivent explorer leur environnement à la recherche de source de nourriture. À la découverte de cette source, elles doivent établir certains chemins qui peuvent être exploités pour que les fourmis puissent se déplacer entre leur nid et cette source. Ces chemins émergent suite à l'évaluation implicite de multiples trajets et la communication de leurs caractéristiques aux autres fourmis via le dépôt de la phéromone. En résumant, nous pouvons dire que ce processus de collection de la nourriture requière l'exploration distribuée, la découverte, la détermination et l'utilisation de chemins de routage dans un environne-

ment dynamique. Ces caractéristiques sont en fait, les mêmes fonctions nécessaires au routage de données dans un réseau de communication.

L'approche commune à tous les algorithmes de colonies de fourmis qui sont appliqués au problème de routage consiste à utiliser des fourmis artificielles représentées par des agents mobiles. Ces derniers correspondent à des paquets de contrôle intelligents. Ces agents fourmis sont libérés de manière concurrente et indépendante à partir des nœuds du réseau pour explorer ce dernier et établir de bons chemins, sur la base de certaines métriques, entre leurs nœuds sources et ceux de destination. Chaque fourmi, $F_{s\to d}$ (*Forward Ant*), qui démarre d'une source s, se déplace d'un nœud à un autre suivant une certaine règle basée sur l'information de phéromone et potentiellement, sur une autre information locale (e.g. dans un réseau de capteur sans fil, cette information peut correspondre à la puissance de transmission nécessaire à la communication entre deux nœuds voisins i et j). Pour que cette fourmi puisse considérer cette information de phéromone durant ses déplacements et comme le montre la figure (4.4), chaque nœud i est doté d'une table de phéromones. Cette table comporte pour chaque destination d, un vecteur d'entrées de valeurs réelles avec une entrée $\tau_{i,j}^{d}$ pour chaque voisin j du nœud i. Ces entrées, qui coïncident aux variables de phéromones, représentent les désirabilités à choisir chaque nœud voisin j de i comme prochain saut pour aller à d. Lorsqu'une fourmi arrive à son nœud de destination, elle évalue la qualité du trajet parcouru et se convertit en une autre, appelée $F_{d\to s}$ *(BackwardAnt).* Cette dernière prend le chemin inverse de sa fourmi $F_{s\to d}$ correspondante pour se retourner à son nœud source tout en mettant à jour les tables de phéromones des nœuds visités.

Destination \ Voisin	j	m	k
m	$\tau_{i,j}^{m}$	$\tau_{i,m}^{m}$	$\tau_{i,k}^{m}$
n	$\tau_{i,j}^{n}$	$\tau_{i,m}^{n}$	$\tau_{i,k}^{n}$
d	$\tau_{i,j}^{d}$	$\tau_{i,m}^{d}$	$\tau_{i,k}^{d}$
j	$\tau_{i,j}^{j}$	$\tau_{i,m}^{j}$	$\tau_{i,k}^{j}$
k	$\tau_{i,j}^{k}$	$\tau_{i,m}^{k}$	$\tau_{i,k}^{k}$
l	$\tau_{i,j}^{l}$	$\tau_{i,m}^{l}$	$\tau_{i,k}^{l}$

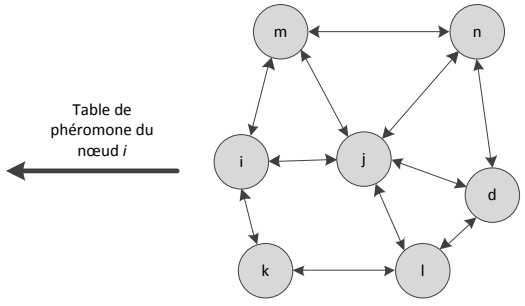


FIGURE 4.4 – Description de la table de phéromone du nœud i.

Comme nous pouvons le constater, ce schéma correspond parfaitement au formalisme général d'un algorithme de colonies de fourmis. Premièrement, les agents fourmis construisent répétitivement et de manière biaisée par les intensités de phéromone, différents chemins entre chaque paire source-destination. Ensuite, ils évaluent la qualité de ces chemins parcourus sur la base des informations collectées. Une fois cette évaluation terminée, ils procèdent au processus de mise à jour des variables de phéromone au niveau de chaque nœud visité.

Même si la majorité des implémentations d'un algorithme de colonies de fourmis qui sont dédiées à ce problème de routage partagent ce même schéma, elles différent selon divers volets. En particulier, elles différent selon la façon dont les agents fourmis sont générées et suivant la manière de mise à jour de la phéromone. Vu cette différence et afin de résumer l'essentiel de ces implémentations, nous reprenons dans cette partie, le cadre général d'une implémentation d'un algorithme de colonies de fourmis pour ce problème de routage qui a été présenté dans [139]. Comme le montre la figure (4.5), ce cadre comporte cinq modules qui représentent les fonctions et structures qui doivent être présentent et assurées au niveau d'un nœud routeur donné : (1) La génération et le management des agents mobiles, (2) Informations de routage, (3) La structure d'un agent, (4) Communications entre agents, et (5) l'acheminement des données. En ayant ce cadre en main, il suffit seulement à savoir comment adapter ces modules afin de développer toute nouvelle implémentation d'un algorithme de colonie de fourmis dédiée au problème de routage en main. Dans ce qui suit, nous détaillons chacun de ces modules.

4.7.1 Module de génération et de management des agents mobiles

Comme nous avons déjà mentionné, une implémentation d'un algorithme de colonie de fourmis qui est dédiée au problème de routage met en opération deux types d'agents fourmis. Une fourmi $F_{s\to d}$ lancée par un nœud source et qui explore le réseau afin de trouver un chemin vers une destination spécifique. Et une deuxième $F_{d\to s}$ qui prend le chemin inverse d'une fourmi $F_{s\to d}$ une fois que cette dernière arrive à sa destination. Cependant, il se peut que d'autres types de fourmis soient utilisés.

4.7.2 Une fourmi $F_{s\to d}$ *(Forward Ant)*

une fourmi de ce genre a comme principal rôle la découverte d'un chemin entre son nœud source et celui de destination. Durant son trajet, elle collecte certaines informations de routage pertinentes à l'état des nœuds et à leurs communications (e.g. l'énergie résiduelle des nœuds, délais expérimenté, etc). Il est à noter qu'il se peut que cette tâche de la collection des informations de routage soit déléguée à une fourmi $F_{d\to s}$ au lieu d'être faite par une fourmi $F_{s\to d}$. Comme le montre la figure (4.5), trois composants administrent le comportement d'une fourmi $F_{s\to d}$: (a) unité de génération des agents, (b) moteur d'acheminement, (c) module de mise à jour des paramètres. Le premier module définit la façon avec laquelle les agents fourmis sont générés. Cette génération peut être faite soit de manière proactive, et donc périodiquement, comme le cas des protocoles : *EEABR* [140] et *ASAR* [141]. Soit de manière réactive, et donc sur-demande (e.g. lorsqu'une nouvelle route est requise) comme le cas des protocoles :

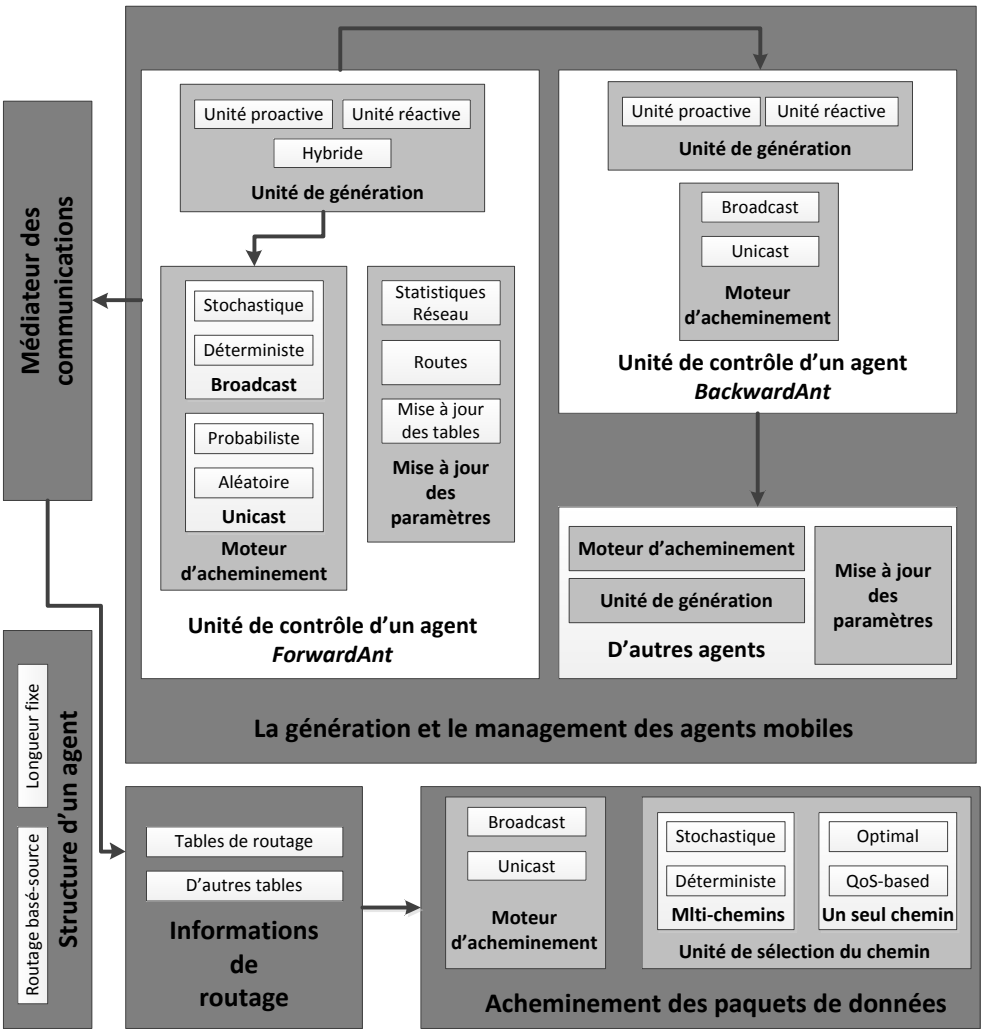


FIGURE 4.5 – Cadre général d'un algorithme de colonies de fourmis appliqué au problème de routage.

AntChain [142] et *ACO-QoSR* [143]. D'autres implémentations, comme *SC* [144] et *JARA* [145], adoptent un schéma hybride qui assemble ces deux méthodes. Une fois qu'un agent fourmi est généré, il commence son exploration en se déplaçant d'un nœud à un autre. Le moteur d'acheminement contrôle cette façon de déplacement des agent fourmis. Généralement, ce moteur envoie cet agent à un seul voisin ou il le diffuse à tous ou une partie de l'ensemble des voisins. Cette sélection du prochain saut se fait communément en se basant sur une règle de déplacement qui assigne, à chaque prochain saut candidat, une certaine probabilité qui dépend de la valeur de phéromone et la valeur heuristique.

4.7.3 Une fourmi $F_{d\to s}$ *(Backward Ant)*

Contrairement à une fourmi $F_{s\to d}$, le comportement d'une fourmi $F_{d\to s}$ dépend de deux composants : unité de génération et moteur d'acheminement. L'unité de génération décide ou pas de la génération d'une telle fourmi à la réception d'une fourmi $F_{s\to d}$ par son nœud de destination. En fait, il est, des fois, inutile de libérer une fourmi $F_{d\to s}$ lorsqu'une fourmi $F_{s\to d}$ trouve un mauvais chemin. Cependant, si une telle génération est décidée, cette fourmi $F_{s\to d}$ se convertit en une autre $F_{d\to s}$ et retrace son chemin-aller pour arriver à son nœud originaire tout en mettant à jour les différentes tables de routage, y compris, les tables de phéromone. D'une manière générale, elle se déplace vers ce nœud originaire sur la base de la liste, dont elle possède, des nœuds qui ont été traversés durant le trajet-aller. Avant de mettre à jour les informations de routage au niveau des nœuds, il est nécessaire qu'elle évalue la qualité de son chemin aller et/ou de retour. Cette évaluation se fait en utilisant les informations de routage dont elle possède et potentiellement celles contenues dans les tables de routage des nœuds. Une fois cette évaluation faite, une fourmi $F_{d\to s}$ procède à la mise à jour des tables de routage des nœuds via le module responsable des communications d'un agent fourmi à l'intérieur du routeur. Durant cette mise à jour, les intensités de phéromone peuvent être renforcées ou évaporées suivant le chemin traversé et sa qualité.

4.7.4 Informations de routage

Ces informations sont des structures de données qui sont maintenues localement au niveau de chaque nœud. Elles regroupent toutes les informations nécessaires au bon fonctionnement de cette opération de routage. Ces structures de données incluent principalement les tables de routage des agents fourmis, des données et d'autres tables qui maintiennent certaines statistiques connexes à l'état des nœuds et celui du réseau. Ces structures de données peuvent également contenir des informations pertinentes aux agents fourmis qui traversent les nœuds. Par exemple, les numéros de séquence de ces agents peuvent être maintenus localement. Cela leur évite la possession de la

liste complète des nœuds traversés qui devient des fois contraignant surtout lorsque le réseau comporte un grand nombre de nœuds comme le cas des réseaux de capteurs sans fil. Les modifications que connaissent ces tables se font via le module responsable des communications d'un agent à l'intérieur du nœud.

4.7.5 Médiateur des communications d'un agent

Ce module, comme son nom l'indique, médite les communications des agents fourmis à l'intérieur des nœuds. Il assure toutes les fonctions nécessaires à l'accès et à la mise à jour des différentes tables de routage.

4.7.6 Acheminement des données

Ce module décide de la façon dont les paquets de données doivent être acheminés. Il fait usage des informations collectées par les agents fourmis et qui sont disponibles au niveau des tables de routage. Ce module comporte deux composants : un moteur d'acheminement et une unité de sélection de chemin. Le moteur d'acheminement s'occupe de l'envoie d'un paquet données en l'envoyant à un seul voisin comme ce qui est communément suivi ou en le diffusant à tous les voisins comme le cas de SC [144]. Outre ce moteur, le composant le plus important est celui responsable de la sélection du chemin à emprunter. Dans ce sens, deux manières sont généralement adoptées. La première consiste à choisir une multitude de chemins. Donc, les paquets de données transitent suivant diverses chemins qui sont choisis de manière stochastique ou déterministe au niveau de la source de données ou au niveau de chaque nœud intermédiaire. Cette façon à faire possède, dans le contexte des réseaux de capteur sans fil, l'avantage d'assurer un équilibre en ce qui concerne la consommation énergétique des nœuds. En revanche et au lieu de choisir de multiples chemins, il suffit à prendre le meilleur chemin déjà défini par les agents fourmis.

4.7.7 Structure d'un agent fourmi

Un agent fourmi prend généralement la forme d'un paquet de contrôle qui est acheminé intelligemment d'un nœud à un autre. Cet agent doit être conscient principalement de sa destination afin de savoir où il doit aller, du trajet parcouru afin de pouvoir retracer le chemin inverse et du coût de ce chemin pour que ce dernier puisse être évalué. Outre ces données, d'autres informations peuvent être utilisées comme les numéros de séquence. Suivant ces données, la taille d'un paquet représentant un agent fourmi peut avoir une longueur variable qui dépend de la liste des nœuds traversés. En outres, cette taille peut être fixe dans le cas où aucune liste est utilisée et les chemins sont retracés sur la base des informations laissées au niveau des nœuds.

4.8 Travaux connexes

Les algorithmes de colonies de fourmis ont été appliqués aux divers problèmes connexes aux réseaux de capteurs sans fil. Outre le problème d'agrégation de données, ces algorithmes ont été utilisés afin de résoudre d'autres problèmes tels que celui de déploiement efficace des nœuds capteurs [146, 147], le problème de trou d'énergie [148, 149] et celui de routage de manière générale [139, 150].

Avant de passer aux travaux connexes à l'application des algorithmes de colonies de fourmis au problèmes d'agrégation et de routage en temps réel des données dans les réseaux de capteurs sans fil, il est utile de présenter les implémentations initiales de ces algorithmes et qui ont constitué la base pour les implémentations ultérieures. Sur le plan historique, l'algorithme *ABC (Ant-Based Control)* [151] est le premier qui a utilisé cette métaphore de colonies de fourmis afin de résoudre ce problème de routage dans les réseaux téléphoniques à base de commutation de circuits. Cependant, les deux implémentions qui ont connu un grand intérêt sont les deux protocoles *AntNet* et *AntHocNet*.

4.8.1 *AntNet*

AntNet [152, 153] a été conçu pour résoudre le problème de routage dans les réseaux filaires. L'objectif a consisté à déterminer de bons chemins qui manifestent des petits temps de latence entre chaque paire de nœuds du réseau. L'utilisation de ce temps de latence comme métrique pour la détermination de ces chemins revient au fait qu'il est proportionnel à la longueur du chemin emprunté en accordance avec l'état courant du réseau (i.e. congestion du trafic réseau) et les caractéristiques physiques de ses constituants (i.e. la vitesse de traitement des nœuds traversés, la capacité de transmission des liens utilisés et le nombre de sauts traversé).

AntNet maintient au niveau de chaque nœud i du réseau deux tables principales. Une première de phéromone T_i qui regroupe des entrées de la forme $\tau_{i,j}^{d}$ définissant la désirabilité appris pour qu'un agent fourmi qui est au niveau du nœud i sélectionne le voisin j pour aller à la destination d. Et une deuxième table M_i qui permet au nœud i d'être conscient du l'état courant du trafic réseau. Chaque entrée de cette table est de la forme $(\mu_{i,d}, \sigma_{i,d}^2, W_{i,d})$ où $\mu_{i,d}$ et $\sigma_{i,d}^2$ représentent respectivement, la moyenne exponentielle mobile et la variance correspondante des temps expérimentés par les agents fourmis $F_{s\rightarrow d}$ durant leur trajets depuis le nœud i jusqu'à leur destination d. $W_{i,d}$ est utilisé pour garder le meilleur temps de latence, $W_{best_{i,d}}$, déjà expérimenté par ces agents fourmis. Les deux mesures $\mu_{i,d}$ et $\sigma_{i,d}^2$ fournissent une idée sur le temps qui peut être envisagé pour aller de i vers d et sur sa stabilité.

De manière concurrente et à partir de chaque nœud du réseau, une fourmi $F_{s\rightarrow d}$

est libérée à chaque intervalle Δt d'une source s pour aller vers une destination d. Cependant, cette fourmi n'est pas automatiquement libérée, mais par contre, elle est libérée sur la base d'une certaine probabilité $p_{s,d}$ qui est calculée en fonction d'une mesure $f_{s,d}$ du flux de données entre s et d comme suit :

$$p_{s,d} = \frac{f_{s,d}}{\sum_{i=1}^{n} f_{s,i}} \tag{4.7}$$

De cette façon, les fourmis adaptent leur exploration suivant les caractéristiques du trafic réseau.

En se déplaçant, une fourmi $F_{s \to d}$ mémorise chaque nœud qu'elle visite et le temps depuis sa source s jusqu'à son nœud courant. Lorsqu'elle arrive au nœud i, elle sélectionne son prochain saut parmi l'ensemble des voisins de i, non déjà visités, suivant la règle de transition suivante qui dépend de la valeur de phéromone $\tau_{i,j}^{d}$ et sur une fonction heuristique $\eta_{i,j}$ qui favorise les voisins ayant peu de paquets en attente dans les files d'attentes qui leur correspondent au niveau du nœud i.

$$p_{i,j}^{d} = \frac{\tau_{i,j}^{d} + \omega \cdot \eta_{i,j}}{1 + \omega \cdot (|N_i| - 1)}, \qquad with \quad \eta_{i,j} = 1 - \frac{q_{i,j}}{\sum_{l=1}^{|N_i|} q_{i,l}} \tag{4.8}$$

Dans cette expression (4.8), N_i représente l'ensemble des voisins du nœud i, $q_{i,j}$ est la longueur, en bits, de la file d'attente du lien $e_{i,j}$ au niveau du nœud i et ω est un paramètre qui contrôle l'importance relative à $\eta_{i,j}$ par rapport à $\tau_{i,j}^{d}$.

Une fois que cette fourmi sélectionne son prochain saut j, elle partage avec les paquets de données la même file d'attente pour qu'elle soit passée au nœud j. À l'arrivée au nœud d, cette fourmi $F_{s \to d}$ se convertit en une fourmi $F_{d \to s}$. Cette dernière hérite les données mémorisées par la fourmi $F_{s \to d}$ et commence son voyage en empruntant le trajet inverse de cette dernière fourmi. Lorsque $F_{d \to s}$ arrive au nœud i après avoir visité j, elle procède à la mise à jour des deux tables T_i et M_i. Les entrées concernées par cette mise à jour sont celles qui correspondent à toute destination d' qui peut être soit d, soit toute autre destination *dest* appartenant au trajet $S_{i \to d}$ avec $dest \neq d$ de la fourmi $F_{s \to d}$.

Chaque valeur $W_{i,d'}$, $\mu_{i,d'}$ et $\sigma_{i,d'}^{2}$ d'une entrée de la table M_i est mise à jour en utilisant les informations stockées dans la mémoire de la fourmi $F_{d \to s}$, en particulier, en utilisant le temps déjà expérimenté, $t_{i \to d'}$, par la fourmi $F_{s \to d}$ durant son trajet du nœud i jusqu'à la destination d'. $\mu_{i,d'}$ et $\sigma_{i,d'}^{2}$ sont mis à jour respectivement comme suit :

$$\mu_{i,d'} = \mu_{i,d'} + \varsigma \cdot (t_{i \to d'} - \mu_{i,d'})$$
$$\sigma^2_{i,d'} = \sigma^2_{i,d'} + \varsigma \cdot ((t_{i \to d'} - \mu_{i,d'})^2 - \sigma^2_{i,d'}$$

Dans ces deux équations, ς est un paramètre qui détermine le nombre de mesures de temps qui vont affecter effectivement le calcul de cette moyenne. Cette mise à jour de ces deux valeurs $\mu_{i,d'}$ et $\sigma^2_{i,d'}$ suit le même modèle utilisé par Jacobson/Karels afin d'estimer le temps après lequel une retransmission d'un message dans une session TCP doit être faite [80].

Outre ces statistiques contenues dans M_i, la table de phéromone T_i est mise à jour en renforçant l'intensité de phéromone des entrées $\tau^{d'}_{i,k}$ sur la base d'un paramètre de renforcement r de la façon suivante :

$$\tau^{d'}_{i,k} = \begin{cases} \tau^{d'}_{i,k} + r \cdot (1 - \tau^{d'}_{i,k}) & si\ k = j \\ \tau^{d'}_{i,k} - r \cdot \tau^{d'}_{i,k} & \forall k \in N_i,\ k \neq j \end{cases} \tag{4.9}$$

Suivant cette mise à jour, l'intensité de la phéromone qui correspond au voisin j (e.g. voisin d'où la fourmi $F_{d \to s}$ est venue) est renforcée de manière à ce que les petites valeurs de phéromone sont plus renforcées par rapport aux valeurs grandes. Cette façon à faire contribue à favoriser l'exploration rapide de nouveaux chemins. En contre-partie, les intensités qui correspondent aux autres voisins, autre que j, sont évaporées de manière à ce que la somme des intensités de phéromone de tous les voisins de i est égale à 1. La valeur du paramètre de renforcement r est calculée en fonction du temps expérimenté $t_{i \to d'}$ par la fourmi $F_{s \to d}$ et des paramètres du modèle M_i. Elle est définie comme suit :

$$r = c_1 \cdot \left(\frac{W_{best_{i,d'}}}{t_{i \to d'}} \right) + c_2 \cdot \left(\frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) + (t_{i \to d'} - I_{inf})} \right) \tag{4.10}$$

Où $W_{best_{i,d'}}$ représente le meilleur temps déjà expérimenté durant l'intervalle $W_{i,d'}$. I_{inf} et I_{sup} sont des estimations de l'intervalle de confiance de la moyenne $\mu_{i,d'}$. c_1 et c_2 contrôlent l'importance relative aux deux termes de l'équation (4.10). Dans cette équation, le premier terme, qui est le plus important, évalue le ratio entre $W_{best_{i,d'}}$ et T alors que le deuxième qui assume le rôle de correction évalue la différence qui existe entre T et I_{inf} en accordance avec l'extension de l'intervalle de confiance.

Après cette mise à jour et différemment à une fourmi $F_{s \to d}$, la fourmi $F_{d \to s}$ est mise en attente au niveau d'une file d'attente dédiée qui est différente à celle utilisée pour la mise en attente des paquets de données. Cela permet d'accélérer le processus de mise à jour des valeurs de phéromones et celles qui correspondent à l'état du réseau.

4.8.2 *AntHocNet*

AntHocNet [154] a été conçu spécialement pour les réseaux ad hoc. Ses différentes caractéristiques ont été développées pour tenir compte de la dynamique du réseau et de la bande passante limitée de ses nœuds. Contrairement à *AntNet* qui est considéré comme un protocole proactif où les agents fourmis travaillent en continue afin de garder les tables de routage à jour, *AntHocNet* est un protocole hybride qui considère deux phases de fonctionnement. Une première phase réactive avec laquelle le processus d'établissement d'une route entre un nœud source et un autre de destination est amorcé lorsque cette communication est nécessaire. Et une deuxième phase proactive responsable de la maintenance de cette route et l'amélioration de sa qualité.

Lorsqu'un nœud source s veut communiquer avec une destination d sans avoir aucune information de routage concernant cette destination, il amorce l'établissement d'une nouvelle route vers cette destination en diffusant un agent fourmi réactif $F_{s\to d}$ à tous ses voisins. Chaque voisin i qui reçoit cet agent le rediffuse à nouveau s'il ne trouve aucune entrée dans sa table de phéromone qui corresponde à cette destination. Sinon, il l'envoie à l'un de ses voisins j ayant une entrée $\tau^d_{i,j}$ correspondante dans sa table de routage. Ce voisin j est choisi selon la probabilité de sélection suivante :

$$p^d_{i,j} = \frac{(\tau^d_{i,j})^\beta}{\sum_{k\in N^d_i}(\tau^d_{i,k})^\beta}, \quad \beta \geq 1 \tag{4.11}$$

Où N^d_i représente l'ensemble de voisins de i à travers lesquels un chemin menant à d est connu et β est un paramètre qui contrôle le niveau d'exploration des fourmis.

Durant ses déplacements, chaque fourmi $F_{s\to d}$ enregistre chaque nouveau nœud visité. Lorsqu'elle arrive à sa destination d, elle se convertit en une autre $F_{d\to s}$ et commence son trajet pour revenir à s en se basant sur le chemin P déjà emprunté par $F_{s\to d}$. Durant ce trajet, elle calcule de manière incrémentale le temps t nécessaire pour qu'un paquet de données arrive à d lorsqu'il emprunte le chemin P. Lorsqu'elle arrive au nœud i après avoir été à j, elle met à jour l'entrée $\tau^d_{i,j}$ de sa table de phéromone de la façon suivante :

$$\tau^d_{i,j} = \gamma \cdot \tau^d_{i,j} + (1-\gamma)\cdot \triangle\tau^d_{i,j}, \quad \gamma \in [0,1] \tag{4.12}$$

avec

$$\triangle\tau^d_{i,j} = \left(\frac{t_{i,d} + h \cdot T_{hop}}{2}\right)^{-1} \tag{4.13}$$

Dans l'équation (4.12), $\triangle\tau^d_{i,j}$ est une mesure qui définit la quantité de phéromone ajoutée et qui dépend de la qualité du chemin trouvé. Elle est calculée suivant l'équation (4.13) en fonction du temps $t_{i,d}$, calculée par la fourmi $F_{d\to s}$, représentant le temps

envisagé pour aller de i à d et du nombre de sauts h séparent ces deux nœuds comme évaluer par cette fourmi. T_{hop} est le temps nécessaire pour se déplacer entre deux nœuds avec un trafic peu congestionné.

Lorsque la fourmi $F_{d\to s}$ arrive à s, les paquets de données commencent à être acheminés vers d étant donné qu'un chemin vers cette destination est déjà établi. À ce moment, la phase proactive de *AntHocNet* commence. Durant cette phase, des agents fourmis proactifs $F_{s\to d}$ sont libérés périodiquement à partir de s. Ces agents se comportent de la même façon que leurs homologues réactifs en choisissant à chaque fois les prochains sauts suivant l'équation (4.11) tout en ayant une petite probabilité d'être diffusés afin de pouvoir explorer de nouvelles routes. Cette exploration répétée des agents proactifs est accompagnée avec un autre processus qui sert à bien guider ces agents fourmis durant leur recherche. Avec ce processus, chaque nœud diffuse régulièrement, à son voisinage, un message l'informant de sa présence. Cela permet aux nœuds d'être conscients de leur voisins immédiats et donc à garder un état cohérent de leurs tables de phéromone.

4.8.3 ACO-ST

Ce protocole [155] est l'une des premières implémentations d'un algorithme de colonie de fourmis pour résoudre le problème d'agrégation de données dans les réseaux de capteurs sans fil. Son but consiste à définir une structure de routage qui maximise cette agrégation tout en produisant un coût de communication minimal. Cela revient à définir un arbre de Steiner de coût minimal qui connecte l'ensemble de nœuds sources avec le nœud puits.

Les auteurs dans [155] proposent deux variations de leur implémentation. Une première centralisée et une autre distribuée. Ces deux variations partagent le même principe à part les spécificités liées à la façon avec laquelle elles sont implémentées sur chaque architecture. Dans ce qui suit, nous présentons le principe général tout en se focalisant sur l'implémentation distribuée.

Avec cette implémentation, une fourmi $F_{s\to Puits}$ est libérée de chaque nœud source. Cette fourmi se déplace d'un nœud à un autre jusqu'à ce qu'elle arrive au nœud puits. La sélection de chaque prochain saut au niveau d'un nœud i se fait suite à la règle (4.1). Donc, ce choix se fait sur la base d'une valeur de phéromone et une autre heuristique. Cette dernière mesure la proximité de i de l'arbre de Steiner qui est en cours de construction. Elle est utilisée afin d'orienter la recherche des fourmis vers les routes déjà établis, et donc, afin de favoriser le chevauchement des routes. Chaque fourmi $F_{s\to Puits}$ porte, en plus de la liste des nœuds déjà visités, le coût $pCost$ du chemin déjà emprunté. Ce coût est mis à jour à chaque déplacement de la fourmi vers le voisin j en l'incrémentant de la distance Euclidienne $\delta_{i,j}$ séparant les deux nœuds i et j. Cette

mise à jour n'est faite que si aucune autre fourmi d'un autre nœud source a déjà passé par i. Sinon, la fourmi $F_{s \to Puits}$ arrête de mettre à jour son coût $pCost$ et suit le chemin déjà emprunté par la première fourmi jusqu'au nœud puits. Pour pouvoir mettre en œuvre ce processus, chaque nœud i maintient un compteur tag_i qui indique le nombre de fourmis $F_{s \to Puits}$ déjà passés par i durant l'itération courante de l'algorithme.

Lorsque $F_{s \to Puits}$ arrive au nœud puits, elle est mise en attente jusqu'à ce que toutes les autres fourmis des autres nœuds sources arrivent à ce nœud puits. Lorsque ce dernier constate l'arrivée de toutes ces fourmis, il additionne leurs $pCost$ afin d'évaluer le coût total $Cost$ de l'arbre de Steiner établi. À ce moment, il envoie à chaque source une fourmi $F_{Puits \to s}$ qui emprunte le même chemin de sa fourmi $F_{s \to Puits}$ correspondante. Chacune de ces fourmis porte avec elle le coût $Cost$ de l'arbre de Steiner construit afin de mettre à jour les variables de phéromone des nœuds. Outre ce coût, une fourmi $F_{Puits \to s}$ comporte deux autres champs : $pCost$ et $rCost$. $pCost$ est utilisé afin de mesurer le distance Euclidienne qui sépare son nœud courant du nœud puits. Il est initialisé à zéro au niveau du nœud puits et mis à jour à chaque déplacement. Le deuxième champs $rCost$ mesure la distance Euclidienne qui sépare chaque nœud de l'arbre de Steiner construit. Il est incrémenté à chaque déplacement, cependant, il est réinitialisé à zéro à chaque fois que la fourmi $F_{Puits \to s}$ détecte un branchement dans l'arbre. Cette détection se fait à l'aide des compteurs tag lorsque par exemple $tag_i < tag_j$ si on suppose que $F_{Puits \to s}$ est au niveau de i après avoir visitée j.

En se basant sur ces trois informations : $cost$, $pCost$ et $rCost$, cette fourmi met à jour la table de phéromone et une autre table additionnelle qui comporte les valeurs heuristiques. Chaque fourmi $F_{Puits \to s}$ qui est au niveau du nœud i et qui arrive de j met à jour l'entrée $\tau_{i,j}$ de phéromone comme suit :

$$\tau_{i,k} = \begin{cases} (1-\rho) \cdot \tau_{i,k} + \rho \cdot \frac{Q}{cost} & si\ k = j \\ (1-\rho) \cdot \tau_{i,k} & \forall k \in N_i,\ k \neq j \end{cases} \tag{4.14}$$

Dans cette équation, ρ correspond au coefficient d'évaporation et Q est une constante. Outre cette table de phéromone, chaque valeur heuristique $\mu_{i,j}$ de la table additionnelle est mise à jour de la façon suivante :

$$\mu_{i,j} = \gamma \cdot rCost + \lambda \cdot pCost \tag{4.15}$$

où γ et λ sont deux paramètres qui contrôlent, respectivement, l'importance relative à $rCost$ et $pCost$. Comme nous pouvons le constater à partir de l'équation (4.15), la recherche des fourmis est biaisée de manière à sélectionner, à chaque fois, les nœuds qui leur permettent à se rapprocher du nœud puits et en même temps qui leur permettent de s'orienter vers les routes qui ont été déjà établis. Cependant, cette mise à jour n'est faite seulement si cette nouvelle valeur $\mu_{i,j}$ est meilleure à celle précédemment calculée.

La fourmi $F_{Puits \to s}$ répète ce processus à chaque nœud traversé jusqu'à ce qu'elle arrive à son nœud source. À ce moment-là, une autre fourmi $F_{s \to Puits}$ est libérée pour une nouvelle itération de l'algorithme.

Cette implémentation et d'après les expérimentations réalisées arrive à construire de bonnes solutions. Cependant, sa réalisation n'est possible qu'avec la considération de certaines hypothèses qui sont difficiles à être assurées dans le contexte des réseaux de capteurs sans fil. Par exemple, le nœud puits ne libère les fourmis $F_{Puits \to s}$ qu'après avoir reçu toutes les fourmis $F_{s \to Puits}$ de tous les nœuds sources. Cela met en obligation la transmission fiable de chacune de ces fourmis qui peut être facilement perdue si on considère les spécificités réelles des réseaux sans fil.

4.8.4 Ant-aggregation

Cet algorithme proposé dans [156] vise aussi à résoudre le problème d'agrégation des données. De manière similaire au précédent travail, chaque fourmi $F_{s \to d}$ qui est libérée du nœud source s se déplace suivant l'équation (4.1) sur la base d'une valeur de phéromone et une autre heuristique. La phéromone est définie en fonction des coûts des arbres d'agrégation qui sont à chaque fois construits alors que la valeur heuristique favorise la sélection des voisins qui mènent vers le nœud puits ou ceux qui mènent vers le plus proche nœud des routes déjà établis. À chaque itération de l'algorithme, la première fourmi qui est libérée d'un nœud source donnée cherche le meilleur chemin qui connecte ce nœud avec le nœud puits. Les autres fourmis qui sont libérées des autres nœuds sources après que la première fourmi termine son exploration cherchent à établir les meilleurs chemins qui connectent leurs nœuds sources avec les plus proches nœuds des routes préalablement construites. Ces fourmis terminent leur exploration une fois qu'elles se rencontrent avec l'un de ces nœuds qui est considéré ensuite comme point d'agrégation. Après que toutes les fourmis terminent leur exploration et arrivent à leurs destinations, l'algorithme évalue le coût total de l'arbre d'agrégation construit et libère d'autres fourmis $F_{d \to s}$ à partir de chaque nœud de destination pour emprunter les chemins inverses de leurs fourmis $F_{s \to d}$ correspondantes et mettre à jour les tables de routage nécessaires. La phéromone est mise à jour suivant l'équation (4.14) alors que chaque valeur heuristique $\eta_{i,j}$ est mise à jour de la façon suivante :

$$\eta_{i,j} = \gamma \cdot cost_{reaching\_aggregation} + \zeta \cdot cost_{destination} + k \cdot cost_{ij\_correlation} \tag{4.16}$$

Dans cette équation (4.16), $cost_{reaching\_aggregation}$ et $cost_{destination}$ représentent, respectivement, une estimation de la distance pour arriver, à partir du nœud i, à un point d'agrégation ou au nœud puits via le voisin j. $cost_{ij\_correlation}$ est un paramètre qui mesure le degré de corrélation des données de i et j. Après la fin de cette itération,

l'algorithme réitère avec une nouvelle permutation de nœuds sources afin de trouver une meilleure solution.

4.8.5 DA-ACA

DA-ACA (Data Aggregation using Ant Colony Algorithm) [5] est un autre algorithme qui s'intéresse au problème d'agrégation dans les réseaux de capteurs sans fil. Les auteurs de ce travail tirent leur idée du précédent travail en se basant sur l'observation que la probabilité pour que deux chemins distincts se chevauchent est peu élevée. Pour cette raison, ils proposent leur nouvel algorithme de colonie de fourmis où ils étendent la région de recherche autour de chaque chemin individuel de manière à ce que des nœuds d'agrégation émergent et de nouveaux chemins qui se chevauchent au niveau de ces nœuds sont formés. Comme nous pouvons le voir à partir de la figure (4.6), chaque chemin connectant chacun des deux sources A et B est étendu de manière à ce que des nœuds tels que G, L et F qui se situent à l'intersection des nouveaux chemins étendus sont observés. Avec ce travail, un chemin entre un nœud source s et le nœud puits est dit étendu à 1 saut si tous les nœuds qui sont à 1 saut de ce chemin sont informés de manière à ce qu'ils aient conscience du chemin qui les connectent avec le nœud source s.

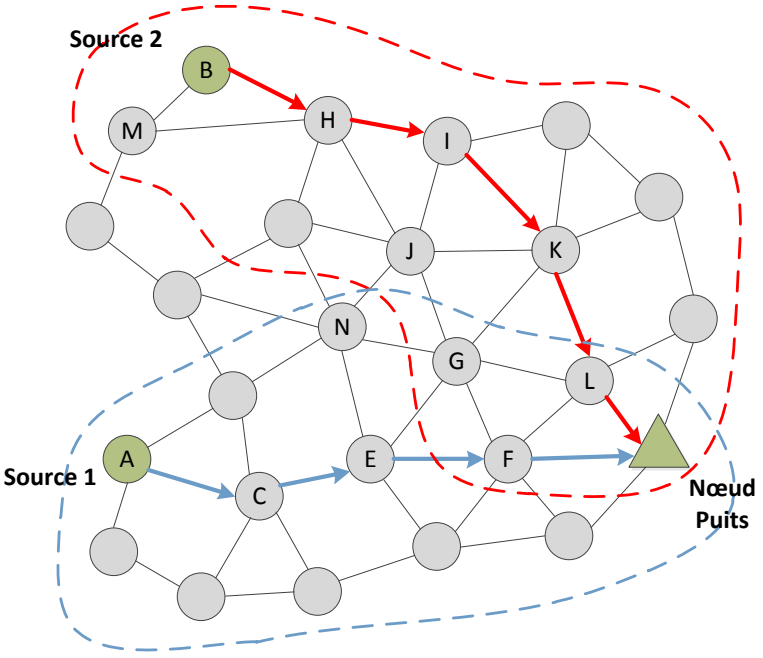


FIGURE 4.6 – Principe de l'algorithme DA-ACA.

Dans une phase initiale, le nœud puits informe, via inondation, chaque nœud capteur du réseau de son identité. Chacun de ces nœuds calcule en se basant sur le message du nœud puits le nombre de sauts qui le sépare de ce dernier. Avec DA-ACA, chaque paquet de données n'est pas émis tout seul, mais par contre, il est accompagné d'un

agent fourmi qui le guide durant ses déplacements. Donc, lorsqu'un nœud capteur donné veut émettre ses données, et donc devient un nœud source, il accompagne son paquet de données avec une fourmi qui sélectionne le prochain saut de ce paquet suivant l'équation (4.1) avec $\alpha = 1$ où la valeur heuristique $\eta_{i,j}$ est l'inverse du nombre de sauts séparant le voisin j du nœud puits. Une fois le prochain saut, j, choisi, un message de sélection qui inclut le paquet de données est émis vers ce nœud. En plus des données, ce message comporte entre autres :

- le nombre de sauts qui séparent son nœud courant de son nœud source s ($D_{i,s}$),
- une valeur EHC qui indique le nombre de sauts auquel le chemin doit être étendu,
- et la durée de vie du message (TTL).

Lorsque le nœud j reçoit ce message, il met à jour une table locale en enregistrant :

- le voisin d'où le message est arrivé,
- le nœud source s de ce message,
- le nombre de sauts qui le sépare de s en incrémentant de 1 la valeur $D_{i,s}$,
- et, en dernier, la valeur TTL.

Si une entrée qui correspond au nœud source s de cette table existe déjà, la valeur $D_{j,s}$ est mise à jour seulement si elle est meilleure par rapport à l'ancienne valeur. Une fois cette table est mise à jour, le nœud j met à jour les champs $D_{i,s}$ et TTL de message de sélection, respectivement par, $D_{j,s}$ et le TTL de la table locale mois 1. Ce message est ensuite diffusé, sans le paquet données, à tous les voisins de j. Chacun de ces voisins se comporte de la même façon avec laquelle le nœud j s'est comporté lorsqu'il a reçu ce message. La rediffusion de ce dernier est arrêtée une fois que la valeur de son champs TTL devienne 0.

La mise à jour de la phéromone est déclenchée dans deux cas. Le premier est lorsqu'un nœud capteur j donné constate que les deux valeurs EHC et TTL de sa table locale sont égales. À ce moment, le nœud j prépare un message de mise à jour de phéromone pour l'émettre à son nœud parent i d'où il a reçu le message de sélection. Ce message comporte principalement :

- Le nombre de sauts qui sépare j du nœud puits (h_j),
- et le coût total des nœuds sources passant par j pour aller au nœud puits ($\Delta\omega_{i,j}$).

Lorsque le nœud i reçoit ce message de mise à jour de phéromone, il met à jour l'entrée $\tau_{i,j}$ de sa table de phéromone de la façon suivante :

$$\tau_{i,j} = (1 - \rho) \cdot \tau_{i,j} + \rho \cdot \Delta\tau_{i,j} \tag{4.17}$$

Où

$$\Delta\tau_{i,j} = [1 + (h_i - h_j)] \cdot \Delta\omega_j \tag{4.18}$$

avec

$$\Delta\omega_j = \sum_{k \in R_j} (H_{k,j})^{-1} + (h_j)^{-1} \tag{4.19}$$

Comme nous pouvons le constater à partir l'équation (4.18), si le nœud j est plus proche du nœud puits par rapport à i, (i.e.$h_i - h_j > 0$) alors la quantité de phéromone déposée est importante. Sinon, s'ils sont à des distances égales du nœud puits (i.e. $h_i - h_j = 0$), alors, une quantité $\Delta\omega_j$ est déposée. Dernièrement et dans le cas où le nœud i est plus proche du nœud puits par rapport à j (i.e. $h_i - h_j < 0$), alors aucune quantité de phéromone n'est déposée.

$\Delta\omega_j$ est calculé suivant l'équation (4.19) en considérant l'inverse de h_j et également l'inverse du nombre de sauts total séparant le nœud j de chaque nœud source de l'ensemble R_j qui inclut tous les nœuds sources que le nœud j connaisse. Donc, à chaque fois que $\Delta\omega_j$ est petit, la quantité de phéromone ajoutée est plus grande. Cela encourage les futures fourmis qui se déplacent avec les paquets de données à emprunter ces trajets.

Outre ce premier cas, chaque nœud agrégateur, et lorsqu'il reçoit une fourmi, diffuse un message de mise à jour de phéromone à tous ses voisins afin qu'ils mettent à jour les entrées dans leur table de routage comme précédemment mentionné.

Afin d'oublier les mauvais trajets, chaque nœud i met à jour chaque entrée $\tau_{i,j}$ de sa table de phéromone suivant l'équation (4.20) s'il ne reçoit aucune fourmi dans un intervalle de temps donné.

$$\tau_{i,j} = (1 - \rho) \cdot \tau_{i,j} \tag{4.20}$$

Après un certain temps, les chemins sur lesquels il y a une grande intensité de phéromone, et donc ceux qui mènent vers les meilleurs points d'agrégation, vont attirer la majorité des fourmis, y compris, les paquets de données afin que ces derniers soient agrégés au niveau de ces nœuds et être acheminés ensuite vers le nœud puits.

4.8.6 La famille d'algorithmes DAACA

Dans [8], les auteurs proposent une famille de quatre algorithmes, appelée DAACA, visant à établir dynamiquement des arbres d'agrégation de données. Différemment aux travaux vus précédemment, Le but consiste à favoriser le chevauchement des routes vers le nœud puits tout en minimisant la consommation énergétique qui résulte suite à la collection des données et surtout à veiller à ce que cette consommation soit équilibrée sur l'ensemble des nœuds du réseau. Le fonctionnement basique de ces algorithmes considère les trois phases suivantes : *initialisation*, *transmission des paquets de données* et *mise à jour des variables de phéromones*. Comme nous pouvons le constater à partir

de ces trois phases, un algorithme DAACA ne considère aucun message de contrôle qui représente une fourmi $F_{s \to Puits}$ ou une autre $F_{Puits \to s}$. Mais par contre, chaque nœud du réseau est considéré comme étant une fourmi artificielle alors que les paquets qui sont échangés entre ces nœuds sont utilisés afin de mettre à jour les variables de phéromone. En fait, ces algorithmes DAACA sont proposés afin de minimiser la charge qui peut être engendrée suite à l'exploitation importante de messages de contrôle. Les quatre algorithmes proposés partagent le même principe de fonctionnement. La différence consiste en la façon dont les variables de phéromone sont mises à jour avec chacun de ces algorithmes.

Avec un algorithme DAACA, chaque nœud i du réseau maintient pour chaque voisin j les informations suivantes :

- l'identifiant de ce nœud j (id),
- une valeur $E'_{distance}(i,j)$ qui est utilisée afin de calculer la valeur heuristique $\eta_{i,j}$,
- L'énergie résiduelle estimée de j ($E_{estimate}(i,j)$),
- L'intensité de phéromone sur le lien $e_{i,j}$ ($\tau_{i,j}$),
- Et la probabilité à choisir j ($p_{i,j}$).

Après une phase d'initialisation dans laquelle chaque nœud du réseau définit ses voisins et enregistre leurs coordonnées, la transmission des données commence lorsqu'un nœud source s détecte un évènement. Le message de ce nœud s est acheminé ensuite d'un nœud à un autre jusqu'à ce qu'il arrive au puits. Chaque nœud i recevant un tel message sélectionne le prochain saut suivant l'équation (4.1) où la valeur heuristique $\eta_{i,j}$ est définie comme suit :

$$\eta_{i,j} = \frac{1}{E'_{distance}(i,j)} \tag{4.21}$$

Où

$$E'_{distance}(i,j) = \frac{E_{distance}(i,j)}{e1(i) \cdot e2(i,j)} \tag{4.22}$$

avec

$$E_{distance}(i,j) = l \cdot E_{elec} + l \cdot \varepsilon_{amp} \cdot \delta^2 \tag{4.23}$$

$$e1(i) = \frac{E_{Cur}(i)}{E_{init}} \tag{4.24}$$

$$e2(i,j) = \frac{E_{estimate}(i,j)}{E_{init}} \tag{4.25}$$

La valeur $E_{distance}(i,j)$ représente l'énergie qui est consommée lorsque i communique avec j. Elle est calculée suivant l'équation (4.23) avec laquelle l représente la taille du

message à envoyer, E_{elec} est l'énergie par bit consommée par le capteur pour activer l'émetteur-récepteur et $\varepsilon_{amp}\delta^2$ est l'énergie par bit consommée par l'amplificateur. $e1(i)$ et $e2(i,j)$ sont calculées respectivement suivant les deux formules (4.24) et (4.25) avec lesquelles E_{init} correspond à l'énergie initiale des nœuds, $E_{Cur}(i)$ correspond à l'énergie courante du nœud i, alors que $E_{estimate}(i,j)$ correspond à l'énergie résiduelle du voisin j comme estimée par i. Cette dernière valeur est calculée comme suit :

$$E_{estimate}(i,j) = E_{init} - \frac{E_{init} - E_{estimate}(i,j)}{times(i,j)} \cdot [times(i,j) + 1] \tag{4.26}$$

Dans cette équation (4.26), $times(i,j)$ représente le nombre de transmissions qui ont été déjà effectuées entre i et j. Comme nous pouvons constater à partir de toutes ces formules qui sont employées pour évaluer la valeur heuristique $\eta_{i,j}$, un algorithme DAACA favorise, lorsqu'il sélectionne un prochain saut, les voisins dont l'énergie nécessaire pour que i communique avec eux n'est pas importante et ceux qui ont un niveau énergétique élevé. Cette façon à faire pénalise tout nœud épuisé pour prendre un autre ayant une source énergétique abondante de manière à ce que la consommation en énergie soit équilibrée.

Après la sélection de ce prochain saut, le nœud i procède à toute agrégation potentielle et envoie le paquet résultant au prochain saut pour qu'il soit acheminé au nœud puits. Donc, un paquet de données est transmis d'un nœud source au nœud puits dans chaque round. Après un certain nombre de rounds, appelé *roundToUpdate,* chaque nœud i du réseau procède à la mise à jour de ses variables locales, y compris, celles de phéromones. Pour la mise à jour de phéromone, chaque nœud évapore dans un premier temps l'intensité de phéromone de chaque voisin j comme suit :

$$\tau_{i,j} = (1 - \rho) \cdot \tau_{i,j}$$

Où ρ correspond à la fraction de phéromone qui n'est pas évaporée. Après cette évaporation, le nœud i sélectionne le voisin avec la plus grande énergie résiduelle, comme estimée par i, (e.g. le voisin j) et renforce l'intensité de phéromone vers ce voisin comme suit :

$$\tau_{i,j} = \tau_{i,j} + E_{distance}(i,j) \tag{4.27}$$

$E_{distance}(i,j)$ est calculée suivant l'équation (4.23). Outre ce dépôt de phéromone et de manière similaire à l'algorithme DA-ACA, chaque voisin d'un nœud qui connait le chevauchement de deux ou plusieurs chemins renforce l'intensité de phéromone vers ce nœud suivant l'équation (4.27). Cela a comme effet, l'orientation des futures paquets de données vers ces points d'agrégation afin qu'ils soient agrégés.

Avec cette façon de mise à jour de phéromone, on vient de présenter la version

basique de la famille DAACA appelée Basic-DAACA. Trois autres versions ont été aussi proposées dans [89] suivant la façon dont cette mise à jour de phéromone est faite.

La deuxième version qui est appelée ES-DAACA (Elitist Strategy DAACA) et en plus à ce que la version basic-DAACA fait, renforce, à chaque *roundToUpdate,* l'intensité de phéromone des liens appartenant aux meilleurs chemins qui connectent chaque nœud source avec le nœud puits. Cela accroit la probabilité d'exploration des solutions les plus prometteuses dans les futures rounds. Pour réaliser cette stratégie, chaque paquet qui est émis d'un nœud source enregistre le chemin traversé durant son trajet vers le nœud puits et l'énergie totale qui est consommée le long de ce chemin. Lorsque le nœud puits reçoit ce paquet, il vérifie si cette énergie totale est inférieure à M_{PCost} qui représente l'énergie totale consommée le long du meilleur chemin déjà trouvé. Si tel est le cas, alors le nœud puits met à jour cette valeur M_{PCost}. En utilisant cette valeur et à chaque *roundToUpdate* rounds, le nœud puits procède à une mise à jour de phéromone globale. Pour cela, l'intensité de phéromone sur chaque lien $e_{i,j}$ qui fait partie de ce meilleur chemin est renforcée de cette façon :

$$\tau_{i,j} = \tau_{i,j} + M_{PCost}$$

Le troisième algorithme de la famille DAACA, appelé MM-DAACA (Maximum & Minimum DAACA), est similaire à ES-DAACA. La seule différence consiste à limiter l'intensité de phéromone sur chaque lien entre l'intervalle $[\tau_{min}, \tau_{max}]$ où τ_{min} et τ_{max} représentent les deux valeurs, minimale et maximale, dont cette intensité peut prendre. Le but ici consiste à favoriser la sélection diversifiée des chemins et éviter de tomber dans le cas de stagnation où un même chemin peut être excessivement utilisé.

Le dernier algorithme de la famille DAACA, appelé ACS-DAACA (Ant colony System DAACA), et par rapport à Basic-DAACA, introduit deux autres mises à jour de phéromone. Une première globale qui est similaire à celle de ES-DAACA. Elle est effectuée suivant cette équation :

$$\tau_{i,j} = (1 - \rho) \cdot \tau_{i,j} + \rho \cdot M_{PCost}$$

Et la deuxième est une mise à jour locale où l'intensité de phéromone sur chaque lien $e_{i,j}$ qui est visité par un paquet de données est mise à jour de la façon suivante :

$$\tau_{i,j} = (1 - \zeta) \cdot \tau_{i,j} + \rho \cdot \Delta\tau_{i,j}^{best}$$

avec

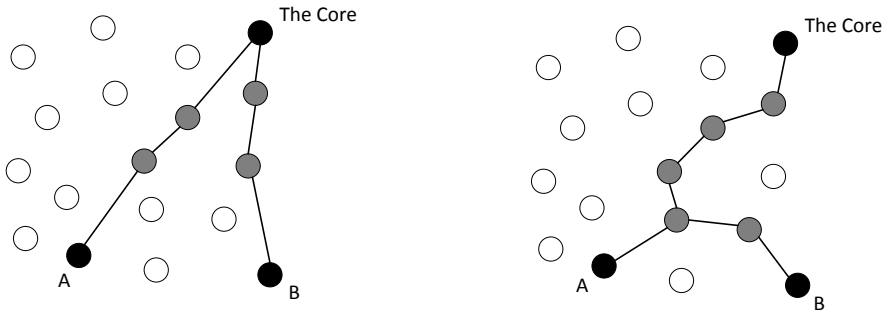
$$\Delta\tau_{i,j}^{best} = \min_{j \in N(i)} \tau_{i,j}$$

Cette dernière mise à jour favorise la sélection diversifiée de chemins.

Les quatre algorithmes proposés dans ce travail ont été comparés avec le précédent travail DA-ACA et l'ensemble d'algorithmes LMST, PEDAP, PEDAP-PA et L-PEDAP présentés dans la section (3.6). Les résultats de simulations, et par rapport à ces travaux, ont montré la supériorité des algorithmes de la famille DAACA en ce qui concerne la capacité à équilibrer la consommation énergétique sur l'ensemble du réseau et à prolonger la durée de vie du réseau.

4.8.7 CSTMAN

CSTMAN [11] est un protocole qui est dédié au problème de Multicast dans les réseau ad hoc. Il vise à construire un arbre de Steiner entre un nœud source et autres nœuds destinataires, assigner à chaque relayeur une puissance de transmission appropriée, minimiser la puissance totale de transmission de l'arbre obtenu et garantir une délivrance en temps borné des données aux différentes destinations. En fait, le problème qui est traité dans ce travail correspond à celui d'agrégation dans les réseaux de capteurs sans fil étant donné que les deux problèmes sont équivalents au problème d'établissement d'un arbre de Steiner. Cet algorithme CSTMAN est présenté dans cette section afin de pouvoir présenter notre algorithme DPIDA qui emprunte certaines idées de lui.



(a) Arbre initial. (b) Arbre final.

FIGURE 4.7 – Arbres initial et final.

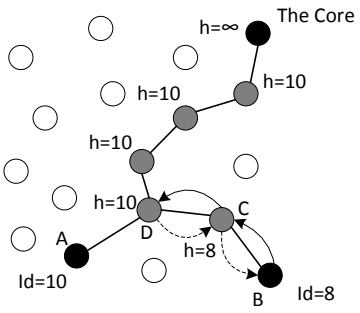


FIGURE 4.8 – Principe de CSTMAN.

CSTMAN procède en deux phases. Une première initiale où un ensemble initial de nœuds relayeurs est défini et une deuxième phase, dans laquelle, des agents logiciels coopèrent ensemble, en imitant du comportement collectif des fourmis, afin d'évoluer l'ensemble initial des relayeurs vers un deuxième ayant une somme des puissances de transmission minimale. Le principe de CSTMAN est illustré par la figure (4.7) où l'ensemble initial et l'ensemble final des relayeurs entre la source de données (cœur) et trois destinataires (nœuds noir) sont montrés. Initialement, le nœud source inonde le réseau par des paquets CORE ANNOUNCE lorsqu'il a des données à transmettre. Les nœuds destinataires lui renvoient, en retour, des messages JOIN REQUEST qui sont acheminés suivant les chemins inverses tout en vérifiant le respect de la borne temporelle. Cet acheminement, et comme le montre la figure (4.7a), génère un arbre initial qui ressemble à l'arbre de plus court chemin avec des nœuds relayeurs qui utilisent des puissances de transmission importantes. Ensuite, chaque nœud destinataire libère périodiquement des agents fourmis, appelés FORWARD ANT, dont l'objectif est de trouver un chemin de coût minimal qui le connecte avec le reste de l'arbre. Cette fourmi se déplace d'un nœud à un autre et se convertit en une BACKWARD ANT seulement si elle se croise avec un nœud relayeur qui fait partie du chemin d'un autre destinataire et le délai de son nœud source jusqu'au nœud cœur via ce relayeur ne dépassant pas le délai exigé. Cette BACKWARD ANT retourne à son nœud source via le chemin inverse de son chemin-aller. Durant son trajet, elle dépose une certaine quantité de phéromone qui est fonction du coût trouvé et met à jour d'autres structures de données comme la table des délais. Avec le temps, un nouvel ensemble de nœuds relayeurs avec des puissances de transmission moins importantes émerge et qui constitue, comme le montre la figure (4.7b), un arbre avec des chemins individuels qui sont fusionnés ensemble.

Cependant, cette fusion doit obéir à une certaine stratégie afin d'éviter la situation où deux destinataires essayent de se connecter l'un avec l'autre et cesse d'être connecté avec le reste de l'arbre. Pour cela, CSTMAN associe à chaque nœud, relais ou destinataire (s'il relaie les données d'un autre nœud destinataire) une hauteur. Cette dernière correspond au plus grand identifiant des nœuds de destinations se basant sur ce nœud pour se connecter avec le nœud cœur. La hauteur de ce dernier est égale à ∞. Donc, un destinataire est autorisé à connecter directement soit, avec le nœud cœur, soit, avec un relais ayant une hauteur supérieure à son identifiant. Cette stratégie est illustrée par la figure (4.8) où le destinataire 10 qui a l'identifiant le plus élevé établit son chemin avec le cœur. Le nœud destinataire 8 est autorisé à se connecter soit avec le cœur, soit avec le chemin du destinataire 10.

Cet algorithme CSTMAN a été proposé comme une extension d'un autre algorithme ECMANSI [7] qui a été lui-même développé à partir d'un premier algorithme appelé MANSI [6]. Avec ce dernier, le but a consisté à établir un arbre de multicast ayant un nombre de sauts minimal sans la considération d'aucune contrainte temporelle. MANSI

a été ensuite davantage développé par les mêmes auteurs afin de proposer ECMANSI où le but a consisté à minimiser seulement la puissance de transmission totale des nœuds de l'arbre de multicast construit.

4.9 Conclusion

Dans ce chapitre, nous nous sommes intéressés aux algorithmes de colonies de fourmis et à leur application aux réseaux de capteur sans fil. Nous avons abordé certaines notions comme celles d'auto-organisation et de mise en réseau bio-inspirée. Nous avons présenté l'origine de ces algorithme et le formalisme général d'un algorithme de colonie de fourmis. La façon dont ces algorithmes sont appliqués au problème de routage dans les réseaux de communication a été également exposée. Et à la fin, un ensemble d'implémentations de ces algorithmes pour résoudre le problème d'agrégation des données dans les réseaux de capteurs sans fil ont été détaillées. Le but derrière ce chapitre a consisté à donner une idée sur les potentialités que présentent ces algorithmes et à faire un tour d'horizon sur les principales approches connexes à notre thématique.

Dans le chapitre suivant, nous proposons notre première implémentation de ces algorithmes qui est appliquée au problème de routage pour agrégation de données. Le but consiste à définir, suite à l'occurrence d'un ou de plusieurs évènements, des structures de routage qui maximisent l'agrégation des données, et donc le chevauchement des routes, et minimisent l'énergie totale qui est consommée durant la collection des données prélevées tout en veillant à ce que cette collection soit menée de manière fiable.

Chapitre 5

Routage à basse consommation énergétique

5.1 Introduction

La conservation de l'énergie est une question clé dans les réseaux de capteurs sans fil. Comme nous l'avons vu dans le premier chapitre, différentes techniques peuvent être utilisées pour optimiser la consommation énergétique. L'agrégation des données est l'une de ces techniques. En exploitant la corrélation qui peut exister entre les données prélevées, elle vise à agréger ces données et éliminer la redondance qui peut exister afin de minimiser la taille et le nombre de messages, particulièrement redondants, qui circulent dans le réseau. D'un autre côté, le contrôle de topologie qui se réfère à la façon avec laquelle les nœuds du réseau coordonnent leurs décisions en ce qui concerne leurs puissances de transmission représente également une technique qui permet aussi d'optimiser cette consommation. De plus, c'est une technique qui peut minimiser considérablement la probabilité d'occurrence des collisions et d'accroître la capacité du réseau.

Dans la littérature, des travaux tels que InFRA[88, 104], DRINA[79, 105], DDAARP [157] et DST[9] visent à établir des structures de routage qui maximisent l'agrégation des données et ayant un nombre minimal de sauts. Le but ici consiste à minimiser le nombre de transmissions qui sont nécessaires à la collection des données. En fait, la connectivité entre les différents nœuds de cette structure est définie par leurs niveaux de puissances de transmission. Ces nœuds ne sont pas obligés d'utiliser la même puissance de transmission, mais par contre, ont le potentiel de diminuer leurs puissances afin de minimiser la puissance totale de transmission. Cela contribue à minimiser davantage l'énergie qui est consommée par ces nœuds lors de la communication. Dans ce travail, nous allons plus loin en proposant le protocole PALDA[3] (*Power-efficient routing based on Ant colony optimization and LMST for in-network Data Aggregation*) qui vise à :

1. Établir une structure de routage qui maximise le nombre de routes qui se chevauchent, et par conséquent, maximise l'agrégation des données. Cette structure correspond à un arbre de Steiner,
2. Assigner à chaque nœud de cette structure une puissance de transmission appropriée,
3. Minimiser la somme des puissances de transmission assignés,
4. Et assurer une assignation de puissance de transmission symétrique qui considère la bidirectionnalité des liens entre les nœuds de la structure formée afin de pouvoir délivrer les données prélevées de manière fiable.

Le protocole proposé se base sur deux éléments clés. Le premier est pertinent à l'exploitation de l'algorithme LMST afin de construire, au sein de chaque cluster, un arbre ayant un coût énergétique minimal et qui sert à la collection des données. Le deuxième élément est pertinent à l'utilisation des algorithmes de colonies de fourmis afin de former les routes entre les différents cluster-heads et le nœud puits en déterminant les nœuds relais et leurs puissances de transmission. En fait, des travaux antérieurs qui utilisent ces algorithmes de colonies de fourmis pour l'établissement d'une structure de routage qui correspond à un arbre de Steiner tels que, DA-ACA[5], MANSI[6], ECMANSI[7] et la famille d'algorithmes DAACA[8], laissent les agents fourmis explorer le réseau en cherchant dans toutes les directions afin de pouvoir se rencontrer avec des nœuds de l'arbre de routage existant, et par conséquent, assurer le chevauchement des routes. Différemment à ces travaux, nous établissons une structure géométrique sous-jacente afin d'orienter la recherche des fourmis seulement vers les parties nécessaires du réseau au lieu de les laisser chercher aveuglement dans tous les sens. Cette façon à faire contribue à l'accélération de la vitesse de convergence de la solution finale. Suivant la façon de construction de cette structure géométrique, deux variations de PALDA sont proposées. La première, PALDA-S (Statique) où les routes sont établies suivant l'ordre d'occurrence des évènements et sont utilisées durant la période totale de leurs occurrences. Et la deuxième, PALDA-D (Dynamique) où les routes sont formées suivant les distances des cluster-heads du nœud puits et elles sont reconstruites à chaque occurrence d'un nouvel évènement afin d'améliorer la qualité de l'arbre final de routage.

Le présent chapitre décrit notre première contribution en présentant les modèles (réseau et évènement) adoptés, une formulation du problème traité, une description détaillée de PALDA avec ses deux variations, la méthodologie de simulation suivie et les résultats de simulation obtenus suite à la comparaison qui a été conduite avec les travaux antérieurs.

5.2 Modèle réseau et formulation du problème

5.2.1 Modèle réseau

Dans ce travail, le modèle réseau suivant est supposé :

1. Notre réseau est constitué d'un ensemble de nœuds capteurs statiques et un seul nœud puits.
2. Les nœuds capteurs peuvent ajuster leurs portées de communication en ajustant leurs niveaux de puissance de transmission, et cela , sans dépasser une puissance maximale $Power_{max}$ qui correspond à une portée maximale R_{max}.
3. Chaque nœud capteur est conscient de ses coordonnées, celles de ses voisins et de celles du nœud puits.
4. Soit $G = \{V, E\}$ le graphe qui modélise la topologie de notre réseau où :
   (a) $V = \{v_1, v_2, ..., v_n\}$ est l'ensemble des sommets représentant les différents nœuds du réseau tel que $|V| = n$ et v_1 est le nœud puits .
   (b) Et E l'ensemble d'arcs (liens) qui relient ces nœuds et qui est défini comme suit :
   $$E = \{(u, v)|u \in V \, , \, v \in V \; et \; \delta_{u,v} \leq R_{max}\}$$
   Sachant que :
   $$\delta_{u,v} = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2} \tag{5.1}$$
   représente la distance euclidienne entre les deux nœuds u et v , avec $u = (x_u; \, y_u)$ et $v = (x_v; \, y_v)$.
5. Nous considérons des évènements statiques dont chacun est décrit par une région d'influence. Un modèle binaire de détection est supposé. Donc, chaque nœud qui est l'intérieur de la région d'influence d'un évènement le détecte. Soit $S = \{s_1, s, ..., s_m\}$ l'ensemble des nœuds sources, c.à.d. les nœuds qui détectent un évènement, tel que $|S| = m$ et $S \subseteq (V - \{v1\})$.
6. Une fonction d'agrégation parfaite est supposée. C.à.d. k paquets de taille l sont agrégés en un seul paquet de même taille l. Cela correspond au cas où de simples agrégateurs tels que le min, le max, la moyenne, etc. sont utilisés. Cette fonction est exécutée au niveau des points d'agrégation.
7. De manière similaire à [158], la puissance de transmission nécessaire pour que le nœud i communique avec j est calculée en assumant la formule :
   $$power_{ij} = \delta_{ij}^{\alpha} \tag{5.2}$$

où α est l'exposant d'affaiblissement de propagation *« path loss » et* δ_{ij} est la distance entre i et j.

5.2.2 Définition du problème

Dans ce travail, l'objectif est de définir, une fois qu'un évènement ou plusieurs arrivent, une structure de routage qui maximise l'agrégation des données et dont la puissance totale de transmission de l'ensemble des nœuds constituant cette structure soit minimale. En fait, cette structure est un arbre de Steiner qui connecte l'ensemble des nœuds sources avec le nœud puits et dont la somme des puissances de transmission des nœuds capteurs sources et nœuds relais de l'arbre résultant soit minimal. Notre problème est formulé de la façon suivante :

Étant donné :

le graphe $G = \{V, E\}$ qui modélise la topologie de notre réseau, où V l'ensemble des sommets qui correspondent à l'ensemble des nœuds du réseau et E l'ensemble d'arcs qui correspondent aux liens entre ces nœuds, avec un nœud puits $v_1 \in V$ et un ensemble $S \subseteq (V - \{v1\})$ de nœuds sources, trouver le sous-graphe $G' = \{V', E'\}$ de G, où $V' = S \cup L \cup \{v_1\}$ avec L représente l'ensemble des nœuds relais [1], et une assignation des puissances de transmission $p : V' \rightarrow R^+$ tel que :

1. G' est un arbre qui est enraciné au niveau du nœud puits v_1 et qui couvre tous les nœuds sources dans S.
2. $\sum_{v \in G' \, et \, v \neq v_1} power(v)$ est minimisée, où $power(v) = max_{u \in N_v} power_{v,u}$ avec $N_v = \{u | u \in V' \, et \, e_{v,u} \in E'\}$ l'ensemble des voisins de v dans G'.

5.3 Description de PALDA

Notre protocole PALDA est un protocole réactif avec lequel un processus d'assignation des rôles est déclenché une fois un évènement est détecté. Les nœuds capteurs qui détectent le même évènement sont organisés en cluster, parmi cet ensemble un cluster-head est élu. Ce dernier agrège les paquets qui arrivent des membres du cluster et envoie le paquet résultant au nœud puits. Le paquet agrégé est routé ensuite d'un nœud à un autre jusqu'à ce qu'il arrive au nœud puits. Pour pouvoir réaliser ce processus et définir la structure de routage, notre protocole considère les rôles suivants :

- **Collaborateur :** c'est un nœud qui détecte un évènement et qui représente un membre d'un cluster donné. Il est responsable de rapporter ses données captées vers le cluster-head (Coordinateur),

1. Un nœud relais est tout nœud dans l'arbre construit qui n'est ni nœud feuille ni nœud racine. Dans notre travail, un tel nœud peut être un nœud source.

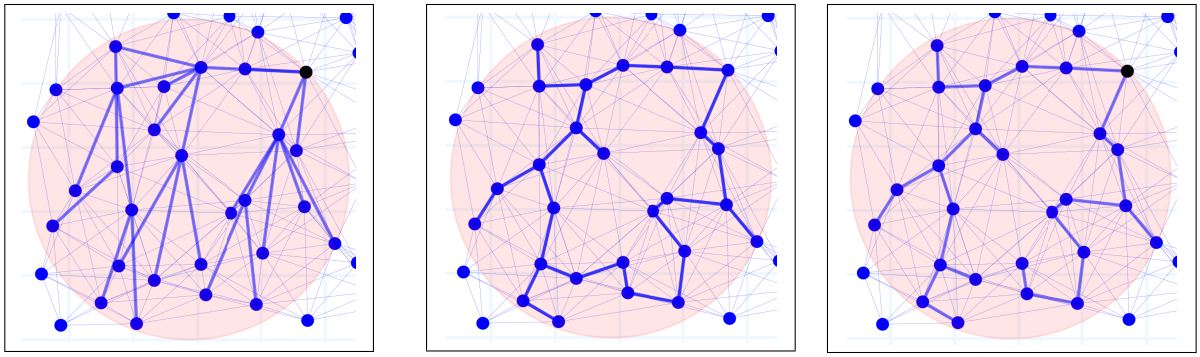
- **Coordinateur :** c'est un nœud qui détecte un évènement et qui représente le cluster-head de l'ensemble des nœuds qui forment un même cluster (détectant le même évènement). Il rassemble les données captées à partir des nœuds collaborateurs, les agrège et rapporte le paquet résultant vers le nœud puits,
- **Relais :** c'est un nœud qui achemine les données qui lui arrive de ses fils vers le nœud puits,
- **Puits :** c'est un nœud qui s'intéresse aux différents évènements qui se produisent.

Après une phase d'initialisation dans laquelle le nœud puits diffuse sa position à l'ensemble du réseau et chaque nœud définit son voisinage en enregistrant l'identifiant et la position de chaque voisin, le réseau reste inactif jusqu'à l'occurrence d'un ou plusieurs évènements. Lorsqu'un évènement arrive, les trois phases suivantes sont exécutées :

- **Phase 1 :** dans cette phase, les clusters sont formés et les rôles coordinateur et collaborateur sont assignés aux nœuds qui détectent un nouvel évènement. En outre, au sein d'un cluster, une topologie creuse qui se base sur l'algorithme localisé de l'arbre couvrant de poids minimal (LMST) est construite sur le sous-graphe qui correspond au sous-réseau contenant seulement les nœuds membres d'un cluster. Ensuite, un arbre de plus court chemin pondéré, qui est enracinée au niveau du coordinateur et qui couvre tous les collaborateurs est défini au-dessus de cette nouvelle topologie creuse.
- **Phase 2 :** lorsqu'un coordinateur est élu, il informe le nœud puits de sa position. Le nœud puits lui renvoie les positions de tous les coordinateurs qui existent déjà. Il informe également ces derniers de la position du nouveau coordinateur. Ces positions permettent aux coordinateurs d'établir une structure géométrique sous-jacente qui est exploitée par les agents fourmis lors de la formation des routes et l'assignation des puissances de transmission.
- **Phase 3 :** les nœuds assumant le rôle relais sont identifiés et l'arbre de routage qui relie les coordinateurs avec le nœud puits est établi. L'identification des nœuds relais et l'établissement de cet arbre se font suite à un processus d'assignation des puissances de transmission qui se base sur l'algorithme de colonie de fourmis.

5.3.1 Formation des clusters

À l'occurrence d'un évènement, les nœuds qui le détectent entrent dans la phase d'élection du cluster-head. Ce processus est décrit par l'algorithme 5.1. Pour cette élection, tous les nœuds qui détectent un évènement sont éligibles. Différentes stratégies peuvent être appliquées pour l'élection du nœud coordinateur (e.g. le nœud ayant le plus petit identifiant, celui qui a la plus grande énergie résiduelle, etc). Dans ce travail et de manière similaire à DST [9], le nœud le plus proche au nœud puits est élu comme



(a) (b) (c)

FIGURE 5.1 – Deux façons d'établissement des routes à l'intérieur d'un cluster : la première, illustrée par la figure (5.1c), où un arbre du plus court chemin pondéré, qui est enraciné au niveau du nœud coordinateur (le nœud noir) et qui couvre tous les nœuds collaborateurs est établi au-dessus de la topologie LMST (figure (5.1b)). Cette dernière topologie est calculée au-dessus du sous-graphe qui correspond au sous-réseau qui est formé seulement par les membres qui détectent l'évènement (le disque rose). La deuxième possible façon, illustrée par la figure (5.1a), consiste à construire un arbre du plus court chemin (en termes de nombre de sauts) au-dessus de la topologie originale.

cluster-head. (lignes 9 et 10 de l'algorithme 5.1). Dans le cas d'égalité, le nœud qui a le plus petit identifiant est choisi. (lignes 11 et 12 de l'algorithme 5.1). À la fin de cette phase, un seul nœud est élu comme coordinateur et les autres membres assument le rôle collaborateur.

Afin de pouvoir collecter les données au sein du cluster, un arbre qui est enraciné au niveau du coordinateur et qui couvre tous les collaborateurs est nécessaire. Une solution possible et comme montre la figure (5.1a) consiste à établir un arbre du plus court chemin au-dessus la topologie originale. Cependant, pour diminuer la puissance de transmission qui est utilisée par chaque membre du cluster nous procédons différemment. Par conséquent et en se basant sur l'algorithme localisé de l'arbre couvrant de poids minimal (LMST), nous construisons, à l'intérieur du cluster et comme le montre la figure (5.1b), une nouvelle topologie creuse au-dessus du sous-graphe qui correspond au sous-réseau qui est formé seulement par les membres du cluster. Ensuite et comme le montre la figure (5.1c), un arbre du plus court chemin qui est enraciné au niveau du nœud coordinateur (nœud noir dans la figure (5.1c)) et qui couvre tous les nœuds collaborateurs est établi sur les arcs de cette nouvelle topologie. Cet arbre est utilisé ensuite pour la collection des données à l'intérieur du cluster et leur délivrance au coordinateur.

Ce processus de construction de cette nouvelle topologie et de formation de l'arbre à l'intérieur du cluster se font conjointement avec le processus d'élection du coordinateur. Au début, chaque nœud qui détecte l'évènement diffuse à son voisinage un message

d'annonce de détection en utilisant sa puissance maximale de transmission (ligne 4 de l'algorithme 5.1). Il collecte ensuite, durant un intervalle[2] $t_{collect}$, les messages d'annonce de ses voisins (ligne 5 de l'algorithme 5.1). Après cette collection, chaque nœud construit son arbre couvrant de poids minimal (MST) en considérant seulement ses voisins d'où il a reçu des messages d'annonce et identifie ses nouvelles voisins dans la topologie formée (ligne 7 de l'algorithme 5.1). Pour former un MST qui est énergétiquement efficace, nous considérons comme poids d'un arc séparent deux nœuds la puissance de transmission nécessaire pour qu'ils puissent communiquer. Cette puissance est calculée suivant l'équation (5.2). Dans notre travail, nous choisissons d'utiliser la version LMST$^-$(voir la section 3.6.1.1 pour plus de détails à propos de cette version). Pour que cela puisse être possible, les nœuds membres échangent leurs MST locaux avec leurs voisins en utilisant les messages d'annonce d'intention de leadership durant la mise en place des routes à l'intérieur du cluster.

L'établissement de l'arbre à l'intérieur du cluster se fait à l'aide des messages d'annonce d'intention de leadership qui sont diffusés par chaque coordinateur candidat. Chaque nœud source qui reçoit ce type de message le rediffuse, après mise à jour, à ses voisins dans la topologie creuse s'il trouve qu'il n'a jamais reçu un message du coordinateur candidat spécifié dans ce message (ligne 17 de l'algorithme 5.1). Il le rediffuse également dans le cas où le chemin qui mène vers ce coordinateur candidat via l'expéditeur du message est plus avantageux en termes de la puissance totale de transmission (lignes 19-24 de l'algorithme 5.1). Après un intervalle $t_{discovery}$, chaque collaborateur choisit comme parent le voisin qui mène vers le coordinateur candidat le plus proche du nœud puits (ligne 30 de l'algorithme 5.1).

5.3.2 Établissement de la structure géométrique sous-jacente

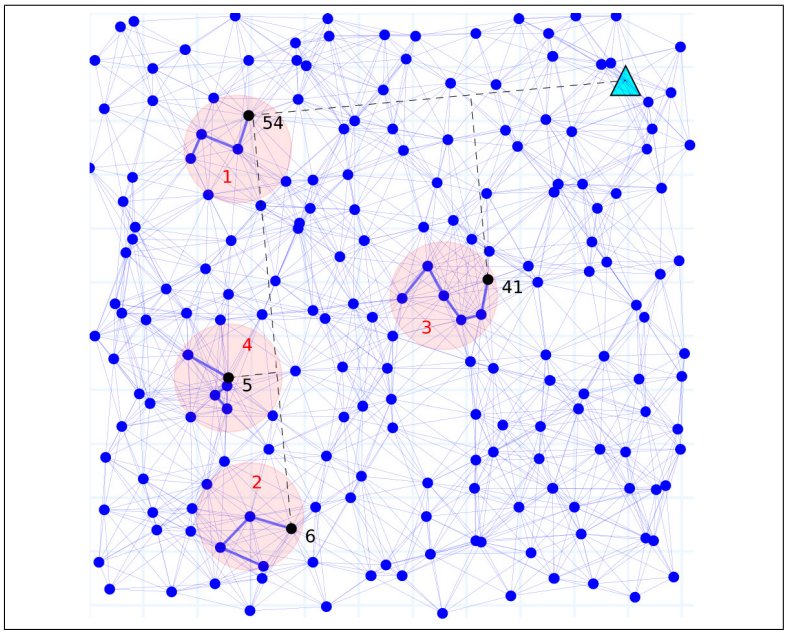
De manière similaire à DST [9] et YEAST [106], PALDA se base sur l'établissement d'une structure géométrique sous-jacente pour la formation de l'arbre de routage qui relie les coordinateurs avec le nœud puits. Cette structure est exploitée par les agents fourmis afin d'identifier les nœuds relais et leurs puissances de transmission.

Tout nœud source qui est élu comme coordinateur informe le nœud puits de sa position. Le nœud puits l'informe ensuite des coordonnées des nœuds coordinateurs existants. Il informe également les autres coordinateurs des coordonnées de ce nouveau coordinateur. Chaque coordinateur, en se basant sur ces positions, ses propres coordonnées et celles du nœud puits calcule son propre segment. Cela génère une structure géométrique qui est formée par la connexion d'un ensemble de segments qui démarrent des nœuds coordinateurs. L'établissement des segments peut se faire en suivant différentes approches [106]. Dans ce travail, nous considérons deux variations de PALDA

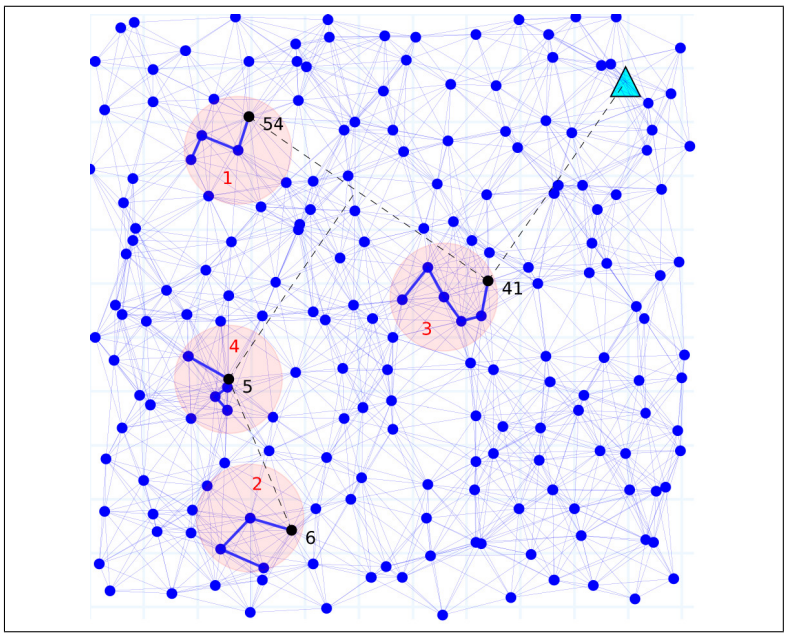
2. Cet intervalle est utilisé pour permettre à un nœud d'entendre toutes les annonces.

Algorithme 5.1 Formation d'un cluster, élection du cluster-head et la formation des routes au sein du cluster.

1: **for each** $u \in S$ **do**
2: $role_u \longleftarrow coordinator$;
3: *// Node u announces event detection by broadcasting an Event_Announcement_Packet (EAP)*
4: Node u broadcasts an EAP ;
5: Node u Collects $EAPs$ for $t_{collect}$;
6: *// Node u computes its neighbors on sparse topology considering only its neighbors* $w \in S$ *(*N_u*),*
7: Node u establishes its LMST and identifies its new neighbors on sparse topology N_u^{LMST} ;
8: **for each** $w \in N_u$ **do**
9: **if** $distanceToSink(u) > distanceToSink(w)$ **then**
10: $role_u \longleftarrow collaborator$;
11: **else if** $distanceToSink(u) = distanceToSink(w)$ **And** $id(u) > id(w)$ **do**
12: $role_u \longleftarrow collaborator$;
13: **end if**
14: **end for**
15: **if** $role_u = coordinator$ **then**
16: $CCL.add\{id(u), distanceToSink(u)\}$; *//CCL is the list of coordinators candidates*
17: Node u announces coordinator intention to its neighbors N_u^{LMST} ; *//event-scoped flooding*
18: **end if**
19: **while** Coordinator Intention Packet (CIP) received in $t_{discovery}$ **do**
20: **if** update required for CIP **then**
21: Node u updates the parent node leading to this candidate coordinator ;
22: Node u updates and broadcasts the received CIP to its neighbours N_u^{LMST} ;
23: **end if**
24: **end while**
25: //smallestDTS(CCL) corresponds to the smallest distance among the distances from each candidate coordinator in CCL to the sink
26: **if** $role_u = coordinator$ **And** $distanceToSink(u) \neq smallestDTS(CCL)$ **then**
27: $role_u \longleftarrow collaborator$;
28: **end if**
29: **if** $role_u = collaborator$ **then**
30: Node u chooses as parent node the one leading to the closest candidate coordinator to the sink ;
31: **end if**
32: **end for**



(a) PALDA-S.



(b) PALDA-D.

FIGURE 5.2 – L'établissement de la structure géométrique sous-jacente.

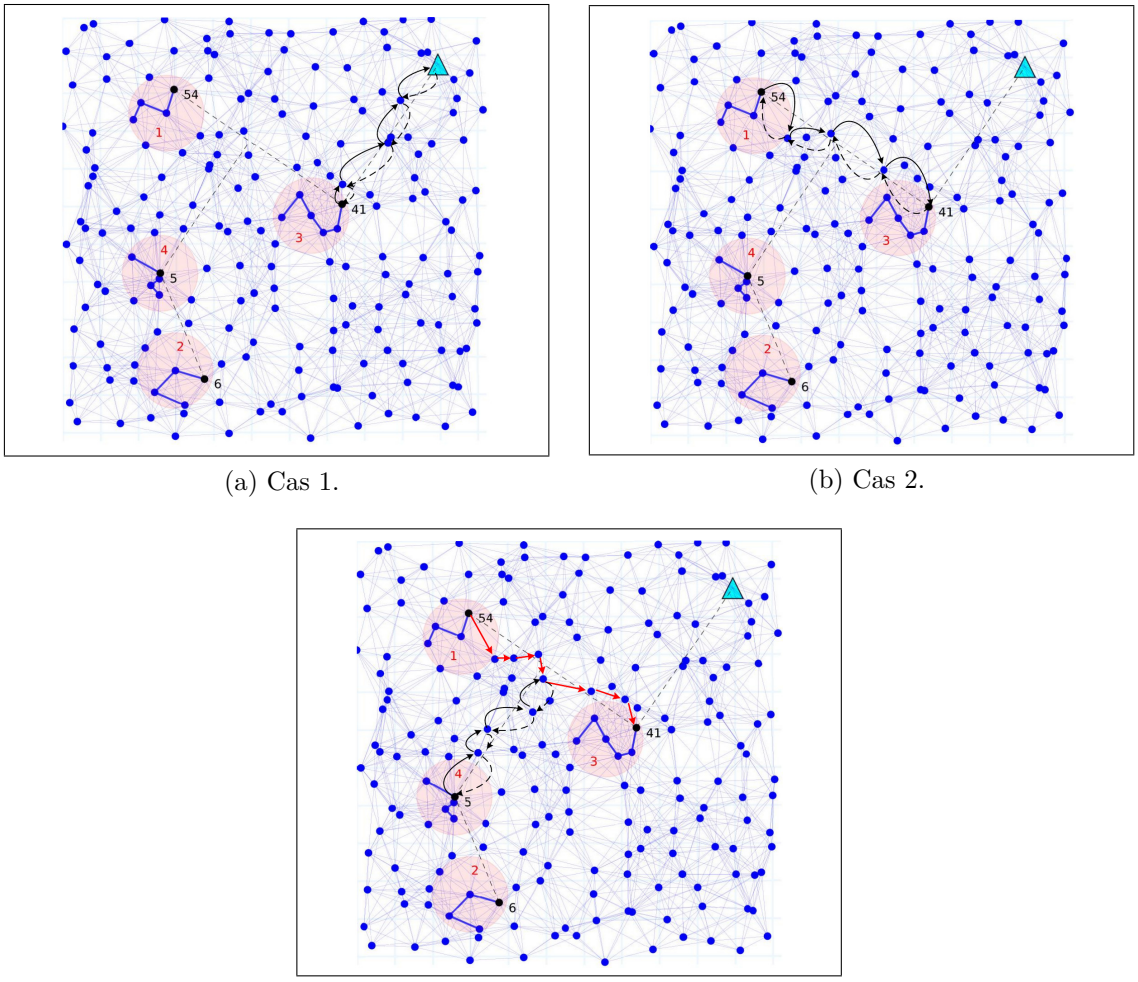
suivant l'approche adoptée lors de l'établissement de cette structure géométrique sous-jacente.

1. **PALDA-S (Statique) :** avec cette approche, les coordinateurs établissent leurs segments suivant l'ordre d'occurrence des évènements. Donc, le coordinateur connexe au premier évènement qui se produit est le premier qui forme son segment entre lui et le nœud puits. Ensuite, vient le tour du coordinateur responsable du deuxième évènement qui se passe et qui établit son segment entre lui et le plus proche points des segments déjà crées. Ces opérations se répètent jusqu'à ce que tous les coordinateurs établissent leurs segments. La structure qui est formée en suivant cette approche est illustrée par la figure (5.2a) où l'ordre d'occurrence d'un évènement est noté en rouge à l'intérieur de chaque évènement. Avec cette approche, un segment qui est initialement créé reste inchangé jusqu'à la fin de l'occurrence de l'évènement. Cela génère une structure de routage statique où les mêmes routes (créées durant la phase 3) sont utilisées durant toute la période d'occurrence des évènements. La complexité algorithmique de ce processus est $O(e)$ où e est le nombre d'évènements.
2. **PALDA-D (Dynamique) :** avec cette approche, le coordinateur le plus proche au nœud puits est le premier qui forme son segment entre lui et le nœud puits. Ensuite, le deuxième coordinateur qui est plus proche au nœud puits établit son propre segment le séparant du plus proche points des segments déjà formés. Ces opérations se répètent jusqu'à ce que tous les coordinateurs établissent leurs propres segments. La structure qui est formée en suivant cette approche est illustrée par la figure (5.2b). Avec cette approche, les segments sont reconstruits à chaque fois qu'un nouvel évènement se passe. Cela permet à améliorer la qualité de l'arbre de routage final formé. La complexité algorithmique de ce processus est $O(e)$ où e est le nombre d'évènements.

Tout coordinateur qui veut établir son segment, calcule en premier, et suivant l'approche adoptée, les segments des autres coordinateurs qui ont déjà établi leurs segments. Une fois créés, il forme son propre segment le séparant du plus proche point de ces segments.

5.3.3 Formation des routes

Après l'établissement de la structure géométrique sous-jacente, les nœuds coordinateurs procèdent au processus de détermination des routes qui les relient avec le nœud puits. Durant cette phase, l'ensemble des nœuds relais et leurs puissances de transmission sont déterminés. Ce processus est basé sur l'algorithme de colonie de fourmis. Donc, chaque coordinateur libère, chaque *ANT_INTERVAL*, une Forward Ant avec



(a) Cas 1.

(b) Cas 2.

(c) Cas 3.

FIGURE 5.3 – les nœuds de destination potentiels d'une fourmi Forward Ant donnée. Une Forward Ant libérée d'un coordinateur donné se déplace d'un nœud à un autre, suivant le segment de droite de ce coordinateur, jusqu'à ce qu'elle arrive à son nœud de destination ou bien elle se rencontre avec un nœud relais de ce nœud. À ce moment et comme deuxième étape, cette Forward Ant se transforme en une fourmi Backward Ant et prend le chemin inverse vers son coordinateur source correspondant tout en déposant de la phéromone le long du chemin.

l'objectif de trouver un chemin de coût minimal qui le relie avec le reste de l'arbre de routage. Suivant l'extrémité du segment de chaque coordinateur, ce dernier est autorisé à se connecter avec l'un des nœuds suivants :

1. **le nœud puits :** par exemple, le coordinateur 41 dans la figure (5.3a). Dans ce cas, une Forward Ant porte, comme destination et position de destination, respectivement l'identifiant du nœud puits et sa position,
2. **un autre coordinateur :** par exemple, le coordinateur 54 dans la figure (5.3b) est autorisé à se connecter avec le coordinateur 41. Dans ce cas, une Forward Ant porte, comme destination et position de destination, respectivement l'identifiant de ce coordinateur et sa position,
3. **un nœud relais responsable de l'acheminement des données d'un autre coordinateur :** Ce dernier coordinateur doit être l'extrémité du segment auquel le segment du premier coordinateur est connecté (e.g. le coordinateur 5 dans la figure (5.3c) est autorisé à se connecter avec l'un des relais sur lesquels le coordinateur 54 compte pour l'acheminement de ses données). Pour cette raison, chaque nœud qui est choisi comme relais enregistre l'identifiant du coordinateur qui se base sur lui pour l'acheminement des données. En outre, chaque Forward Ant porte, comme destination et position de destination, respectivement l'identifiant de ce coordinateur (pour pouvoir identifier un nœud relais de ce coordinateur) et la position de l'extrémité du segment de son coordinateur source après son prolongement [3] par une certaine distance.

Un nœud capteur i utilise les trois tables suivantes lors de l'exécution de l'algorithme :

1. **Table de phéromone :** une entrée de cette table définit l'intensité de la phéromone $\tau(i, j, d)$ dont i maintient sur le lien qui le sépare de son voisin j par rapport à d. La valeur de phéromone est comprise entre 0 et 1.
2. **Best-cost Table :** cette table est utilisée afin de prendre conscience du coût du meilleur chemin $cost(i, d)$ qui connecte i avec d ou un nœud relais de d.
3. **Table de pistage :** cette table est utilisée principalement pour pouvoir pister les chemins empruntés par les Forward Ants.

Une fourmi peut prendre l'un des deux types suivants : fourmi déterministe et non déterministe. Une fourmi déterministe suit toujours le prochain saut qui fait partie du chemin ayant le meilleur coût déjà trouvé. Son principal objectif est la définition des nœuds relais avec leur nœuds fils et parent et le renforcement du meilleur chemin. En contre-partie, une fourmi non-déterministe détermine son chemin de manière probabiliste afin de découvrir de nouveaux chemins ayant des moindres coûts.

3. Dans ce troisième cas, le segment du coordinateur source d'une Forward Ant est prolongé au niveau de l'extrémité qui est différente de ce coordinateur afin d'assurer la rencontre avec un nœud relais.

5.3.3.1 La variable heuristique :

L'équation (5.3) représente la variable heuristique qui est utilisée pour diriger les choix d'une fourmi vers les nœuds voisins proches de son nœud courant. Elle permet également de favoriser les nœuds voisins qui ne sont pas très loin du segment du coordinateur source de cette fourmi.

$$\eta(i,j) = \frac{1}{power_{i,j} + \delta^{\varphi}_{j,proj_j}} \tag{5.3}$$

Dans l'équation (5.3), $power_{i,j}$ représente la puissance de transmission nécessaire pour que i communique avec son voisin j. Cette puissance est calculée suivant l'équation (5.2). L'incorporation de cette puissance favorise le choix des nœuds voisins proches de i. $\delta_{j,proj_j}$ représente la distance perpendiculaire séparant j de sa projection $proj_j$ sur le segment d'un coordinateur donné. φ est un paramètre qui contrôle l'importance relative à $\delta_{j,proj_j}$. L'idée derrière l'incorporation de cette distance est de faire en sorte qu'une fourmi cherche autour du segment séparant sa source de sa destination et évite de s'orienter vers les nœuds voisins qui sont très loin de ce segment. De cette façon, sa portée de recherche est réduite en privilégiant, à chaque fois, les nœuds voisins qui peuvent assurer de bons chemins. Cela aide à accélérer la vitesse de convergence.

5.3.3.2 La sélection du prochain saut

Chaque Forward Ant qui est au niveau du nœud i et qui veut choisir son prochain saut n pour se connecter avec d ou un relais de d définit, en premier, l'ensemble des prochains sauts candidats CN_i. Cet ensemble consiste en l'ensemble des nœuds voisins de i qui sont plus proches à la position de la destination de cette Forward Ant par rapport à i.

Après la définition des voisins candidats pouvant être des prochains sauts, cette Forward Ant procède à la sélection de son prochain saut n. Si elle est non-déterministe, elle choisit n en utilisant un paramètre aléatoire q et un autre paramètre q_0. Ce dernier est un paramètre qui est compris entre 0 et 1 ($0 \leq q_0 \leq 1$), et q est un nombre aléatoire qui est distribué uniformément dans $[0,1]$. Si $q \geq q_0$, alors le système tend vers une exploration. Dans le cas contraire (i.e. $q < q_0$), il tend vers une intensification.

Dans le cas d'une exploration, une Forward Ant choisit son prochain saut en calculant les probabilités pour choisir chaque voisin $j \in CN_i$ comme suit :

$$p^d_{i,j} = \frac{[\tau(i,j,d)]^{\gamma}.[\eta(i,j)]^{\beta}}{\sum\limits_{k \in CN_i} [\tau(i,k,d)]^{\gamma}.[\eta(i,k)]^{\beta}} \tag{5.4}$$

Dans l'équation (5.4), $\tau(i,j,d)$ est l'intensité de la phéromone dont i maintient sur le lien qui le sépare de j par rapport à d. Une valeur de phéromone basique est ajoutée

aux voisins qui n'ont aucune entrée de phéromone correspondante afin d'augmenter la chance de leur sélection. $\eta(i,j)$ représente la variable heuristique qui est définie par l'équation (5.3) ; γ et β sont deux paramètres qui contrôlent respectivement, l'importance relative à $\tau(i,j,d)$ et $\eta(i,j)$.

En contre-partie et dans le cas d'une intensification, cette Forward Ant choisit le nœud voisin de i offrant l'intensité de phéromone maximale par rapport à d. Suivant ces deux règles, une Forward Ant qui est au niveau du noeud i choisit son prochain saut n parmi l'ensemble des candidats CN_i en utilisant la règle suivante :

$$n = \begin{cases} \arg\max_{j \in CN_i} \tau(i,j,d) & if\ q < q_0 \\ & ou\ fourmi\ déterministe \\ J & autrement \end{cases} \tag{5.5}$$

Dans cette équation, J est le nœud voisin qui est choisi suivant les probabilités qui sont calculées par l'équation (5.4).

5.3.3.3 La mise à jour de la phéromone et des coûts

Une fois qu'une Forward Ant arrive à son nœud de destination d ou un nœud relais de d, elle se transforme en une Bakward Ant. Cette dernière retourne à son coordinateur source en se basant sur les entrées ajoutées dans les tables de pistage des nœuds traversés lors de son trajet aller. De manière similaire à [7], une Bakward Ant utilise deux champs afin de mesurer le coût d'un chemin. Un premier *accCost* qui est initialisé à zéro et qui sert à mesurer le coût lors du trajet de retour. Un deuxième *localCost* qui permet aux autres nœuds de calculer le coût supplémentaire qui doit être ajouté à l'émetteur de la fourmi dans le cas où ce nœud est choisi comme relais. En fait, en considérant ce dernier champ, nous assurons une assignation des puissances de transmissions qui considère la bidirectionnalité des liens entre les nœuds de l'arbre de routage formé. En fait, lorsque l'agrégation est effectuée le long des chemins de routage, la perte des paquets est intolérable étant donné que ces paquets contiennent des informations pertinentes à différentes sources. Pour cela, assurer la bidirectionnalité des liens entre les nœuds constituant ces chemins donne l'opportunité à utiliser des mécanismes de tolérance aux pannes comme celui proposé dans [79] afin de garantir la délivrance fiable des données prélevées.

Lorsqu'un nœud i reçoit ou entend une Bakward Ant de son voisin j, il met à jour ses deux champs *accCost* et *localCost* comme suit [7] :

$$accCost' = accCost + linkCost + extraCost$$

$$localCost' = linkCost$$

où *linkCost* et *extraCost* sont définis comme suit :

$$linkCost = power_{i,j}/Power_{max}$$

$$extraCost = max(linkCost - localCost, 0)$$

Après cette opération, le nœud i met à jour sa table de phéromone et sa table best-cost suivant le type de la fourmi Backward Ant. Afin de mettre à jour ces deux tables, nous suivons le même processus de mise à jour proposé dans [7]. Deux raisons nous ont motivé pour adopter un tel processus. Premièrement et en ce qui concerne la mise à jour de la phéromone, il y a une focalisation sur le meilleur chemin qui est à chaque fois trouvé. Par conséquent, la recherche des fourmis est plus orientée autour de ce chemin. Deuxièmement, l'intensité de la phéromone est comprise entre 0 et 1. Cela permet d'éviter de tomber dans la situation de stagnation et favorise la diversification dans la sélection des chemins.

Algorithme 5.2 La mise à jour des entrées des tables de phéromone et best-cost du nœud i qui correspondent à son voisin j et à la destination d en utilisant $accCost'$.

```
1: if deterministic ant then
2:   cost(i,d) ⟵ accCost' ;
3:   τ(i,j,d) ⟵ τ(i,j,d) + 1/accCost' ;
4: else
5:   if accCost' < cost(i,d) then
6:     cost(i,d) ⟵ accCost' ;
7:     τ(i,j,d) ⟵ 1 ;
8:   else
9:     τ(i,j,d) ⟵ τ(i,j,d) + cost(i,d)/(ζ.accCost') ;
10:  end if
11: end if
```

L'algorithme 5.2 illustre ce processus de mise à jour. Dans le cas d'une Backward Ant non-déterministe et si un meilleur chemin est trouvé alors, le nœud i met à jour le coût du meilleur chemin déjà connu par le coût du ce nouveau chemin. En outre, l'intensité de la phéromone est mise à la valeur maximale autorisée qui est 1. Dans le cas contraire, c.à.d. si le coût du chemin trouvé n'est pas meilleur, alors et comme l'illustre la ligne 9 de l'algorithme 5.2, une petite valeur de phéromone est ajoutée. Dans cette ligne 9, ζ est un paramètre qui contrôle la quantité de phéromone ajoutée. Il est mis à 20 dans nos expérimentations.

D'autre part et dans le cas d'une Backward Ant déterministe, le nœud i met à jour l'entrée dans la table best-cost qui correspond à d par le nouveau coût calculé. Une certaine valeur de phéromone qui est inversement proportionnelle à ce nouveau coût est ajoutée. Par ailleurs, le nœud i devient relais. Il met à jour également l'ensemble qui consiste en son nœud parent et ses nœuds fils et ajuste sa puissance de transmission

de manière à atteindre le plus loin nœud de cet ensemble. Il à noter que cet ensemble est rincé périodiquement afin de s'adapter à la dynamique des routes.

Durant ce processus de mise à jour, la nature Broadcast de la communication sans fil est exploitée et toute Backward Ant entendue est utilisée afin de mettre à jour les différents tables. Cela aide à accélérer la dissémination des informations, et donc, à accélérer la recherche du meilleur chemin. Cependant, il est à noter qu'une Backward Ant est toujours considérée comme une fourmi non-déterministe lors de cette mise à jour.

En plus de cette exploitation de la nature Broadcast de la communication sans fil, l'intensité de phéromone doit être toujours comprise entre 0 et 1. Pour cette raison, la nœud i considère toujours la valeur minimale entre 1 et la valeur $\tau(i, j, d)$ nouvellement calculée ($\tau(i, j, d) \longleftarrow \min\{\tau(i, j, d), 1\}$).

Outre cette mise à jour, chaque nœud i diminue, chaque intervalle EVAPORATION_INTERVAL, l'intensité de la phéromone dans toutes les entrées de sa table de phéromone suivant l'équation 5.6. Cette évaporation donne plus d'importance aux informations les plus actualisées en rendant les anciennes informations moins significatives.

$$\tau(i, j, d)^{'} \longleftarrow (1 - \rho).\tau(i, j, d) \tag{5.6}$$

5.3.4 L'agrégation des données

Lorsque les données prélevées sont routées le long de la structure de routage formée, elles sont agrégées à deux niveaux :

1. **Agrégation Intra-cluster :** chaque collaborateur a le potentiel d'agréger les paquets routées. Ensuite, le nœud coordinateur agrège les paquets reçus de ses membres et envoie le résultat au nœud puits.
2. **Agrégation Inter-cluster :** lorsque les routes entre les différents coordinateurs se chevauchent, les nœuds qui connaissent ces chevauchements agrègent les données reçues et acheminent les données résultantes à leurs nœuds parents afin qu'elles soient routées vers le nœud puits.

5.4 Évaluation des performances

Dans cette section, nous évaluons les performances de notre protocole PALDA et ses deux variations. Pour cela, nous comparons leurs performances avec deux protocoles de routages connus : ECMANSI [7] et DST [9, 106].

ECMANSI est utilisé pour évaluer les performances de PALDA en ce qui concerne la phase d'établissement de l'arbre de routage qui relie les nœuds coordinateurs avec le

nœud puits. En fait, ECMANSI est un protocole qui est dédié au problème de multicast, donc au problème de l'arbre de Steiner, dans les réseaux ad hoc. Il vise à construire un arbre de multicast qui est enraciné au niveau du nœud source (nœud cœur) et qui couvre tous les nœuds destinataires, de manière à ce que, la somme des puissances de transmission des nœuds non feuilles soit minimisée. Comme PALDA, ECMANSI se base sur la métaphore de l'intelligence en essaim des colonies de fourmis. Dans ce travail, nous adaptons ECMANSI de la façon suivante : le nœud cœur qui représente la source des données dans la session de multicast est considéré comme nœud puits, alors que les nœuds destinataires sont considérés comme nœuds coordinateurs. En outre, les paquets de données circulent des nœuds coordinateurs vers le nœud puits et sont agrégés lorsque les routes se chevauchent.

D'un autre côté, DST est un protocole qui est très connu dans la littérature et qui a le même objectif de PALDA à la différence qu'aucun ajustement des puissances de transmission n'est adoptée. Il est également meilleur par rapport à d'autres protocoles tel que InFRA [88] et DRINA [79].

5.4.1 Le simulateur J-Sim

J-Sim [159] est un outil logiciel de simulation des réseaux informatiques. Il offre toute une plateforme de développement de nouveaux protocoles avec l'évaluation de leurs performances. J-Sim est construit sur la base d'une architecture logicielle à base de composants, baptisée ACA (Autonomous Component Architecture), qui imite de la conception des circuits intégrés (i.e. puces électroniques). Avec cette architecture, les entités basiques sont des composants qui sont dotés de ports spécifiques. La communication entre ces différents composants se fait par échange de données via leurs ports. Chaque composant reçevant une donnée exécute les fonctions appropriées immédiatement dans un contexte d'exécution indépendant (thread). L'architecture ACA a l'avantage d'offrir une architecture mettant en relation des composants qui sont faiblement couplées. Par conséquent, un composant peut être individuellement conçu, implémenté et testé. il peut même être réutilisé dans une autre application. En outre, une telle architecture facilite l'implémentation d'une conception inter-couches en sachant seulement comment faire connecter les ports des différents composants représentant les différents protocoles.

J-sim intègre, au-dessus de l'architecture ACA, un cadre applicatif (framework INET) intégrant les principales entités, avec leurs structures, pouvant constituer un réseau informatique et qui servent comme base pour l'implémentation de nouveaux protocoles.

Outre l'architecture ACA et le cadre INET, J-Sim offre des facilités pour intégrer différents langages de scripts tels que Perl, Tcl ou Python. Par défaut, J-Sim a été

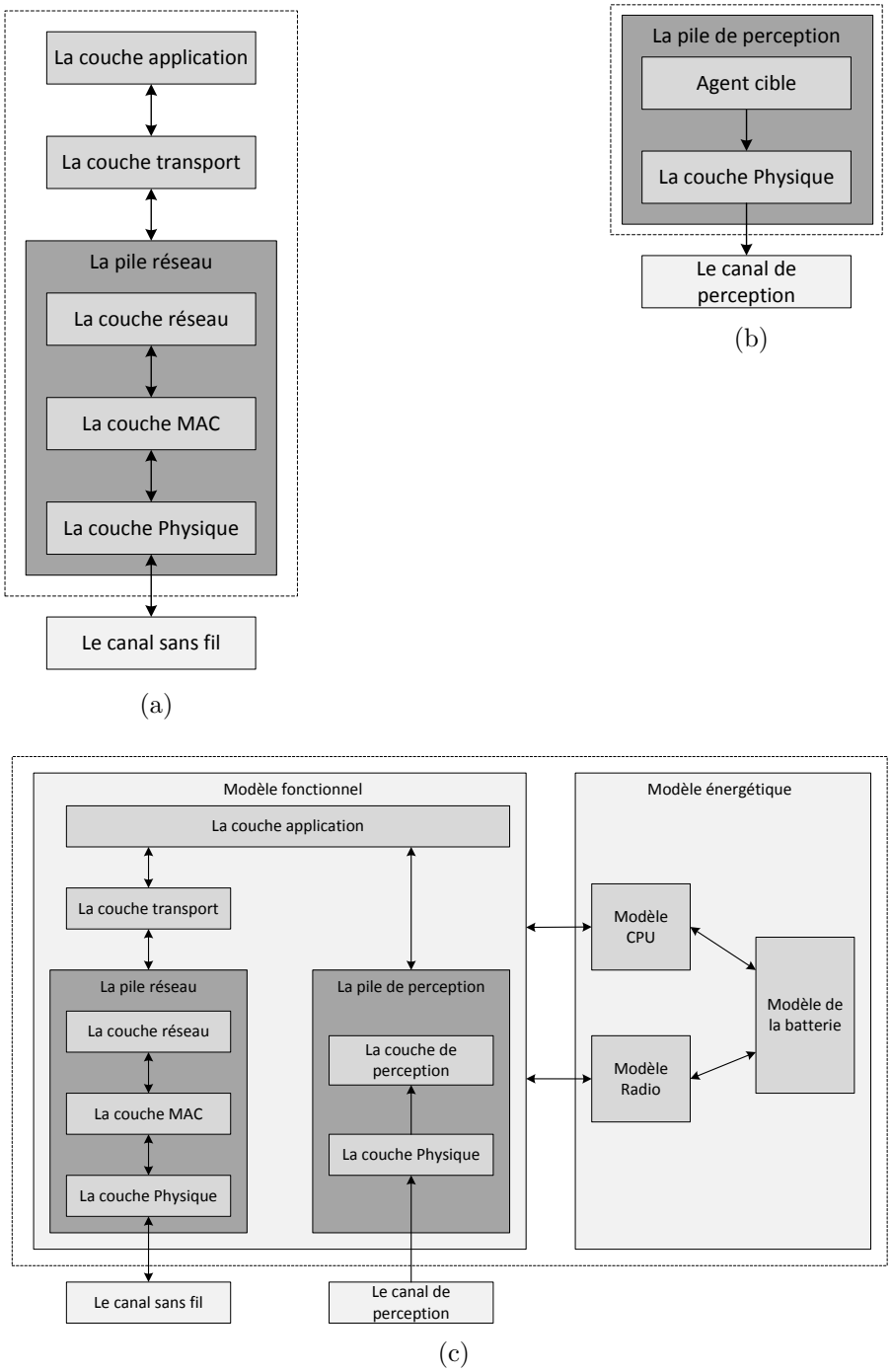


FIGURE 5.4 – Vue interne d'un nœud (5.4a) puits ; (5.4b) cible et (5.4c) capteur avec le cadre applicatif dédié aux WSNs dans J-Sim (les nœuds sont entourés par des traits pointillés).

intégré avec une implémentation Java de l'interpréteur TCL, appelé Jacl, avec une extension Tcl/Java qui définit un nouvel ensemble de commandes TCL. Ces nouvelles commandes permettent aux programmeurs de manipuler, directement, à partir d'un environnement Tcl, des objets Java tel que l'instanciation d'une classe, l'invocation d'une méthode d'un objet donné, etc. Donc, J-Sim met en opération deux langages informatiques. Un premier Java qui sert à développer les classes, donc les composants, qui implémentent nos conceptions de protocoles, de modèles réseaux, etc. Et un deuxième, Tcl/Java, qui sert à connecter les différents composants développés pour concevoir et mener ensuite les scénarios de simulation désirés.

Sur la base de ces principaux éléments de J-Sim et suite à l'engouement de la communauté scientifique vis-à-vis des WSNs, J-Sim a été doté d'un nouveau cadre applicatif dédié spécialement aux WSNs. En allant de la fonction principale d'un WSN qui est la surveillance de l'occurrence d'évènements d'intérêt, ce cadre définit trois types de nœuds :

1. un nœud cible qui génère les évènements ou les stimuli d'intérêt. Un véhicule mobile qui génère des vibrations au sol est un exemple d'un tel nœud,
2. un nœud capteur qui détecte ces évènements,
3. et un nœud puits qui collecte les données émises par les nœuds capteurs.

Comme le montre la figure (5.4c), un nœud capteur comporte deux piles protocolaires, une première de perception, qui lui permet de détecter les évènements produits par les nœuds cibles à travers le canal de perception et, une deuxième de communication, qui lui permet de communiquer avec les autres nœuds, capteurs et puits, et donc pouvoir acheminer les données prélevées vers le nœud puits à travers le canal sans fil. Différemment à un nœud capteur, un nœud cible comporte seulement une pile de perception (figure (5.4b)) alors qu'un nœud puits comporte seulement une pile de communication (figure (5.4a)). Pour plus de détails concernant ce cadre applicatif, le lecteur peut se référencer à [82].

Pour intégrer un nouveau protocole avec J-sim, il suffit seulement à implémenter les détails du protocole proposé comme un composant indépendant. Ensuite et lors de la définition des scénarios de simulation, il faut savoir le connecter avec les autres composants qui implémentent les autres protocoles dans les différentes couches. En suivant cette démarche, nous implémentons notre protocole proposé PALDA avec ses deux variations ainsi que les autres protocoles choisis pour des fins de comparaison (ECMANSI et DST).

5.4.2 Méthodologie

En utilisant le simulateur J-sim, nous conduisons une série de simulations afin d'évaluer les performances de PALDA. Le tableau 5.1 présente les paramètres par défaut.

TABLE 5.1 – Paramètres de simulation

Paramètre	Valeur
Nœud puits	1(Partie supérieure droite)
Nombre de nœuds capteurs	300
Portée de communication (m)	80
# évènements	3
Durée d'un évènement (heures)	3
La durée de la simulation (heures)	4
Intervalle de notification (sec)	10
Densité	20
Taille d'un paquet de contrôle (bytes)	1
Taille d'un paquet de données (bytes)	64
α (path loss exponent)	2
E_{elec}	$50\,nJ/bit$
ε_{amp}	$10\,pJ/bit/m^2$
φ	2
γ, β	1, 1
q_0	0.5
ρ	0.1
ANT_INTERVAL (sec)	3
EVAPORATION_INTERVAL (sec)	2

Pour certaines simulations, ces paramètres seront changés et les changements seront mentionnés dans la partie correspondante. Nous générons 33 différentes topologies où les nœuds sont uniformément déployés. Pour chacune d'elles, différents évènements sont générés à des positions aléatoires. Le premier évènement commence à l'instant 1000 s et les autres commencent à des instants qui sont uniformément distribués dans l'intervalle [1000, 3000] secondes. La densité du réseau est définie suivant la relation $n\pi r_c^2/A$, où n est le nombre de nœuds, r_c est la portée de communication et A l'aire du terrain de captage. Les dimensions de ce terrain sont ajustées suivant le nombre de nœuds considéré dans le scénario simulé afin de maintenir la même densité voulu. Avec PALDA-D, PALDA-S et ECMANSI, chaque coordinateur est autorisé à effectuer, au maximum, un nombre de recherches qui est égal à 150. Cependant, pour PALDA-D, un coordinateur réinitialise ce nombre à 0 lorsque son segment est changé. Les figures présentées dans ce qui suit ont été réalisées en faisant la moyenne des résultats de simulation obtenus depuis les 33 topologies générées, et tracées avec un intervalle de confiance de 95%. Tous les protocoles évalués utilisent la stratégie « periodic simple » [81] avec laquelle les nœuds agrégateurs transmettent périodiquement les données reçues et agrégées. Pour ECMANSI, nous utilisons les mêmes valeurs des paramètres qui sont utilisées dans [7]. Les différents protocoles sont évalués suivant les métriques suivantes :

1. **Le coût de l'arbre de routage :** ce coût représente la puissance totale de

transmission des arbres de routage finaux construit par chaque protocole.

2. **Le nombre des messages de contrôle :** représente le nombre des paquets de contrôle qui sont utilisés pour établir chaque structure de routage et assigner les puissances de transmission aux nœuds.
3. **Le taux d'agrégation :** représente le ratio entre le nombre total des paquets de données émis sur le nombre total des paquets de données reçus par le nœud puits.
4. **L'énergie totale consommée :** représente l'énergie totale qui est dépensée par chaque protocole. Elle inclue l'énergie dépensée lors de la transmission des paquets de données plus celle qui est dépensée par les messages de contrôle.

Pour la dissipation d'énergie, nous adoptons le modèle proposé dans [103]. Donc, l'énergie dissipée lors de la transmission d'un message de l-bits sur une distance δ est :

$$E_{TX}(l, \delta) = lE_{elec} + l\varepsilon_{amp}\delta^2$$

Et l'énergie dissipée en recevant un message de l-bits est :

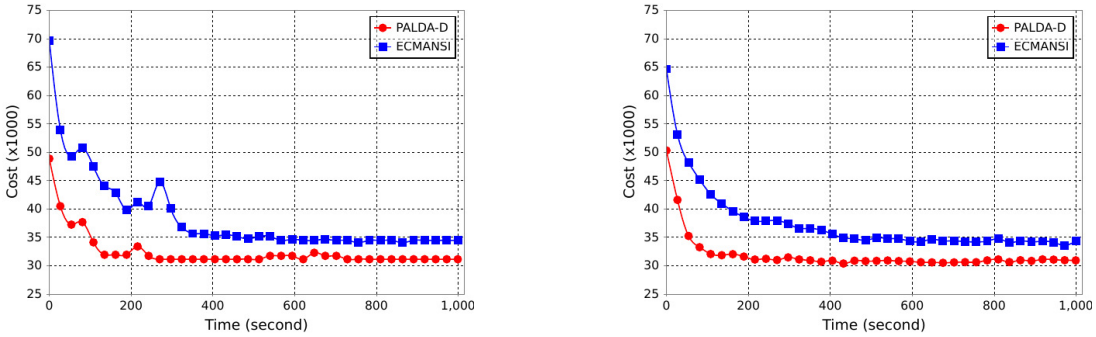
$$E_{RX}(l) = lE_{elec}$$

où E_{elec} est l'énergie par bit consommée par le capteur pour activer l'émetteur-récepteur. $\varepsilon_{amp}\delta^2$ est l'énergie par bit consommée par l'amplificateur. Dans notre modèle, nous ignorons l'énergie dissipée lors de la perception et du traitement par le fait qu'elle est moins importante que celle dissipée lors du transmission.

5.4.3 Vitesse de convergence

Ici, nous évaluons la vitesse de convergence de notre protocole. Pour cela, nous comparons PALDA-D avec ECMANSI en considérant l'évolution du coût de l'arbre de routage, établi par chacun d'eux, au fil de 1000 secondes de simulation (figure (5.5a) et (5.5b)). Dans ce scénario de simulation, nous considérons un réseau de 100 nœuds avec 6 évènements qui arrivent simultanément. Chaque évènement peut être détecté par un seul nœud capteur. Par conséquent, les scénarios évalués ici ne considèrent aucune clusterisation et donc aucun nœud collaborateur. Le nombre de recherches autorisées est ∞.

Comme le montrent les figures (5.5a) et (5.5b), nous constatons, pour chaque protocole, que le coût baisse, en continu, durant les premières secondes et se stabilise petit à petit à chaque fois que le temps avance. Cependant, nous constatons que PALDA-D converge plus rapidement par rapport à ECMANSI. En fait, avec PALDA-D, les agents fourmis, basent leurs recherches sur une structure géométrique sous-jacente.



(a) (b)

FIGURE 5.5 – Le coût de l'arbre de routage en fonction du temps. (a) La même topologie. (b) La moyenne de 33 différentes topologies.

Initialement, chaque coordinateur sait avec quel coordinateur est censé connecter et les fourmis sont orientées en suivant les segments de droite établis par leurs nœuds coordinateurs respectifs. Cela n'est pas le cas de ECMANSI où un arbre de routage initial qui ressemble à l'arbre du plus court chemin reliant les nœuds coordinateurs avec le nœud puits est initialement créé. Ensuite, les fourmis qui sont libérées par chaque coordinateur se déplacent dans toutes les directions afin de trouver un meilleur chemin qui connecte ce coordinateur avec le reste de l'arbre de routage.

Nous remarquons également à partir de ces deux figures que PALDA-D peut construire des arbres ayant de meilleurs coûts par rapport à ECMANSI. Cette supériorité est due en fait à la stratégie adoptée par PALDA-D pour combiner les chemins des nœuds coordinateurs et qui est meilleure par rapport à celle suivie par ECMANSI. En fait, ECMANSI introduit la notion de hauteur qui sert à éviter la situation où deux nœuds coordinateurs essayent de se connecter l'un avec l'autre et cesse d'être connecté avec le reste de l'arbre. Chaque nœud, relais ou coordinateur (s'il relaie les données d'un autre coordinateur), est associé avec une hauteur qui correspond au plus grand identifiant des nœuds coordinateurs se basant sur ce nœud pour se connecter avec le nœud puits. La hauteur de ce dernier est égale à ∞. Donc, un coordinateur est autorisé à se connecter directement soit, avec le nœud puits, soit, avec un nœud relais ayant une hauteur qui est supérieure à son identifiant. Suivant cette approche, nous pouvons dire que l'arbre final qui est construit par ECMANSI dépend des identifiants des nœuds coordinateurs et leurs positions. En contre-partie et avec PALDA-D, les coordinateurs les plus proches du nœud puits sont les premiers qui établissent leurs chemins. Cela aide à favoriser la minimisation du coût de l'arbre de routage.

5.4.4 Impact de la taille de l'évènement

Afin de montrer la supériorité de notre protocole en ce qui concerne la manière de clusterisation adoptée, nous évaluons ici l'impact de la taille de l'évènement. Pour cela,

nous comparons PALDA avec DST en considérant un seul évènement et en augmentant le rayon d'évènement de 20 m jusqu'à 100 m. Les résultats qui montrent l'énergie totale consommée par les nœuds qui détectent l'évènement sont présentés dans la figure (5.6).

Comme le montre la figure (5.6), nous constatons que l'énergie totale consommée augmente avec l'augmentation du rayon de l'évènement, car, plus de nœuds sources sont considérés. Cependant, nous constatons que PALDA est meilleur par rapport à DST. Cette supériorité est due au fait que les nœuds collaborateurs avec PALDA reportent leurs données en utilisant un arbre qui est énergétiquement efficace. Cela n'est pas le cas de DST où un arbre du plus court chemin est utilisé.

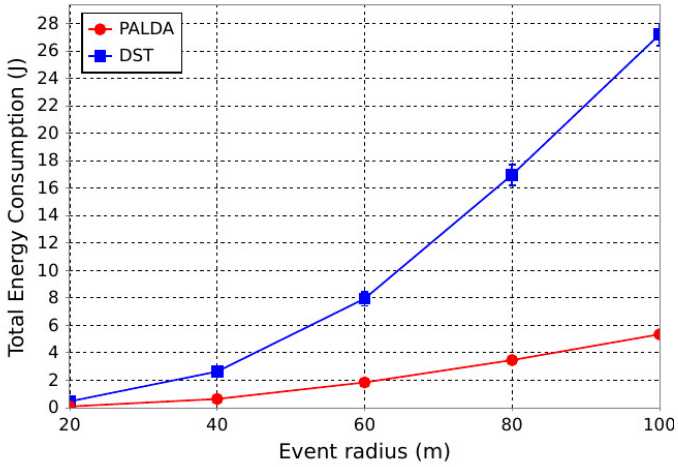
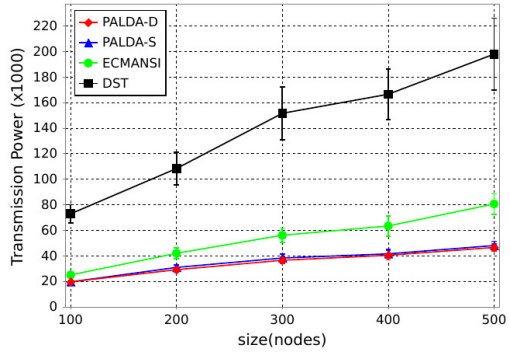


FIGURE 5.6 – L'impact de la taille de l'évènement.

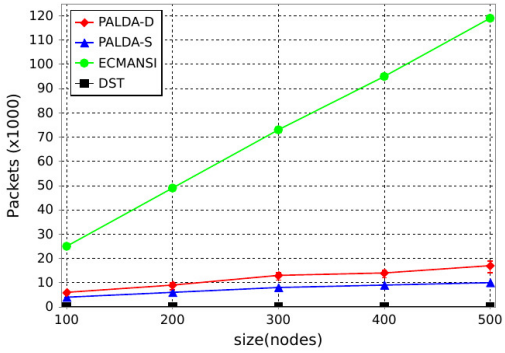
5.4.5 Impact de la taille du réseau

Dans ce scénario de simulation, nous évaluons l'impact de la taille du réseau sur les performances des protocoles. Pour cela, nous augmentons le nombre de nœuds de 100 à 500. De manière similaire à la vitesse de convergence et avec les deux scénarios qui sont évalués dans la suite de cette partie, chaque évènement est détecté par un seul nœud capteur. Par conséquent, aucune clusterisation n'est considérée. Nous procédons de cette façon afin de se focaliser seulement sur la phase d'établissement de l'arbre de routage entre les nœuds coordinateurs. Les résultats sont présentés dans la figure (5.7).

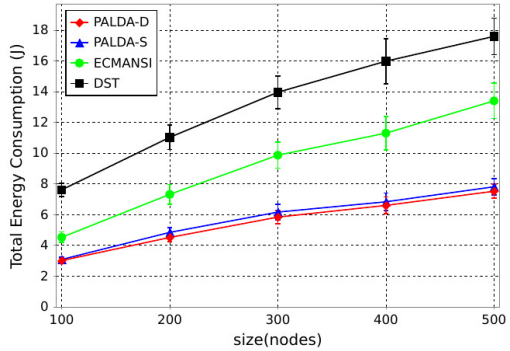
La figure (5.7a) montre que les coûts des arbres de routage augmentent avec l'augmentation du nombre des nœuds, car, les routes sont, à chaque fois, plus longues. Elle montre également que DST établit les mauvais coûts, car, aucun ajustement des puissances de transmission n'est réalisé. PALDA-D et PALDA-S établissent des arbres ayant de meilleurs coûts par rapport à ceux des arbres créés par ECMANSI. Et PALDA-D est légèrement supérieur par rapport à PALDA-S, car, les routes qui sont établies par PALDA-D ne dépendent pas de l'ordre d'occurrence des événements, mais par contre,



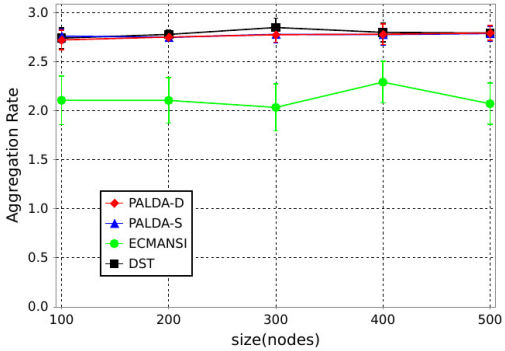
(a) Coût de l'arbre de routage.



(b) Nombre de paquets de contrôle.



(c) Énergie totale consommée.



(d) Taux d'agrégation.

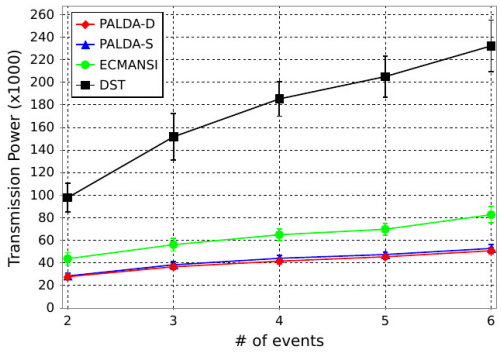
FIGURE 5.7 – Impact de la taille du réseau.

elles sont à chaque fois ré-établies afin d'améliorer le coût de l'arbre final de routage. Cependant, cette supériorité est accompagnée avec une dépense plus importante en ce qui concerne le nombre des messages de contrôle (figure (5.7b)). Dans ce sens, DST est celui qui dépense le moins de messages de contrôle et ECMANSI est celui qui dépense le plus. Cette importante dépense de ECMANSI est due en particulier, à l'utilisation, en plus des paquets Forward Ants et Backward Ants, d'autres paquets de contrôle qui sont périodiquement inondés sur l'ensemble du réseau. La figure (5.7d) montre le taux d'agrégation de chaque protocole. ECMANSI est celui qui a le taux le moins élevé car les autres protocoles ont la certitude de trouver des routes qui se chevauchent vu les stratégies qui sont suivies pour combiner les routes des nœuds coordinateurs. En termes de l'énergie totale consommée, la figure (5.7c) illustre des résultats qui sont, particulièrement, en correspondance avec les coûts des arbres finaux construits. Cette énergie est influencée également par le nombre des messages de contrôle dépensés par chaque protocole.

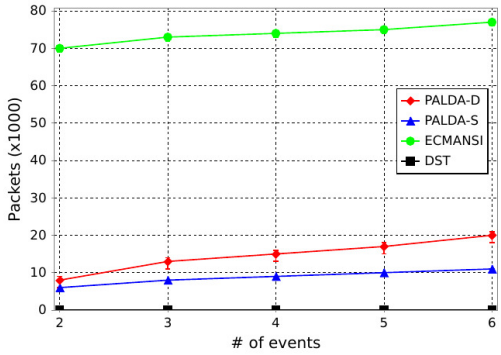
5.4.6 Impact du nombre des évènements

Dans ce scénario de simulation, nous évaluons le comportement des différents protocoles lorsque le nombre des évènements augmente. Pour cela, nous faisons varier ce

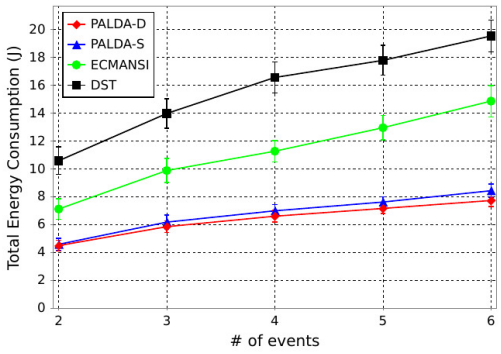
nombre de 2 à 6. Les résultats sont présentés dans la figure (5.8).



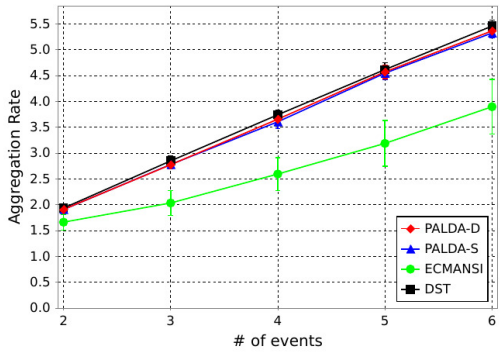
(a) Coût de l'arbre de routage.



(b) Nombre de paquets de contrôle.



(c) Énergie totale consommée.



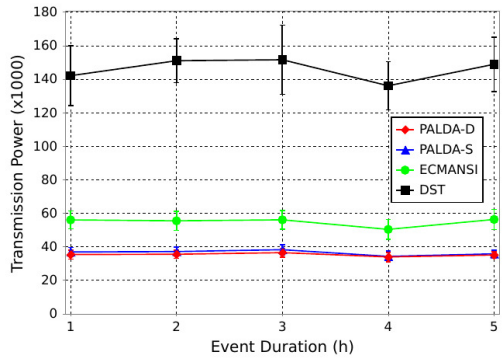
(d) Taux d'agrégation.

FIGURE 5.8 – Impact du nombre des évènements.

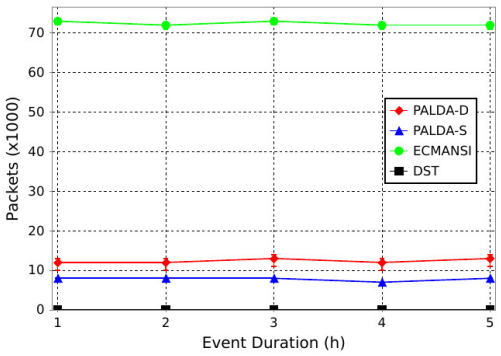
La figure (5.8a) montre que les coûts des arbres de routages qui sont construits par les différents protocoles augmentent avec l'augmentation du nombre des évènements, car, plus de routes sont à chaque fois considérées. Cette constatation est également vraie en ce qui concerne le nombre des messages de contrôle dépensé lors de l'établissement de l'arbre de routage et l'assignation des puissances de transmission (figure (5.8b)). Pour ce qui est du taux d'agrégation et comme le montre la figure (5.8d), nous constatons que ce taux augmente lorsque le nombre des évènements augmente, car, plus de données sont agrégées. La figure (5.8c) montre l'énergie totale consommée par chaque protocole. Cette figure montre une correspondance entre cette énergie et la puissance de transmission totale des arbres de routage formés(5.8a). Elle montre également l'influence du nombre des messages de contrôle.

5.4.7 Effet de la durée d'un évènement

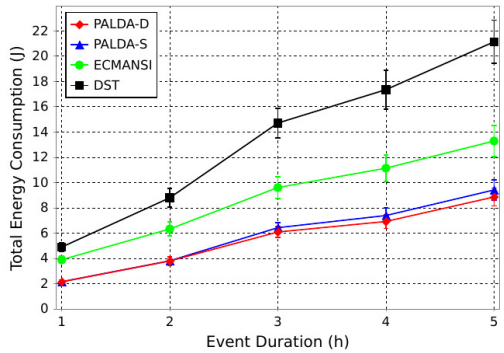
Dans ce scénario de simulation, nous évaluons l'impact de la durée des évènements sur le comportement des protocoles évalués en variant cette durée de 1 heure à 6 heures. Les résultats sont présentés dans la figure (5.9).



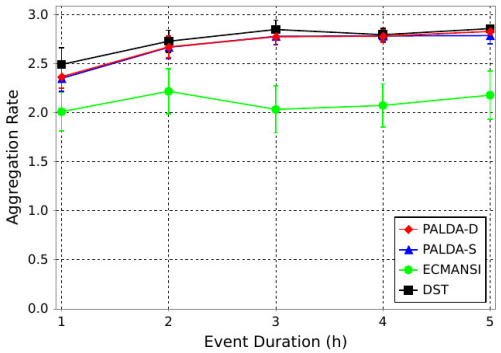
(a) Coût de l'arbre de routage.



(b) Nombre de paquets de contrôle.



(c) Énergie totale consommée.



(d) Taux d'agrégation.

FIGURE 5.9 – Impact de la durée de l'évènement.

De toute évidence, les puissances totales de transmission des arbres construits ((5.9a)), le nombre des messages de contrôle dépensés par chaque protocole ((5.9b)) et les taux d'agrégation achevés ((5.9d)) reste relativement à un même niveau, car, une même configuration est considérée (3 évènements et un réseau de 300 nœuds). En contre-partie, la chose la plus importante est pertinente à l'énergie totale consommée par chaque protocole lorsque la durée des évènements change. Comme le montre la figure (5.9c), nous constatons que lorsque la durée des évènements est courte (1 heure) la différence entre les différentes énergies consommées n'est pas très importante. Cependant, cette différence augmente et devient plus importante à chaque fois que la durée des évènements augmente. Cela est due à la différence entre les coûts des arbres de routage des protocoles évalués.

5.5 Conclusion

Dans ce chapitre, nous avons proposé le protocole PALDA qui est dédié aux réseaux de capteurs sans fil évènementiels. L'objectif de PALDA est de définir, une fois qu'un évènement ou plusieurs arrivent, une structure de routage qui maximise l'agrégation des données et dont la puissance totale de transmission de l'ensemble des nœuds constituant

cette structure soit minimale.

Pour arriver à cet objectif, nous avons suivi deux principales étapes. Durant la première étape et lors de la clusterisation, nous avons formé au sein de chaque cluster un arbre qui est énergétiquement efficace afin de collecter les données captées par les nœuds membres. Pour ce qui est de la deuxième étape qui correspond à la formation des routes entre les différents coordinateurs, nous nous sommes basés sur les algorithmes de colonies de fourmis afin de définir les nœuds relais et leurs puissances de transmission.

Notre protocole proposé PALDA a été largement comparé avec d'autres protocoles connus, ECMANSI et DST en considérant différents facteurs : le temps de convergence, la taille du réseau, le nombre des évènements et la durée des évènements. Avec la considération d'une structure géométrique sous-jacente et l'adoption d'une meilleure stratégie pour former les routes entre les différents coordinateurs, les résultats obtenus ont montré que PALDA est meilleur que ECMANSI. En plus, même si PALDA requiert plus de messages de contrôle par rapport à DST, les résultats ont montré qu'il le surpasse vu son avantage à ajuster les puissances de transmission des nœuds participant dans la collection des données.

Dans le suivant chapitre, nous visons à inclure la contrainte temps réel. En fait, une assignation agressive des puissances de transmission génère des routes avec un grand nombre de sauts, et donc, un temps de latence important. Pour cela, Il est nécessaire d'assurer un compromis entre le processus d'assignation des puissances de transmission et la délivrance en temps opportun des données captées.

Chapitre 6

Routage à délai borné et à basse consommation énergétique

6.1 Introduction

Comme nous l'avons vu dans le chapitre précédent, les nœuds capteurs qui participent au routage des données captées et agrégées ne sont pas obligés d'utiliser leurs puissances maximales de transmission, mais par contre, ils ont le potentiel de les minimiser afin de conserver davantage d'énergie. Cependant, cette minimisation devient contraignante lorsqu'on a à faire à des applications qui requièrent des communications en temps-réel. Cela est dû au fait qu'une minimisation agressive des puissances de transmission génère des routes avec un grand nombre de sauts, et par conséquent, un temps de latence important. Cette constatation est illustrée par la figure (6.1) qui montre deux structures de routages différentes connectant deux nœuds capteurs sources avec le nœud puits. Comme nous pouvons le voir, avec la deuxième structure de routage (figure (6.1b)) nous pouvons conserver plus d'énergie par rapport à la première structure (figure (6.1a)), étant donné que les nœuds capteurs utilisent des puissances de transmission plus petites. Cependant et comme conséquence, nous allons avoir des chemins avec un grand nombre de sauts.

Dans ce chapitre, nous nous intéressons au problème de la collection efficace et en temps réel des données prélevées par un certain nombre de nœuds capteurs sources. Nous proposons un nouveau protocole, baptisé DPIDA[10] (*Delay-bounded and Power-efficient routing for In-network Data Aggregation*), qui vise à assurer un compromis entre l'énergie qui est consommée durant cette collection et la délivrance, à délai borné, des données captées. Pour arriver à cet objectif, DPIDA réalise conjointement, et en se basant sur la métaheuristique ACO, les opérations suivantes :

1. il établit une structure de routage qui maximise le chevauchement des routes, et donc, l'agrégation des données. Cette structure correspond à un arbre de Steiner.

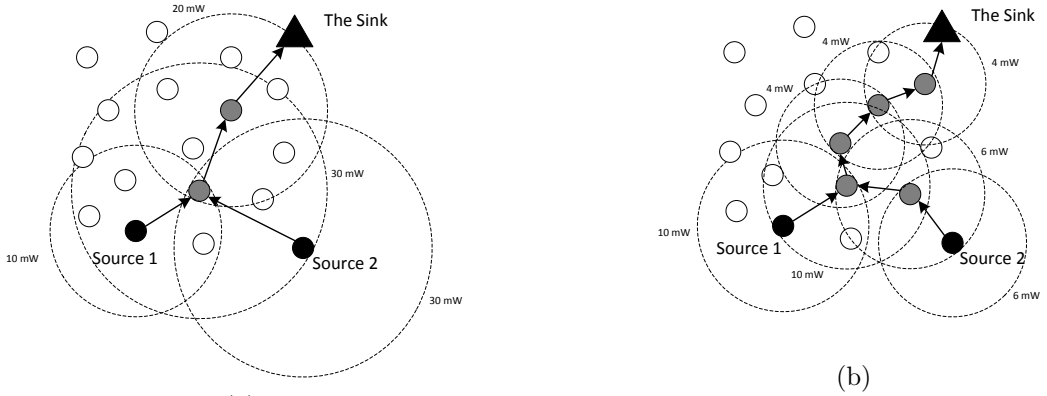


FIGURE 6.1 – Deux structures de routage différentes connectant deux nœuds capteurs sources avec le nœud puits et leurs énergies totales consommées. (6.1a) 90 mW avec une longue route de 3 sauts. (6.1b) 44 mW avec une longue route de 6 sauts.

2. il assigne à chaque nœud capteur, faisant partie de cette structure, une puissance de transmission appropriée.
3. il minimise la somme des puissances de transmission de tous les nœuds capteurs de la structure.
4. il établit des chemins à délais bornés.

Notre protocole DPIDA emprunte certaines idées d'un autre protocole appelé CST-MAN [11] et en introduit d'autres qui améliorent ce dernier en ce qui concerne plusieurs éléments. Autrement dit, nous proposons deux nouvelles stratégies qui définissent la façon dont les chemins des nœuds sources doivent être fusionnés. Ces deux stratégies ont une implication directe sur la qualité de la solution obtenue en termes de puissance de transmission totale de l'arbre final émergé. En plus, nous proposons une nouvelle stratégie afin de restreindre l'espace de recherche des fourmis en les orientant seulement vers la partie nécessaire de cet espace. Cela aide à accélérer la vitesse de convergence de la solution. Outre ces deux contributions, DPIDA diminue la charge engendrée par les messages de contrôle.

6.2 Modèle réseau et formulation du problème

Dans cette section, nous présentons le modèle réseau adopté et la formulation du problème traité.

6.2.1 Modèle réseau

Nous assumons notre modèle réseau comme suit :

1. Notre réseau est constitué d'un ensemble de nœuds capteurs statiques et un seul nœud puits.
2. Les nœuds capteurs peuvent ajuster leurs portées de communication en ajustant leurs niveaux de puissance de transmission, et cela , sans dépasser une puissance maximale $Power_{max}$ qui correspond à une portée maximale R_{max}.
3. Soit $G = \{V, E\}$ le graphe qui modélise la topologie de notre WSN où :
   (a) $V = \{v_1, v_2, ..., v_n\}$ est l'ensemble des sommets représentant les différents nœuds du réseau tel que $|V| = n$ et v_1 est le nœud puits .
   (b) Et E l'ensemble d'arcs (liens) qui relient ces nœuds et qui est défini comme suit :
   $$E = \{(u, v) | u \in V \, , \, v \in V \; et \; \delta_{u,v} \leq R_{max}\}$$
   Sachant que :
   $$\delta_{u,v} = \sqrt{(x_u - x_v)^2 + (y_u - y_v)^2} \tag{6.1}$$
   représente la distance euclidienne entre les deux nœuds u et v , avec $u = (x_u; y_u)$ et $v = (x_v; y_v)$.
4. Soit $S = \{s_1, s, ..., s_m\}$ l'ensemble des nœuds sources qui captent et collectent les données de l'environnement, tel que $|S| = m$ et $S \subseteq (V - \{v1\})$.
5. De manière similaire à [158], la puissance de transmission nécessaire pour que le nœud i communique avec j est calculée en assumant la formule :
   $$power_{ij} = \delta_{ij}^{\alpha} \tag{6.2}$$
   où α est l'exposant d'affaiblissement de propagation *« path loss » et* δ_{ij} est la distance entre i et j.

6.2.2 Définition du problème

Soit un ensemble de nœuds capteurs sources qui captent et doivent délivrer leurs données au nœud puits dans les délais requis. Dans ce travail, l'objectif est de définir une structure de routage qui maximise l'agrégation des données (i.e. maximiser le chevauchement des routes) avec une assignation des puissances de transmission aux nœuds de cette structure. Cette assignation doit minimiser la somme de ces puissances tout en assurant que les délais de transmission des données le long des chemins qui séparent chaque source du nœud puits ne dépassent pas une certaine borne temporelle donnée. Ce problème correspond au problème de l'arbre de Steiner qui est défini comme suit :

Étant donné :

1. le graphe $G = \{V, E\}$ qui modélise la topologie de notre réseau avec un nœud puits $v_1 \in V$ et un ensemble $S \subseteq (V - \{v1\})$ de nœuds sources,
2. Une fonction de délai d définie pour chaque élément de V, $d : V \rightarrow R^+$,
3. Et une contrainte de latence Γ,

Trouver : le sous-graphe $G' = \{V', E'\}$ de G, où $V' = S \cup L \cup \{v_1\}$ avec L représente l'ensemble des nœuds relais [1], et une assignation des puissances de transmission p : $V' \rightarrow R^+$ tel que :

1. G' est un arbre qui est enraciné au niveau du nœud puits v_1 et qui couvre tous les nœuds sources dans S.
2. $\sum_{v \in G' \, et \, v \neq v_1} power(v)$ est minimisée, où $power(v) = max_{u \in N_v} power_{v,u}$ avec $N_v = \{u | u \in V' \, et \, e_{v,u} \in E'\}$ l'ensemble des voisins de v dans G'.

Sous la contrainte :

$\forall s \in S, \sum_{v \in path(s,v_1) \, and \, v \neq v_1} d(v) \leq \Gamma$, où $path(s, v_1)$ représente l'ensemble des nœuds qui constituent le chemin ayant comme origine s et comme extrémité v_1.

6.3 Description de DPIDA

DPIDA vise à minimiser, autant que possible, l'énergie qui est dissipée lors de la collection des données tout en assurant leur délivrance avant les dates limites imposées. Pour cela, DPIDA procède à un processus d'assignation de puissances de transmission afin d'établir un arbre de Steiner avec des nœuds relayeurs dont la somme de leurs puissances de transmission est minimale, et avec des chemins qui sont bornés en délai. Dans notre travail, le délai d'un chemin entre un nœud source et le nœud puits correspond au nombre de sauts de ce chemin (i.e. $d(v) = 1$). Pour arriver à cet objectif, DPIDA procède suivant les trois phases suivantes :

- **Phase 1 :** dans cette phase, chaque nœud capteur sauvegarde les coordonnées du nœud puits et découvre ses voisins et leurs positions.
- **Phase 2 :** lorsqu'un nœud capteur détecte un évènement, i.e. devient un nœud source [2], il informe le nœud puits de sa position. Le nœud puits lui renvoie les coordonnées des autres nœuds sources qui existent. Il informe également ces derniers sources des coordonnées de ce nouveau nœud.

1. Un nœud relais est tout nœud faisant partie de l'arbre établi mais il n'est ni nœud feuille ni nœud racine. Un tel nœud peut être une nœud source.
2. Pour des raisons de simplicité, nous supposons dans notre travail qu'un évènement est détecté par un seul nœud capteur.

- **Phase 3 :** les différents nœuds relayeurs avec leurs puissances de transmission sont définis, donc, l'arbre qui connecte les nœuds sources avec le nœud puits dans les délais imposés est établi. Ce processus est réalisé en se basant sur la métaheuristique de colonies de fourmis.

DPIDA base son fonctionnement sur certaines structures de données qui sont maintenues localement sur chaque nœud du réseau. Les principales structures de données qui sont définies au niveau d'un nœud i sont comme suit :

1. **hauteur$_i$ (h) :** représente la hauteur du nœud i. Elle correspond à l'ordre [3] du nœud source qui compte sur i pour l'acheminement de ses données.
2. **Table des sources :** cette table est utilisée afin qu'un nœud capteur source puisse prendre connaissances des positions des autres nœuds sources. Une entrée de cette table consiste en les éléments suivants : $< id, SrcCoord >$. id correspond à l'identifiant du nœud source et $SrcCoord$ représente ses coordonnées.
3. **Table de phéromone :** une entrée de cette table consiste en les éléments suivants : $< j, h, \tau(i,j,h) >$. $j \in N_i$ où N_i représente les nœuds voisins de i, h correspond à une certaine hauteur et $\tau(i,j,h)$ représente l'intensité de la phéromone sur le lien (i,j) en ce qui concerne la hauteur h.
4. **best-cost table :** une entrée de cette table consiste en les éléments suivants : $< h, cost(i,h) >$. Elle est utilisée afin que le nœud i puisse prendre conscience du coût du meilleur chemin qui le sépare d'un autre nœud de l'arbre de routage ayant la hauteur h.
5. **Delay-From-Sink table :** cette table sauvegarde les délais qui séparent i du nœud puits via ses différents voisins. Une entrée de cette table est de la forme : $< j, delay(i,j) >$ où $j \in N_i$ et $delay(i,j)$ correspond au délai qui sépare i du nœud puits via le voisin j.

6.3.1 Configuration du réseau

Durant cette phase, le nœud puits inonde ses coordonnées sur l'ensemble du réseau et chaque nœud capteur découvre ses voisins et les informe de sa position. Ce processus commence à partir du nœud puits qui diffuse un message de configuration du réseau (NCM). Ce dernier comporte les trois champs suivants : ID, CoordSender et Coord-Sink qui représentent, respectivement, l'identifiant du nœud qui diffuse ce message, les coordonnées de ce dernier nœud et les coordonnées du nœud puits. Chaque nœud capteur i reçevant ce message ajoute un nouveau voisin à son ensemble des voisins en sauvegardant son identifiant et ses coordonnées. Si le nœud i reçoit un message NCM

3. cette notion d'ordre sera détaillée dans la section 6.3.3

pour la première fois, il met à jour les champs du message NCM et le diffuse en utilisant sa puissance maximale de transmission. L'algorithme 6.1 illustre ce processus.

Algorithme 6.1 Configuration du réseau.

1: The sink broadcasts a Network Configuration Message (NCM) ;
2: **foreach** node i receiving NCM **do**
3: Node i adds a new neighbor to its neighbors set N_i using ID_{NCM} and CoordSender_{NCM} ;
4: *// If node i receives a NCM message for the first time, it updates its fields and*
5: *// rebroadcasts it. Otherwise, it discards it.*
6: **if** firstSending **then**
7: $\text{CoordSink} \leftarrow \text{CoordSink}_{NCM}$;
8: $\text{ID}_{NCM} \leftarrow \text{ID}_i$;
9: $\text{CoordSender}_{NCM} \leftarrow \text{Coordinates}_i$;
10: firstSending $\leftarrow$ false ;
11: Node i broadcasts the NCM message with the updated values ;
12: **end if**
13: **end for**

6.3.2 Notification des nœuds sources

Après une première phase où le nœud puits inonde sa position sur l'ensemble du réseau et chaque nœud capteur découvre ses voisins et sauvegarde leurs positions, le réseau reste inactif jusqu'à l'occurrence des évènements. Lorsqu'un nœud capteur détecte un évènement, donc devient source, il notifie le nœud puits de l'occurrence d'un évènement et l'informe de ses coordonnées. À la réception de la notification du nœud source, le nœud puits lui renvoie les positions des autres nœuds sources. Il informe également ces derniers des coordonnées de ce nouveau nœud source. Chaque nœud capteur source qui reçoit le message du nœud puits met à jour sa table des sources. Les informations qui sont stockées dans cette table sont exploitées durant la formation des routes et permettent à ces nœuds sources d'établir un certain ordre qui définit la façon avec laquelle les différentes routes de ces nœuds doivent être fusionnées.

6.3.3 Formation des routes et assignation des puissances de transmission

Durant cette phase, les nœuds sources forment les différentes routes qui les relient avec le nœud puits en définissant les nœuds relais et leurs puissances de transmission tout en tâchant d'assurer des chemins qui sont bornés en délais. Ce processus est réalisé en se basant sur la métaheuristique de colonies de fourmis. Donc, chaque nœud source libère chaque ANT_INTERVAL des paquets de contrôle, appelés Forward Ant, dont l'objectif est de trouver un chemin de coût minimal qui le relie avec le reste

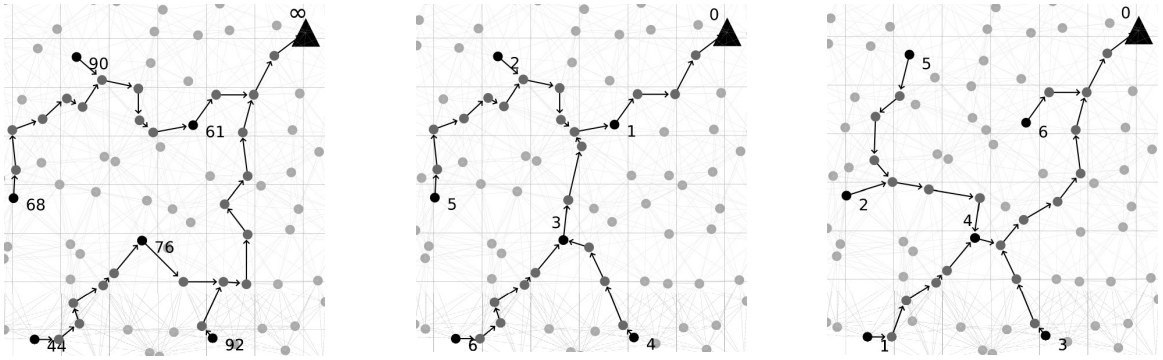
de l'arbre de routage. Comme CSTMAN, DPIDA met en opération deux types de fourmis : déterministe et non déterministe. Une fourmi déterministe suit toujours le prochain saut qui fait partie du meilleur chemin connectant son nœud source avec le reste de l'arbre. Son principal objectif est la définition des nœuds relais avec leur nœuds fils et parent, l'ajustement des puissances de transmission et le renforcement du meilleur chemin. En contre-partie, une fourmi non-déterministe détermine son chemin de manière probabiliste afin de découvrir de nouveaux chemins ayant des moindres coûts.

En se basant sur les informations stockées dans la table des sources, chaque nœud source ordonne les différents capteurs sources, y compris lui-même, sur la base de leurs distances du nœud puits. À la différence de CSTMAN qui utilise les identifiants des nœuds afin de définir la hauteur de chaque nœud, DPIDA utilise à la place l'ordre de chaque nœud source. Suivant cet ordre, deux variations de DPIDA sont proposées :

1. **DPIDA-CFV (Closest-First-Variation) :** dans cette variation, les nœuds capteurs sources sont ordonnés en ordre croissant sur la base de leurs distances du nœud puits.
2. **DPIDA-FFV (Farthest-First-Variation) :** à la différence de la première variation, les nœuds capteurs sources dans cette variation sont ordonnés en ordre décroissant sur la base de leurs distances du nœud puits.

Il est à noter que le nœud puits est toujours le premier dans l'ordre, vient ensuite les autres nœuds sources suivant la variation considérée. En définissant un tel ordre, chaque nœud source est autorisé à se connecter seulement avec les nœuds dont la hauteur est inférieure à son ordre. La figure (6.2) illustre les arbres construits par le protocole CSTMAN et nos deux variation proposées sans la considération d'aucune contrainte temporelle. Pour CSTMAN, le numéro qui est à côté de chaque nœud source représente son identifiant. Pour DPIDA-CFV et DPIDA-FFV, ce numéro représente son ordre. Comme le montre cette figure, les arbres obtenus sont complètement différents. Les résultats de simulation présentés dans la section (6.4) vont montrer que les coûts des arbres seront également différents.

Une Forward Ant qui est libérée par un nœud i est constituée des champs suivants : $order_i$, $allowedSources_i$, *accDelay*, *flag*, *accCost*, *costLimit*, *visitedNodes*. $Order_i$ correspond à l'ordre du nœud i parmi tous les nœuds capteurs sources. $allowedSources_i$ représente l'ensemble des sources avec leurs coordonnées et qui sont, dans l'ordre, avant le nœud i. Cet ensemble est utilisé afin de pouvoir restreindre l'espace de recherche de cette Forward Ant. *accDelay* est le délai qui est accumulé par cette Forward Ant depuis i jusqu'à son nœud courant. *flag, accCost, costLimit* et *visitedNodes* sont définis de manière similaire à CSTMAN. *flag* définit le type de la fourmi (déterministe ou probabiliste). *accCost* correspond au coût qui est accumulé depuis le nœud source d'une Forward Ant jusqu'à son nœud courant. *costLimit* est égal au coût du meilleur chemin



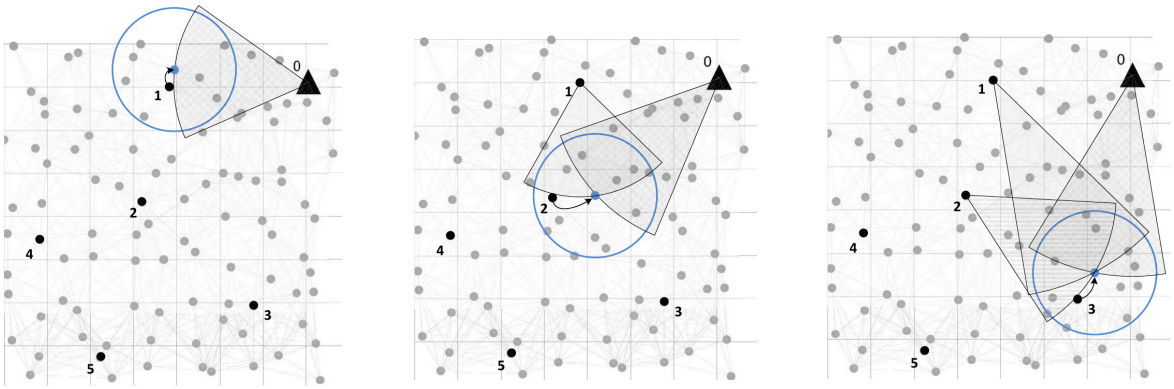
(a) (b) (c)

FIGURE 6.2 – Des instantanés qui illustrent les arbres de routages finaux construits par (6.2a) CSTMAN, (6.2b) DPIDA-CFV and (6.2c) DPIDA-FFV avec $\Gamma = \infty$.

connu connectant i avec le reste de l'arbre plus une certaine valeur. Ce champ accompagné de *accCost* préviennent la perte de cette Forward Ant. Ils sont utilisés seulement dans le cas d'une Forward Ant probabiliste. À la fin, *visitedNodes* est l'ensemble des nœuds qui sont traversés par une Forward Ant durant son voyage. Cet ensemble est utilisé afin d'éviter l'occurrence de boucles.

Durant leurs recherches, chaque Forward Ant explore le réseau afin de pouvoir trouver un chemin qui relie son nœud de départ avec le reste de l'arbre. La stratégie la plus simple, et comme procède CSTMAN, est de laisser les fourmis chercher dans tous les sens. Cependant, l'inconvénient en adoptant une telle stratégie est qu'une fourmi a une grande probabilité de se diriger vers une direction où aucun chemin ne peut être trouvé ou dans un sens qui l'emmène vers un chemin dont elle n'est pas autorisée à se connecter. Par exemple, dans la figure (6.2a), il n'y a aucune utilité de laisser les Forward Ant qui sont libérées par le nœud 90 d'aller vers la partie supérieure gauche. En outre, les Forward Ant qui partent du nœud 92 n'ont aucune utilité s'ils s'orientent vers la partie gauche car le nœud source 92 est autorisé à se connecter seulement avec le puits.

Afin d'éviter cette stratégie aveugle de recherche, DPIDA oriente les Forward Ants, en se basant sur les coordonnées des différents nœuds capteurs sources et celles du nœud puits, seulement vers la partie nécessaire du réseau. Chaque Forward Ant porte avec elle, dans le champ *allowedSources,* les positions des différents nœuds sources qui sont, dans l'ordre, avant le nœud du départ de cette Forward Ant. Autrement dit, les nœuds sources dont le nœud du départ de cette fourmi peut se connecter. Nous notons encore que le nœud puits a l'ordre 0, donc, n'importe quelle nœud source peut se connecter avec lui. Lorsqu'une Forward Ant est au niveau du nœud i et avant qu'elle sélectionne son prochain saut, elle détermine, parmi ses voisin N_i l'ensemble CN_i qui représente les



(a) (b) (c)

FIGURE 6.3 – Stratégie de restriction de l'espace de recherche. Une Forward Ant qui est libérée du nœud capteur source ayant l'ordre : (6.3a) 1, (6.3b) 2 et (6.3c) 3, et qui est au niveau du nœud capteur bleu restreint l'ensemble des nœuds voisins candidats à être des prochains sauts pour inclure seulement ceux qui l'emmènent vers la partie du réseau où elle peut se rencontrer, respectivement, avec : (6.3a) Le nœud puits directement ayant l'ordre 0, (6.3b) le chemin du nœud source ayant l'ordre 1 ou avec le nœud puits directement et (6.3c) les chemins des nœuds sources ayant les rangs 1 ou 2 ou avec le nœud puits directement.

voisins qui sont candidats à être des prochains sauts. Cet ensemble est déterminé en se basant sur les positions des sources dans son ensemble $allowedSources$, sur la position du nœud puits, sur le délai qu'elle a déjà accumulé $accDelay$ et sur les nœuds qu'elle a déjà visité et qui sont enregistrés dans sa liste $visitedNodes$. Il est défini comme suit :

$$CN_i = \cup_{k \in (allowedSources + \{sink\})} CN_i^k \tag{6.3}$$

où

$$\begin{aligned} CN_i^k \;=\; & \{j \mid \quad j \in N_i \\ & \quad et \quad \delta_{i,k} > \delta_{j,k} \\ & \quad et \quad delay(i,j) + accDelay \leq \Gamma \\ & \quad et \quad j \notin visitedNodes\} \end{aligned}$$

Comme défini dans la formule (6.3), l'ensemble CN_i regroupe, dans un premier temps, les voisins qui assurent l'avancement de cette fourmi vers seulement la partie du réseau où elle peut se rencontrer avec un nœud (relais, source ou puits) dont elle est autorisée à se connecter. Comme l'illustre la figure (6.3), une fourmi qui est libérée d'un nœud capteur source ayant l'ordre $order$ et qui se trouve au niveau d'un nœud donné (nœud bleu) restreint l'ensemble des voisins candidats pour inclure seulement ceux qui le ramènent vers la partie du réseau où elle peut se rencontrer avec un chemin d'un autre

nœud source ayant un ordre qui est inférieur à *order*. En adoptant une telle stratégie, les fourmis vont être guidées durant leurs recherches pour se focaliser seulement sur la partie du réseau qui est importante pour eux. Ce qui permet d'accélérer la vitesse de convergence de la solution. Outre cette stratégie de sélection, les voisins qui ne satisfont pas la borne temporelle et ceux qui ont déjà été visités sont éliminés de l'ensemble CN_i.

Après la définition des voisins candidats, cette Forward Ant sélectionne son prochain saut n. Si elle est non-déterministe, elle choisit n en utilisant un paramètre aléatoire q et un autre paramètre q_0. Ce dernier est un paramètre qui est compris entre 0 et 1 ($0 \leq q_0 \leq 1$), et q est un nombre aléatoire qui est distribué uniformément dans $[0, 1]$. Si $q \geq q_0$, la règle d'exploration est utilisée. Dans le cas contraire (i.e. $q < q_0$), la règle d'intensification est utilisée. Si Forward Ant est déterministe, le choix de n se fait toujours par l'application de la règle d'intensification.

Dans le cas d'une exploration, une Forward Ant dont l'ordre de son nœud de départ est *order*, sélectionne son prochain saut en calculant la probabilité pour choisir chaque voisin $j \in CN_i$ suivant l'équation suivante :

$$p_{i,j} = \frac{[d(i,j,order)]^{\gamma}.[\eta(i,j)]^{\beta}}{\sum_{k \in CN_i} [d(i,k,order)]^{\gamma}.[\eta(i,k)]^{\beta}} \tag{6.4}$$

où

$$d(i,j,order) = \max_{h<order} \frac{\tau(i,j,h)}{cost(i,h)} \tag{6.5}$$

et

$$\eta(i,j) = \frac{1}{power_{i,j}} \tag{6.6}$$

Dans (6.4), $d(i,j,order)$ représente la désirabilité à choisir le voisin j pour se connecter avec un nœud de l'arbre de routage qui a une hauteur inférieure à *order*. Elle est calculée suivant l'équation (6.5) en favorisant les voisins qui ont une intensité importante de phéromone et qui assurent des chemins ayant des moindres coûts afin de rejoindre l'arbre de routage. Une valeur de désirabilité basique est ajoutée à tous les voisins candidats afin d'augmenter la probabilité de sélection des voisins qui n'ont aucune entrée correspondante dans la table de phéromone de i. $\eta(i,j)$ est la valeur heuristique qui permet de favoriser les voisins dont les puissances de transmission nécessaires pour que i communique avec eux ne sont pas importantes. γ et β sont deux paramètres qui contrôlent respectivement, l'importance relative à $d(i,j,order)$ et $\eta(i,j)$.

Dans le cas d'une intensification, cette Forward Ant choisit le voisin qui offre la meilleure désirabilité.

Suivant ces deux règles, une Forward Ant qui est au niveau du nœud i choisit son prochain saut n parmi les voisins candidats CN_i en utilisant la règle suivante :

$$n = \begin{cases} \arg\max_{j \in CN_i} d(i, j, order) & si\ q < q_0\ ou \\ & fourmis\ déterministe \\ J & autrement \end{cases} \quad (6.7)$$

Dans cette dernière équation, J est le nœud voisin qui est sélectionné suivant les probabilités calculées par l'équation (6.4).

Il se peut qu'une Forward Ant ne trouve aucune entrée qui corresponde à une hauteur dont elle est autorisée à se connecter dans la table de phéromone du nœud i. Il se peut aussi qu'elle ne trouve aucun voisin qui satisfasse la contrainte de la délivrance à délai borné. Dans ces deux situations, une Forward Ant sélectionne le voisin le plus proche du nœud puits. Un tel voisin est toujours choisi lorsque les fourmis sont libérées pour la première fois. Par conséquent, ces fourmis vont établir un arbre initial qui est constitué, généralement, de chemins individuels connectant les nœuds sources avec le nœud puits. Avec le temps et avec le travail coopératif des fourmis, ces chemins vont être fusionnés de manière à avoir un nouvel arbre ayant un moindre coût.

Une fois qu'une Forward Ant arrive au niveau d'un nœud (i.e. relais, source ou puits) qui fait partie de l'arbre de routage et dont la hauteur est inférieure à l'ordre de son nœud de départ, elle vérifie tout d'abord si le délai depuis son nœud source jusqu'au nœud puits via ce nœud rencontré ne dépasse pas la borne temporelle imposée. Si ce n'est pas le cas, cette Forward Ant continue sa tâche de recherche. Dans le cas contraire, elle se transforme en une Bakward Ant. Cette dernière est constituée des champs suivants : *height, flag, accDelay, VisitedNodes, accCost, localCost. height* correspond à la hauteur du nœud au niveau duquel elle s'est convertie en une Backward Ant. *flag* détermine le type de cette fourmi. *accDelay* sert à la définition des délais qui séparent les différents nœuds recevant cette fourmi du nœud puits. Il est initialisé par $delay(i, parent_i)$ qui représente le délai qui sépare le nœud i, qui est le nœud au niveau duquel elle s'est convertie, du nœud puits via le nœud parent de i. *VisitedNodes* est l'ensemble des nœuds traversés par la Forward Ant correspondante. *accCost* est initialisé à zéro et sert à mesurer le coût du chemin qui sépare le nœud recevant cette fourmi du nœud de l'arbre rencontré lors du trajet de la fourmi Forward Ant correpondante. *localCost* permet aux autres nœuds de calculer le coût supplémentaire qui doit être ajouté à l'émetteur de la fourmi dans le cas où cet émetteur est choisi comme relais. Avec l'utilisation de ce dernier champ, nous assurons une assignation des puissances de transmissions qui considère la bidirectionnalité des liens entre les nœuds de l'arbre de routage formé. En fait, lorsque l'agrégation est effectuée le long des chemins de routage, la perte des paquets est intolérable étant donné que ces paquets contiennent des informations pertinentes à différentes sources. Pour cela, assurer la bidirectionnalité des liens entre les nœuds constituant ces chemins donne l'opportunité à utiliser des

mécanismes de tolérance de pannes comme celui proposé dans [79] afin de garantir la délivrance fiable des données prélevées.

Une Backward Ant emprunte le chemin inverse de sa Forward Ant correspondante en se basant sur son ensemble *VisitedNodes*. Lorsqu'un nœud i reçoit ou entend une telle fourmi de son voisin j, il met à jour les deux champs $accCost$ et $localCost$ comme suit [11] :

$$accCost' = accCost + linkCost + extraCost$$

$$localCost' = linkCost$$

où $linkCost$ et $extraCost$ sont définis comme suit :

$$linkCost = power_{i,j}/maxPower$$

$$extraCost = max(linkCost - localCost, 0)$$

Après cette opération, la valeur du champ *accDelay* du Backward Ant reçue est incrémenté à 1. Cette nouvelle valeur est utilisée, ensuite, par le nœud i afin de mettre à jour l'entrée $delay(i, j)$ dans sa table des délais. Cependant, il est à noter que cette mise à jour du délai est effectuée seulement si cette Backward Ant est déterministe. Lorsque le nœud i termine cette opération, il met à jour sa table de phéromone et sa table best-cost en faisant appel à la procédure *updatePathInformation* présentée par l'algorithme 6.2. Les valeurs des arguments de cette procédure sont comme suit : *last* correspond au nœud j ; *next* est le prochain saut de cette fourmi suivant son ensemble *VisitedNodes* ; *cost* correspond à $accCost'$; *height* et *flag* correspondent, respectivement, aux valeurs des champs *height* et *flag* de cette fourmi.

La mise à jour de la table de phéromone et la table best-cost se fait de manière similaire à CSTMAN. La seule différence est que DPIDA utilise les fourmis déterministes pour définir les nœuds relais et donc les routes qui sont utilisées pour l'acheminement des données. Comme le montre l'algorithme 6.2 et dans le cas d'une Backward Ant déterministe, le nœud i met à jour le coût qui correspond à la hauteur *height* par le nouveau coût calculé. Une certaine valeur de phéromone qui est inversement proportionnelle à ce nouveau coût est ajoutée. Le nœud i devient relais et met à jour l'ensemble qui est constitué de ses nœuds fils plus son nœud parent. Il ajuste également sa puissance de transmission de manière à atteindre le voisin le plus éloigné de cet ensemble. Il est à noter que ce dernier est rincé périodiquement afin de s'adapter à la dynamique des routes.

Dans le cas d'une Backward Ant non-déterministe et si le nœud i remarque qu'un meilleur chemin vers un nœud de l'arbre ayant une hauteur *height* est trouvé, le coût du meilleur chemin qui mène i vers ce nœud est mis à jour. En outre, l'intensité de la

Algorithme 6.2 Procédure *updatePathInformation(last, next, height, cost, flag)* exécutée par le nœud i.

1: **if** $flag = deterministic$ **then**
2: $cost(i, height) \longleftarrow cost$;
3: $\tau(i, last, height) \longleftarrow \tau(i, last, height) + \frac{1}{cost}$;
4: $role_i \longleftarrow relay$;
5: $parent_i \longleftarrow last$;
6: $children_i \longleftarrow children_i \cup \{next\}$;
7: **else**
8: **if** $cost < cost(i, height)$ **or** $cost(i, height)$ not defined **then**
9: $cost(i, height) \longleftarrow cost$;
10: $\tau(i, last, height) \longleftarrow 1$;
11: **else**
12: $\tau(i, last, height) \longleftarrow \tau(i, last, height) + \frac{cost(i,height)}{\zeta.cost}$;
13: **end if**
14: **end if**

phéromone est mise à la valeur maximale autorisée qui est 1. Contrairement et dans le cas où le nœud i ne remarque pas un meilleur chemin, une valeur minimale de phéromone est ajoutée.

Il est à noter que l'intensité de phéromone doit être toujours comprise entre 0 et 1. Pour cette raison, la nœud i considère toujours la valeur minimale entre 1 et la valeur $\tau(i, j, d)$ nouvellement calculée ($\tau(i, j, d) \longleftarrow \min\{\tau(i, j, d), 1\}$).

Nous soulignons le fait que cette mise à jour des tables se fait à chaque écoute d'une fourmi Backward Ant. Cette exploitation de la nature Broadcast de la communication sans fil, aide à accélérer la dissémination des informations, et par conséquent, l'émergence du meilleur chemin. Cependant, nous notons que la mise à jour de la table de phéromone et la table best-cost lors de l'écoute d'une fourmi Backward Ant se fait toujours de manière non-déterministe.

Outre ce processus de mise à jour, chaque nœud i diminue, chaque intervalle EVAPORATION_INTERVAL, l'intensité de la phéromone dans toutes les entrées de sa table de phéromone suivant l'équation (6.8). Cette évaporation donne plus de signification aux informations actualisées en rendant les anciennes informations mois importantes.

$$\tau(i, j, h) \leftarrow (1 - \rho).\tau(i, j, h) \tag{6.8}$$

Dans (6.8), ρ représente la fraction de la phéromone qui est évaporée.

TABLE 6.1 – Paramètres de simulation.

Paramètre	Valeur
Puits	1 (Partie supérieure droite)
Nombre de nœuds capteurs	100
Portée maximale (m)	80
Nombre de nœuds sources	6
Densité	20
γ, β, q_0 et ρ	1, 1, 0.5 et 0.1
α (path loss exponent)	2
Taille d'un paquet de contrôle (bytes)	1
Taille d'un paquet de données (bytes)	64
Temps de simulation (sec)	1000
Data rate (paquets/sec)	1
ANT_INTERVAL (sec)	3
EVAPORATION_INTERVAL (sec)	2

6.4 Évaluation des performances

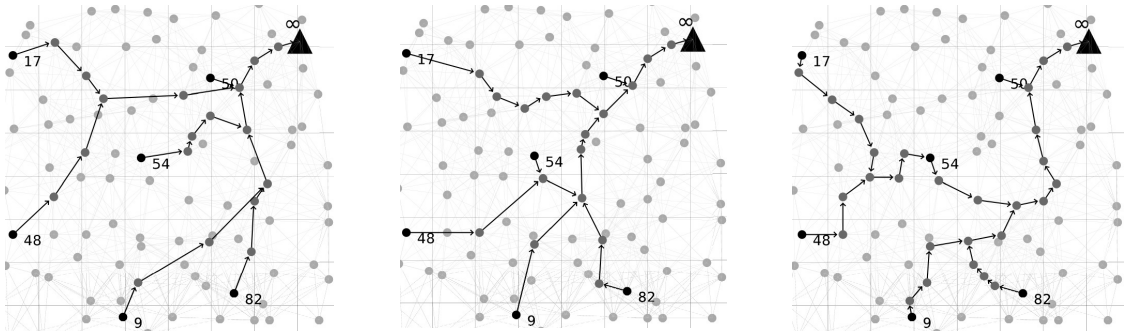
Dans cette section, nous évaluons les performances de notre protocole DPIDA avec ses deux variations en le comparant avec deux autres protocoles bien connus : CSTMAN[11] et DEDA[12]. Cette évaluation est effectuée en utilisant le simulateur J-sim. Les paramètres par défaut de simulation sont présentés dans le tableau (6.1). Pour certaines expérimentations, ces paramètres seront changés et ces changements seront mentionnés dans le section correspondante. Nous générons 33 différentes topologies aléatoires (différentes seeds). Pour chaque topologie, différents nœuds capteurs sources sont choisis aléatoirement. L'aire du terrain de captage est définie selon la relation suivante : $\sqrt{n\pi r_c^2/d}$ où n représente le nombre de nœuds capteurs, r_c représente le rayon de communication et d est le nombre moyen de voisins d'un nœud capteur (densité). La majorité des figures qui sont présentées dans la suite de cette section ont été réalisées en considérant la moyenne des résultats de simulation obtenus à partir des 33 topologies générées. Ces figures vont être montrées avec un intervalle de confiance de 95%. Pour CSTMAN et en ce qui concerne les valeurs des paramètres pertinents au protocole lui-même, nous utilisons les mêmes valeurs adoptées dans [11]. Les différents protocoles évalués ici sont comparés en considérant comme métriques :

1. **Le coût de l'arbre de routage :** ce coût correspond à la puissance totale de transmission de l'arbre final de routage qui est construit par chaque protocole.
2. **Nombre de paquets de contrôle :** Est le nombre de paquets de contrôle qui sont utilisés afin d'établir chaque structure de routage et assigner les puissances de transmission appropriées aux nœuds.
3. **L'énergie totale consommée :** représente l'énergie totale qui est dépensée par chaque protocole. Elle inclut l'énergie dissipée lors de la transmission des données

plus celle qui est consommée par les paquets de contrôle.

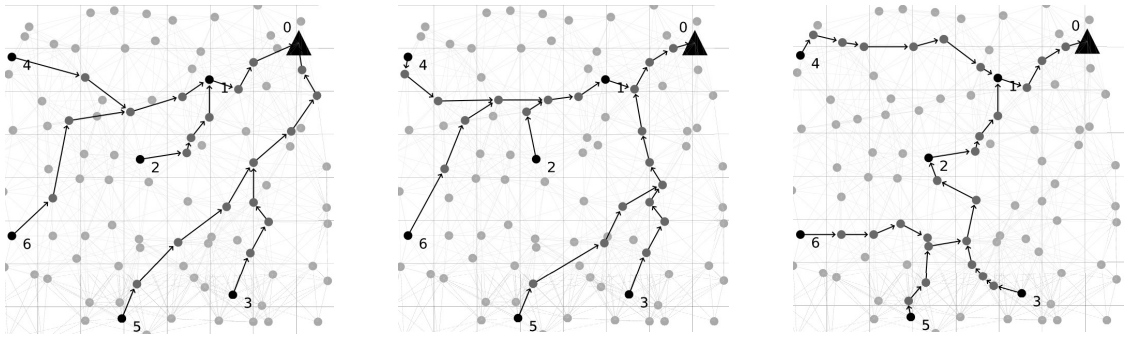
6.4.1 Impact de la borne temporelle

Dans un premier temps, nous validons visuellement la satisfaction de la borne temporelle en montrant pour chaque protocole évalué, et comme l'illustrent les figures (6.4), (6.5), (6.6) et (6.7), des instantanés qui montrent les arbres finaux de routages construits , respectivement, par CSTMAN, DPIDA-CFV, DPIDA-FFV et DEDA en considérant les trois bornes temporelles suivantes : 8, 10 et ∞. Pour pouvoir générer ces instantanées, nous avons intégré au simulateur J-sim, et en se basant sur la bibliothèque d'interface utilisateur JavaFx, notre propre interface graphique qui nous a permis de visualiser les différents arbres de routage élaborés par chacun des protocoles intégrés.



(a) (b) (c)

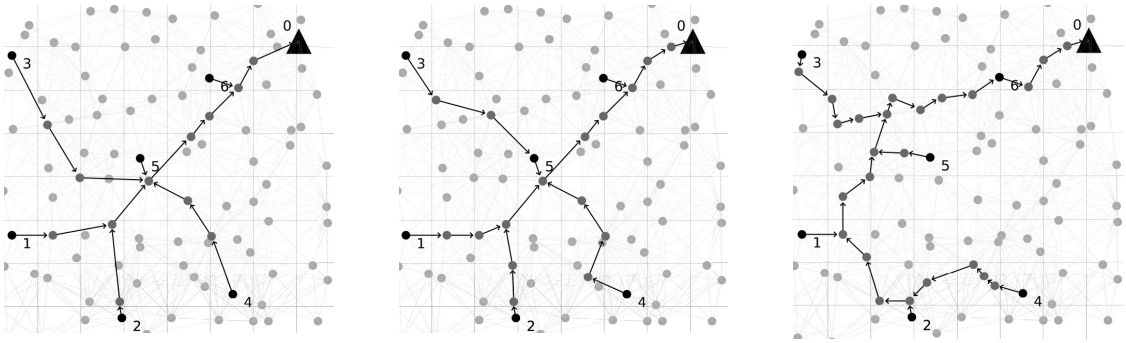
FIGURE 6.4 – Des instantanés montrant les arbres finaux de routage construits par CSTMAN avec Γ égal à : (6.4a) 8 sauts, (6.4b) 10 sauts et (6.4c) ∞ sauts.



(a) (b) (c)

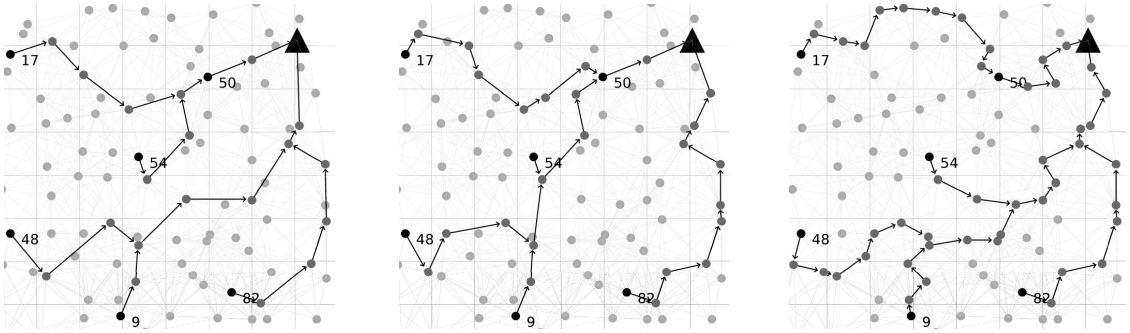
FIGURE 6.5 – Des instantanés montrant les arbres finaux de routage construits par DPIDA-CFV avec avec Γ égal à : (6.5a) 8 sauts, (6.5b) 10 sauts et (6.5c) ∞ sauts.

À partir de ces instantanés, nous pouvons constater que le nombre moyen de sauts des routes établies par un même protocole augmente à chaque fois que la borne tem-



(a) (b) (c)

FIGURE 6.6 – Des instantanés montrant les arbres finaux de routage construits par DPIDA-FFV avec Γ égal à : (6.6a) 8 sauts, (6.6b) 10 sauts et (6.6c) ∞ sauts.



(a) (b) (c)

FIGURE 6.7 – Des instantanés montrant les arbres finaux de routage construits par DEDA avec Γ égal à : (6.7a) 8 sauts, (6.7b) 10 sauts et (6.7c) ∞ sauts.

porelle augmente. Cela est dû aux stratégies adoptées par les différents protocoles afin de minimiser les puissances de transmission des nœuds faisant partie des structures de routage établies tout en assurant la satisfaction des bornes temporelles imposées. À partir de ces instantanés, nous pouvons également constater, pour chaque structure de routage construite, que le nombre de sauts des routes séparant chaque nœud source du nœud puits ne dépasse jamais la borne temporelle considérée. En outre, si nous comparons les différents arbres finaux de routage construits par les différents protocoles avec une même borne temporelle, nous pouvons voir qu'ils sont complétement différents. Les courbes qui sont présentées dans la suite de cette section vont montrer la différence en ce qui concerne leurs coûts.

Outre cette validation visuelle, nous montrons l'évolution du coût de l'arbre de routage établi par les fourmis, le long de 1000 secondes de simulation, en exécutant la variation DPIDA-FFV avec trois différentes bornes temporelles (Γ=8, 10 et ∞) et avec une même topologie. Comme le montre la figure (6.8), nous constatons, pour chaque borne que, le coût diminue continuellement durant les premières secondes et se stabilise

avec l'avancement du temps. Au début de la recherche, chaque fourmi libérée sélectionne le voisin de son nœud courant qui est le plus proche du nœud puits. Cela génère un arbre initial avec des nœuds qui utilisent des puissances de transmission importantes et qui est constitué de chemin individuels séparant chaque nœud source du nœud puits. Avec le temps, ces chemins sont fusionnés et de nouveaux nœuds qui utilisent des puissances de transmission moins importantes sont, à chaque fois, déterminés. Cela permet d'établir un nouvel arbre ayant un moindre coût. Une deuxième constatation qui est tirée de la figure (6.8), est que le coût de l'arbre final de routage est réduit à chaque augmentation de la borne temporelle. Cela montre l'adaptation dont DPIDA peut assurer avec la variation de cette borne.

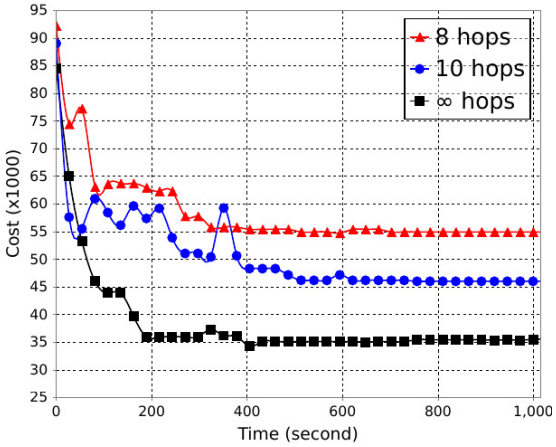


FIGURE 6.8 – L'évolution du coût de l'arbre de routage établi par les fourmis, le long de 1000 secondes de simulation, en exécutant la variation DPIDA-FFV avec trois différentes bornes temporelles (Γ=8, 10 et ∞) et avec une même topologie.

Dans une autre expérimentation, nous montrons le coût de l'arbre final de routage (l'arbre qui émerge) en exécutant chaque protocole avec différentes bornes temporelles. La figure (6.9) montre la moyenne des résultats de simulation obtenus en exécutant chaque protocole avec les 33 différentes topologies et avec ces différentes bornes temporelles. Comme nous pouvons le constater, lorsque cette borne est élevée, le coût de l'arbre final de routage qui est obtenu en exécutant la variation DPIDA-CFV est meilleur par rapport aux coûts obtenus suite à l'exécution de la variation DPIDA-FFV et les deux autres protocoles CSTMAN et DEDA. Cela est dû au fait qu'avec DPIDA-CFV, les nœuds sources qui sont les plus proches du nœud puits sont les premiers qui établissent leurs chemins, ce qui favorise la construction d'arbres de moindres coûts. Cela n'est pas le cas pour les deux autres protocoles où les nœuds sources ayant les plus grands identifiants et ceux qui sont les plus loin du nœud puits sont, respectivement, les premiers qui établissent leurs chemins avec CSTMAN et les premiers qui établissent leurs chemins avec DPIDA-FFV. Pour DEDA, nous pouvons constater qu'il établit des arbres ayant les coûts les plus élevés. Cela est dû au fait qu'il n'est pas conçu de ma-

nière à maximiser le chevauchement des routes. Et même si les routes se chevauchent, il ne se chevauchent pas assez tôt.

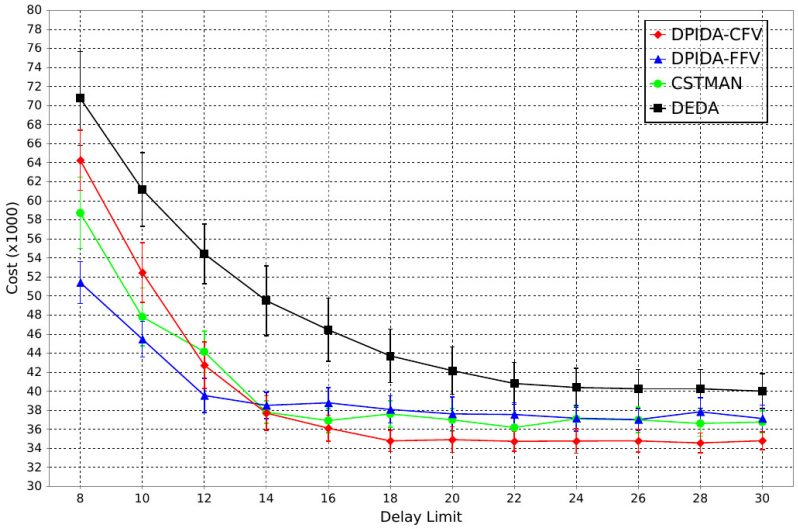


FIGURE 6.9 – la moyenne des coûts des arbres de routages construits en exécutant chaque protocole avec les 33 différentes topologies et avec différent Γ.

Lorsque la borne temporelle est basse et comme le montre la figure (6.9), nous constatons que la variation DPIDA-FFV donne de meilleurs résultats par rapport aux autres protocoles. Cela est expliqué en constatant les figures (6.4a) et (6.5a) qui montrent des instantanés qui illustrent les arbres de routage finaux établis par CSTMAN et DPIDA-CFV avec Γ=8. Suivant ces deux figures, nous pouvons constater que les nœuds sources qui sont loin du nœud puits ont la tendance à utiliser des chemins ayant des coûts importants afin de se connecter avec un nœud de l'arbre dont ils sont autorisés à se connecter avec et qui leur satisfait la contrainte de la borne temporelle. En revanche, avec DPIDA-FFV et comme le montre la figure (6.6a), les nœuds sources les plus loin du nœud puits établissent leurs chemins en premier. Cela donne l'opportunité aux autres sources de se connecter avec les nœuds de ces chemins qui sont les plus proches d'eux tout en satisfaisant la contrainte de la borne temporelle.

6.4.2 Vitesse de convergence

Dans cette section, nous présentons la vitesse de convergence de nos deux variations DPIDA-CFV et DPIDA-FFV et celle du protocole CSTMAN. Pour cela, nous montrons l'évolution du coût de l'arbre de routage construit par chaque protocole le long de 1000 secondes de simulation avec $\Gamma = \infty$. La figure (6.10) montre la moyenne des résultats de simulation obtenus en exécutant chaque protocole avec les 33 topologies générées. Comme illustré dans cette figure, les solutions obtenues avec DPIDA-CFV et DPIDA-FFV commencent à se stabiliser avant celles qui sont obtenues par CSTMAN. En outre, nous pouvons constater que CSTMAN met plus de temps afin d'arriver au même coût

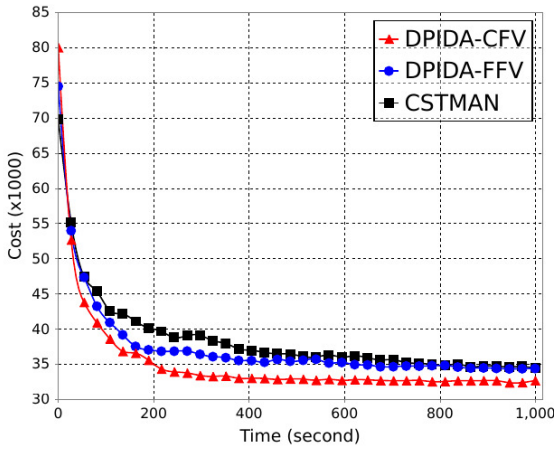


FIGURE 6.10 – La moyenne des coûts des arbres de routage construits par DPIDA-CFV, DPIDA-FFV et CSTMAN le long de 1000 secondes de simulation pour les 33 topologies générées et avec $\Gamma = \infty$.

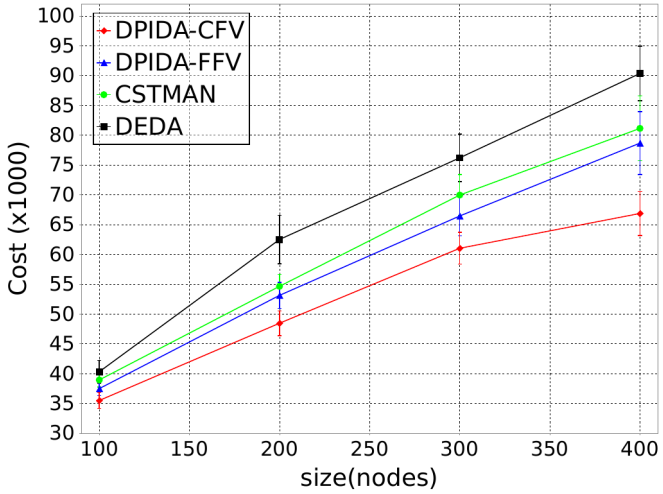
des arbres établis par DPIDA-FFV. Cela est dû à la stratégie suivie par DPIDA afin d'orienter la recherche des fourmis seulement vers les parties appropriées du réseau, ce qui aide à restreindre leur espace de recherche. Cela n'est pas le cas avec CSTMAN où les fourmis cherchent dans tous les sens, même dans les parties inappropriées.

6.4.3 Impact de la taille du réseau

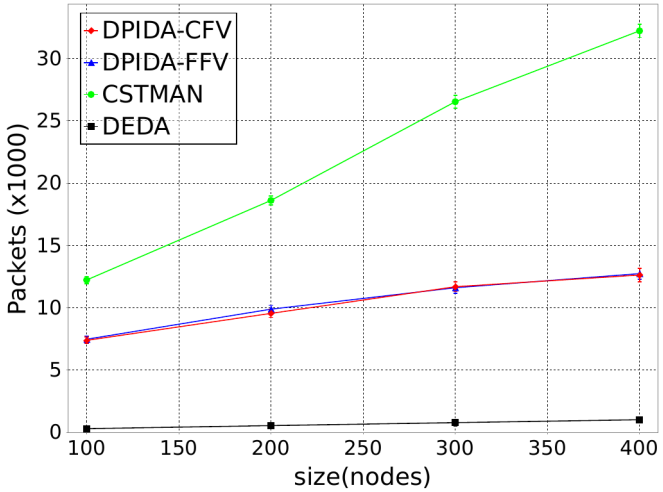
Dans ce scénario de simulation, nous évaluons l'impact de la taille du réseau sur les performances des différents protocoles évalués. Pour cela, nous augmentons le nombre de nœuds capteurs de 100 à 400 en considérant deux bornes temporelles (Γ) : 12 et ∞. Le choix de cette borne temporelle de 12 est dû à la raison suivante : en fait, avec un réseau de 400 nœuds et une densité de 20, nous obtenons un terrain de captage de $634 * 634\ m^2$. Avec une portée maximale de communication qui est égale à 80 m, il faut au minimum 12 sauts pour qu'un nœud capteur source puisse communiquer avec le nœud puits sur une distance de 909 m représentant la longueur du diagonale du terrain de captage. En considérant une borne de 12 sauts, nous assurons que n'importe quel nœud source puisse établir un chemin obéissant à cette borne, et cela qu'elle que soit sa position, lorsqu'un réseau de 400 nœuds est considéré.

Avec ce scénario et avec celui qui est simulé dans la section (6.4.4), et pour les trois protocoles CSTMAN, DPIDA-CFV et DPIDA-FFV, chaque nœud capteur source est autorisé à effectuer un nombre de recherche maximal qui est égal à 150.

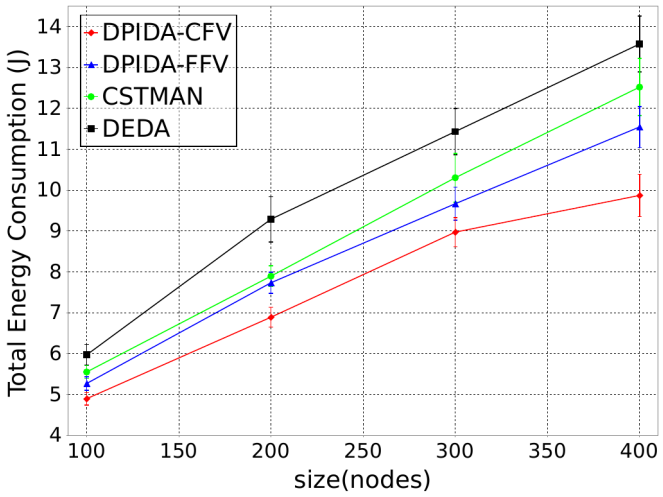
Pour une borne de ∞ et comme le montre la figure (6.11a), nous pouvons voir que DPIDA-CFV arrive toujours à construire les meilleurs arbres de routage. D'un autre côté, pour une borne de 12, et comme le montre la figure(6.12a), les arbres construits par DPIDA-FFV sont les meilleurs. Ces résultats montrent que nos deux variations proposées gardent la qualité des arbres établis même avec l'augmentation du nombre



(a) Le coût de l'arbre de routage.

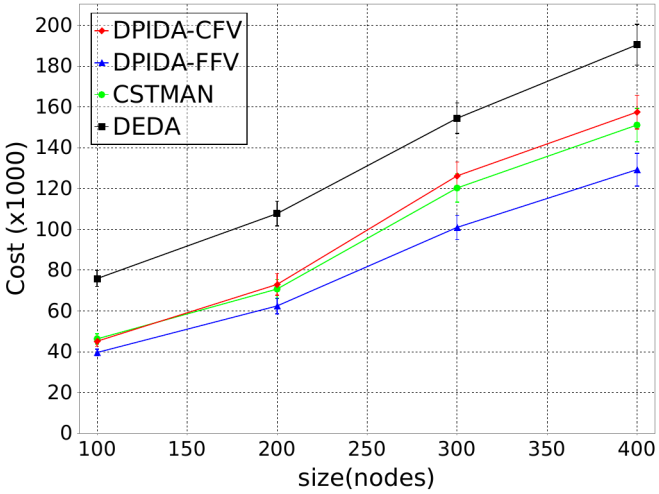


(b) Nombre de paquets de contrôle.

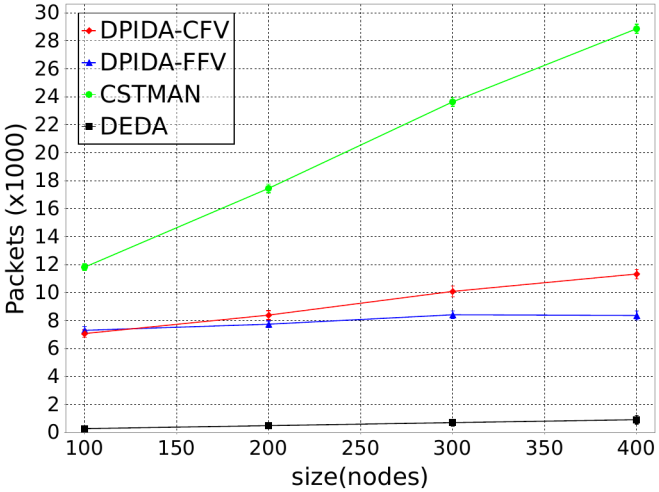


(c) L'énergie totale consommée.

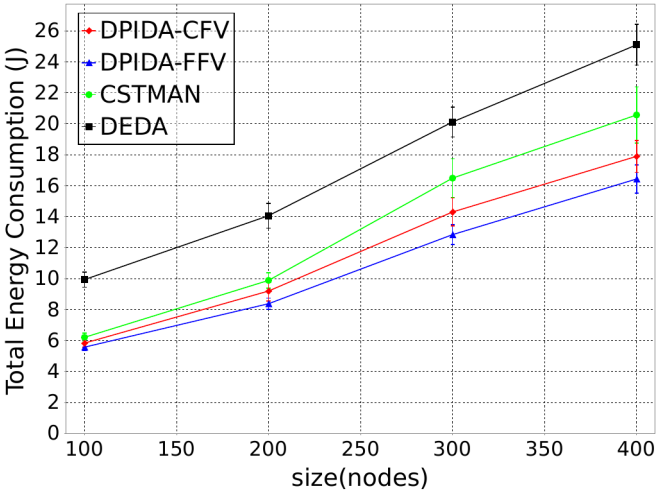
FIGURE 6.11 – Impact de la taille du réseau avec $\Gamma = \infty$.



(a) Le coût de l'arbre de routage.

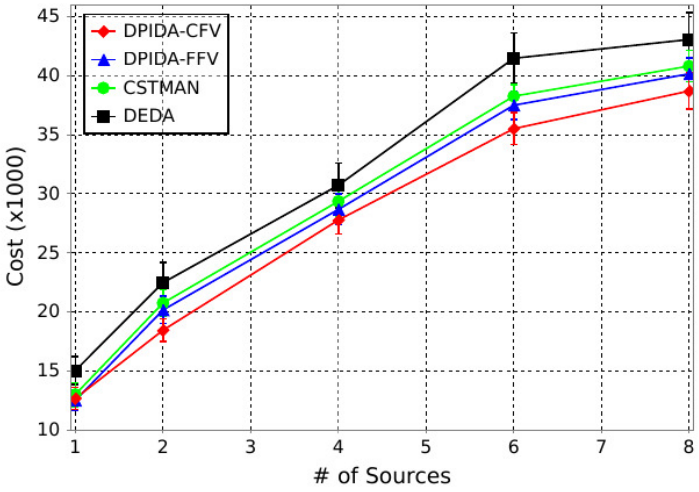


(b) Nombre de paquets de contrôle.

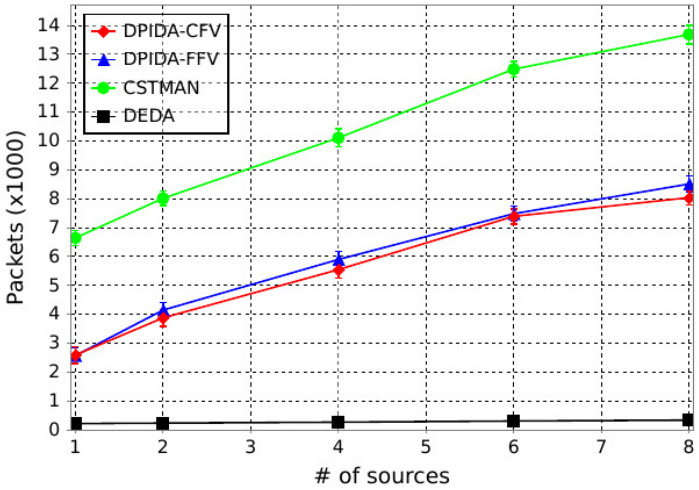


(c) L'énergie totale consommée.

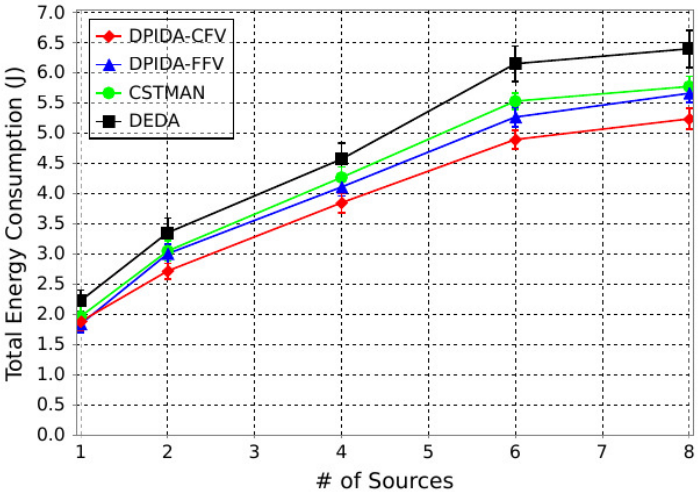
FIGURE 6.12 – Impact de la taille du réseau avec $\Gamma = 12$.



(a) Le coût de l'arbre de routage.

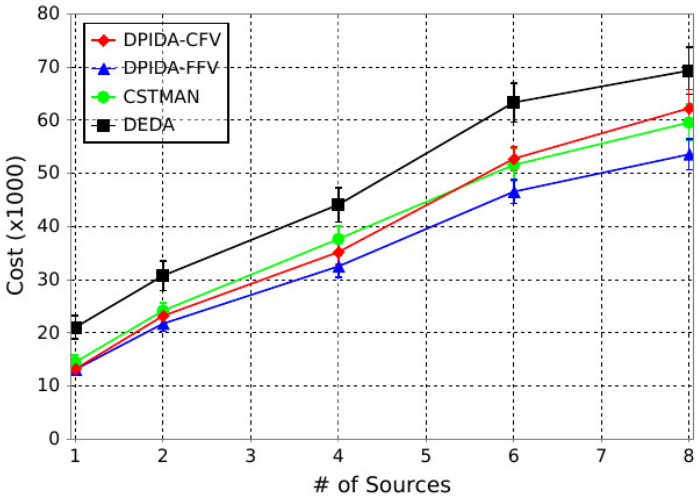


(b) Nombre de paquets de contrôle.

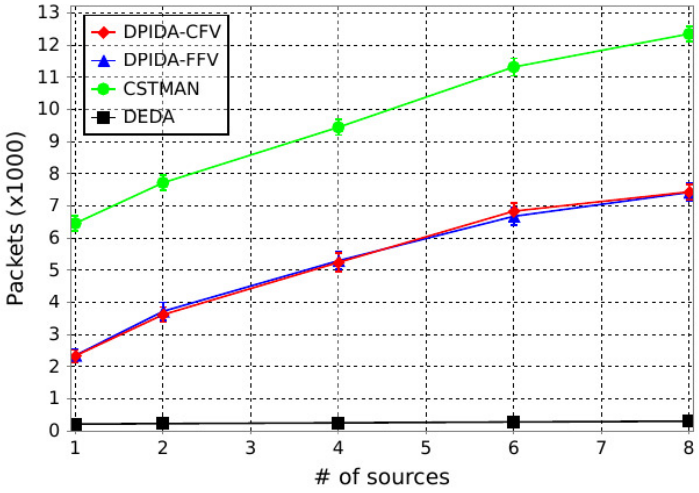


(c) L'énergie totale consommée.

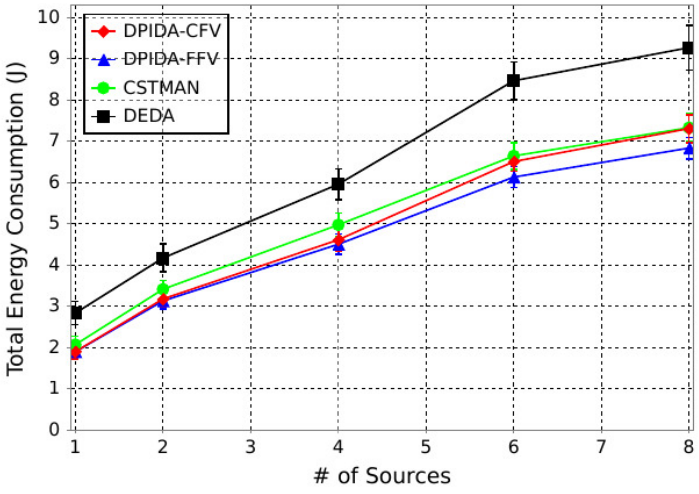
FIGURE 6.13 – Impact du nombre de nœuds sources avec $\Gamma = \infty$.



(a) Le coût de l'arbre de routage.



(b) Nombre de paquets de contrôle.



(c) L'énergie totale consommée.

FIGURE 6.14 – Impact du nombre de nœuds sources avec $\Gamma = 10$.

de nœuds du réseau. Outre ces résultats, les deux figures (6.11b) et (6.12b) montrent que notre proposition DPIDA est plus évolutif par rapport à CSTMAN, car, ce dernier requière plus de paquets de contrôle. Cette dépense importante est dû, en plus des paquets Forward Ant et Backward Ant, aux messages CORE ANNOUNCE qui sont inondés périodiquement sur l'ensemble du réseau et aux messages JOIN REQUEST qui sont envoyés en retour par les nœuds sources. Toujours à partir des deux figures (6.11b) et (6.12b), nous pouvons constater que DEDA requière moins de messages de contrôle. Toutefois, et comme le montre les deux figures (6.11c) et (6.12c), DEDA est moins efficace par rapport à notre proposition DPIDA, car, le coût des arbres construits par ce dernier est meilleur. Cette supériorité est dû au fait que notre protocole DPIDA favorise le chevauchement des routes, ce qui n'est pas le cas avec DEDA. Une dernière constatation intéressante que nous pouvons la constater à partir de la figure (6.12b) est que DPIDA-FFV utilise moins de paquets de contrôle en comparaison avec DPIDA-CFV à chaque augmentation dans le nombre de nœuds. Cela est expliqué par le fait que les fourmis avec DPIDA-CFV ont plus de difficulté pour se rencontrer avec un nœud approprié de l'arbre de routage.

6.4.4 Impact du nombre de nœuds sources

Dans ce scénario de simulation, nous évaluons le comportement des différents protocoles lorsque le nombre de nœuds capteurs sources augmente. Pour cela, nous augmentons le nombre de nœuds sources de 1 à 8 avec la considération des deux bornes temporelles suivantes : 10 et ∞. les figures (6.13) et (6.14) montrent les résultats de simulation pour, respectivement, $\Gamma = \infty$ et $\Gamma = 10$.

Comme illustré dans les deux figures (6.13a) et (6.14a), les coûts des arbres de routage construits augmentent avec l'augmentation du nombre de nœuds sources, car, plus de routes sont à chaque fois créées. Cette constatation est également vraie en ce qui concerne le nombre de paquets de contrôle utilisés (figures (6.13b) et (6.14b)). Pour l'énergie totale consommée et comme illustré dans les deux figures (6.13c) et (6.14c), nous constatons des résultats qui sont en correspondance avec les coûts des structures de routage établies.

6.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la collection efficace et en temps-réel des données captées par un ensemble de nœuds capteurs sources. Le but consistait à définir une structure de routage qui maximise le chevauchement des routes et dont la somme des puissances de transmission des nœuds de cette structure soit minimale. Tout cela en assurant des chemins qui sont bornés en délais. Notre protocole DPIDA

proposé ici est une amélioration du protocole CSTMAN qui vise à résoudre le même problème. Nos améliorations ont porté sur trois contributions principales. La première consistait en la proposition de deux nouvelles stratégies qui définissent la façon dont les chemins des nœuds capteurs sources doivent être fusionnés. Suivant ces deux stratégies, deux variations de DPIDA ont été proposées. Une première, DPIDA-CFV, qui donne un meilleur résultat lorsque la borne temporelle n'est pas importante. Et une deuxième, DPIDA-FFV, qui est meilleure lorsque cette borne est basse. Notre deuxième contribution consistait en la proposition d'une nouvelle stratégie afin d'orienter les fourmis seulement vers la partie nécessaire du réseau. Cela aidait à restreindre l'espace de recherche des fourmis, et par conséquent, à accélérer la vitesse de convergence de la solution. Outre ces deux contributions, notre protocole DPIDA est conçu de manière à minimiser la charge qui est engendrée par les messages de contrôle. Les résultats de simulation ont validé nos différentes propositions.

Conclusion

Dans cette thèse, nous nous sommes intéressés à l'étude d'un cas particulier d'objets coopératifs qui est le cas des réseaux de capteurs sans fil. La capacité de ces réseaux à prélever et à faire communiquer des mesures pertinentes à notre environnement physique a favorisé leur applicabilité à divers domaines de notre vie quotidienne. L'apport que présente ces réseaux s'accompagne, cependant, avec certains problèmes techniques qui entravent leur applicabilité. Parmi ces problèmes, le besoin à économiser leurs ressources énergétiques est certainement le plus important. En fait, les nœuds capteurs constituant ces réseaux sont dotés de batteries qui les alimentent durant leur fonctionnement. Ces nœuds sont censés fonctionner de manière autonome pour de longues durées sans que leurs batteries soient changées ou rechargées. Ce besoin à économiser cette énergie a motivé les chercheurs à proposer diverses techniques de conservation de cette énergie. Parmi les différentes techniques existantes, nous nous sommes intéressés dans cette thèse à deux techniques. La première est celle d'agrégation des données où nous nous sommes intéressés aux schémas de routage qui définissent la façon dont les données sont routées vers le nœuds puits en favorisant leur convergence spatiale afin que ces données puissent être agrégées. Et la deuxième est celle de contrôle des puissances de transmission des nœuds du réseau.

Le premier problème qui a été traité dans cette présente contribution a consisté à définir, une fois qu'un évènement ou plusieurs arrivent, une structure de routage qui maximise l'agrégation des données et dont la puissance totale de transmission de l'ensemble des nœuds constituant cette structure soit minimale. En fait, cette structure est un arbre de Steiner qui connecte l'ensemble des nœuds sources avec le nœud puits et dont la somme des puissances de transmission des nœuds capteurs sources et nœuds relais de l'arbre résultant soit minimal. Ainsi, cette assignation des puissances de transmission doit assurer la bidirectionnalité des liens entre les noeuds de cette structure afin que la collection des données soit faite de manière fiable. Avant de résoudre ce problème, nous avons étudié exhaustivement cette technique d'agrégation de données avec les différentes propositions qui s'intéressent à l'établissement de structures de routage facilitant l'application de cette agrégation. Ainsi, nous avons étudié les algorithmes de

colonies de fourmis, notre solution multi-agents pour résoudre ce problème d'agrégation, et les principaux travaux qui ont appliqué ces algorithmes à ce dernier problème. Après cette étude et afin de pouvoir établir la structure de routage désirée, nous nous sommes basés sur l'algorithme LMST et les algorithmes de colonies de fourmis afin de mener ce processus d'assignation des puissances de transmission et proposer, par conséquent, notre premier protocole PALDA. Notre protocole et les deux protocoles suivants : DST[9] et ECMANSI[7] ont été intégrés au simulateur J-sim pour des fins d'évaluation. Les résultats de simulation ont montré la supériorité de notre première proposition.

Dans un deuxième temps, nous nous sommes intéressés au problème de la délivrance, en délais bornés, des données prélevées par les nœuds capteurs. Cette délivrance, en temps réel des données, se contraste avec la conservation de l'énergie des nœuds capteurs. Par rapport au premier problème, la réduction agressive des puissances de transmission des nœuds génère des routes ayant un grand nombre de sauts qui augmentent le temps de latence des données. C'est pour cette raison que notre deuxième problème a consisté à assurer un compromis entre cette délivrance en temps réel des données et la dissipation de l'énergie des nœuds. Dans ce sens, nous avons proposé le deuxième protocole DPIDA qui ne vise pas qu'à maximiser l'agrégation des données et minimiser la somme des puissances de transmission des nœuds. Il vise également à établir des routes qui soient bornées en délais. DPIDA a été également intégré au simulateur J-sim avec les deux protocoles suivants : CSTMAN[11] et DEDA[12]. Des instantanées qui montrent les arbres d'agrégation construits par les différents protocoles évalués avec différentes bornes temporelles ont été aussi générées. Pour pouvoir générer ces instantanées, nous avons intégré aussi au simulateur J-sim, et en se basant sur la bibliothèque d'interface utilisateur JavaFx, notre propre interface graphique qui nous a permis de visualiser les différentes structures de routage élaborées par les différents protocoles intégrés. Les résultats de simulation avec ces instantanées ont montré que DPIDA peut assurer un compromis entre la délivrance en temps réel des données prélevées et l'énergie totale qui est consommée durant cette collection. Ainsi, ils ont montré sa supériorité par rapport aux deux protocoles de comparaison considérés.

Nos contributions proposées dans cette thèse peuvent être davantage améliorées selon divers volets. En fait, avec les deux protocoles PALDA et DPIDA, l'objectif a consisté, outre la maximisation de l'agrégation des données, à minimiser la puissance de transmission totale des nœuds constituant la structure de routage formée. En fait, si un nœud capteur donné qui a été fortement sollicité pour l'acheminement des données est à chaque fois sélectionné pour cette tâche de routage, il est fort probable que ses ressources énergétiques soient épuisées, ce qui cause sa panne. Comme conséquence, le réseau peut être partitionné, ce qui peut affecter défavorablement ses performances. Même si nous avons considéré implicitement cet équilibrage en consommation d'énergie

avec la version PALDA-D, nous n'avons pas considéré cet équilibrage comme un objectif principal de conception. Pour cette raison, il est intéressant de considérer, en plus des puissances de transmission des nœuds, les énergies résiduelles des nœuds capteurs afin de minimiser la charge sur les nœuds qui ont déjà connu une consommation énergétique importante. Ainsi, il est intéressant de considérer aussi l'énergie qui est consommée lors de la réception des messages qui circulent dans le réseau, car, comme nous l'avons vu au chapitre 2, l'énergie dissipée, suite à l'écoute abusive des messages des autres nœuds, peut également être importante.

Outre cette première amélioration, il est intéressant de réfléchir à minimiser le nombre de paquets de contrôle nécessaires au bon fonctionnement de nos deux protocoles proposés. En fait, avec nos deux implémentations des algorithmes de colonies de fourmis, nous avons modélisé, comme ce qui est communément appliqué, les agents fourmis comme des paquets intelligents, et donc des paquets de contrôle, qui circulent et explorent le réseau pour la recherche de bonnes solutions. En fait, la qualité de la solution trouvée est fonction du nombre d'agents fourmis libérés. En d'autres termes, les agents fourmis arrivent à trouver de meilleures solutions s'ils sont nombreux. Dans ce cas, l'application de ces algorithmes de colonies de fourmis peut être défavorable si le gain en énergie suite à l'application de ces algorithmes et moins important que l'énergie consommée suite à l'utilisation accrue des messages de contrôle. Avec nos deux implémentations, la minimisation du nombre d'agents fourmis libérés a été partiellement achevée en exploitant la nature broadcast du canal sans fil afin d'accélérer le processus de dissémination des informations de routage. Cependant, il est intéressant de réfléchir à minimiser davantage cette génération d'agents fourmis en les intégrant, par exemple, avec les paquets données comme ce qui a été déjà fait avec le protocole DA-ACA [5] et la famille d'algorithmes DAACA [8]. Outre la conservation de l'énergie des nœuds, cela aide même à avoir des protocoles qui évoluent parfaitement avec le nombre de nœuds du réseau tout en assurant l'objectif désiré.

Bibliographie

[1] P. J. Marrón, S. Karnouskos, and D. Minder. *Research Roadmap on Cooperating Objects*. 2009.

[2] M. Bagaa, Y. Challal, A. Ksentini, A. Derhab, and N. Badache. Data aggregation scheduling algorithms in wireless sensor networks : Solutions and challenges. *IEEE Communications Surveys Tutorials*, 16(3) :1339–1368, 2014.

[3] N. Benaouda and A. Lahlouhi. Power-efficient routing based on ant-colony-optimization and lmst for in-network data aggregation in event-based wireless sensor networks. *International Journal of Computing and Digital Systems*, 7(6) :321–336, 2018.

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

[5] W. H. Liao, Y. Kao, and C. M. Fan. Data aggregation in wireless sensor networks using ant colony algorithm. *Journal of Network and Computer Applications*, 31(4) :387 – 401, 2008.

[6] C. C. Shen and C. Jaikaeo. Ad hoc multicast routing algorithm with swarm intelligence. *Mobile Networks and Applications*, 10(1) :47–59, 2005.

[7] C. Jaikaeo, V. Sridhara, and Chien-Chung Shen. Energy conserving multicast for manet with swarm intelligence. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005.*, pages 9 pp.–732, 2005.

[8] C. Lin, G. Wu, F. Xia, M. Li, L. Yao, and Z. Pei. Energy efficient ant colony algorithms for data aggregation in wireless sensor networks. *Journal of Computer and System Sciences*, 78(6) :1686 – 1702, 2012.

[9] L. A. Villas, D. L. Guidoni, R. B. Araújo, A. Boukerche, and A. A. F. Loureiro. A scalable and dynamic data aggregation aware routing protocol for wireless sensor networks. In *Proceedings of the 13th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, pages 110–117, 2010.

[10] N. Benaouda and A. Lahlouhi. Ant-based delay-bounded and power-efficient data aggregation in wireless sensor networks. *International Journal of Pervasive Computing and Communications*, 15(2) :97–119, 2019.

[11] C. C. Shen, K. Li, C. Jaikaeo, and V. Sridhara. Ant-based distributed constrained steiner tree algorithm for jointly conserving energy and bounding delay in ad hoc multicast routing. *ACM Transactions on Autonomous and Adaptive Systems*, 3(1) :3 :27, 2008.

[12] X. Li, S. Yan, C. Xu, A. Nayak, and I. Stojmenovic. Localized delay-bounded and energy-efficient data aggregation in wireless sensor and actor networks. *Wireless Communications and Mobile Computing*, 11(12) :1603–1617, 2011.

[13] P.J. Marrón, D. Minder, and Embedded WiSeNts Consortium. *Embedded WiSeNts Research Roadmap*. Logos Verlag Berlin, 2006.

[14] P. J. Marrón, D. Minderl, and S. Karnouskos. *The Emerging Domain of Cooperating Objects : Definitions and Concepts*. Springer Briefs in Electrical and Computer Engineering. Springer, 2012.

[15] S. Karnouskos, P. Marrón, G. Fortino, L. Mottola, and J. R. Martinez de Dios. *Applications and Markets for Cooperating Objects*. Springer Briefs in Electrical and Computer Engineering. Springer, 2014.

[16] A. Sanfeliu, J. Andrade-Cetto, M. Barbosa, R. Bowden, J. Capitan, A. Corominas, A. Gilbert, J. Illingworth, L. Merino, J.M. Mirats, P. Moreno, A. Ollero, J. Sequeira, and M.T.J. Spaan. Decentralized Sensor Fusion for Ubiquitous Networking Robotics in Urban Areas. *Sensors*, 10 :2274–2314, 2010.

[17] R. Figura, C.Y. Shih, M. Ceriotti, S. Fu, F. Brockmann, H. Nebot, F. Alarcón, A. Kropp, K. Kondak, M. Schwarzbach, A. J. Viguria, M. Mulero-Pázmány, G. Dini, J. Capitán, and P. J. Marrón. Kassandra : A framework for distributed simulation of heterogeneous cooperating objects. *Journal of Systems Architecture*, 73 :28 – 41, 2017. Special Issue on Reliable Software Technologies for Dependable Distributed Systems.

[18] M. Banâtre, P. J. Marrón, A. Ollero, and A. Wolisz. *Cooperating Embedded Systems and Wireless Sensor Networks*. Wiley-IEEE Press, 1 edition, 2010.

[19] F. Dressler. *Self-Organization in Sensor and Actor Networks*. Wiley series in communications networking & distributed systems. Wiley, 2007.

[20] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks : a survey. *Computer Networks*, 38(4) :393 – 422, 2002.

[21] K. Holger and W. Andreas. *Protocols and Architectures for Wireless Sensor Networks*. Wiley-Interscience, New York, NY, USA, 2007.

[22] I. F. Akyildiz and I. H. Kasimoglu. Wireless sensor and actor networks : research challenges. *Ad Hoc Networks*, 2(4) :351 – 367, 2004.

[23] I. F. Akyildiz and M. C. Vuran. *Wireless Sensor Networks*. John Wiley & Sons, Inc., 2010.

[24] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He. The liteos operating system : Towards unix-like abstractions for wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, Washington, DC, USA, 2008. IEEE Computer Society.

[25] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, Washington, DC, USA, 2004. IEEE Computer Society.

[26] K. Sohraby, D. Minoli, and T. Znati. *Wireless Sensor Networks : Technology, Protocols, and Applications*. Wiley-Interscience, 2007.

[27] C. Cordeiro and D. Agrawal. *Ad Hoc and Sensor Networks : Theory and Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2nd edition, 2011.

[28] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5) :51–58, 2000.

[29] R. Kacimi. *Techniques de conservation d'énergie pour les réseaux de capteurs sans fil*. PhD thesis, Université de Toulouse - Institut National Polytechnique de Toulouse - INPT (FRANCE), 2009.

[30] T. Rault, A. Bouabdallah, and Y. Challal. Energy efficiency in wireless sensor networks : A top-down survey. *Computer Networks*, 67 :104 – 122.

[31] K. S. Deepak and A. V. Babu. Energy consumption analysis of modulation schemes in ieee 802.15.6-based wireless body area networks. *EURASIP Journal on Wireless Communications and Networking*, 2016(1), 2016.

[32] S. Cui, A. J. Goldsmith, and A. Bahai. Energy-constrained modulation optimization. *IEEE Transactions on Wireless Communications*, 4(5) :2349–2360, 2005.

[33] A. Nosratinia, T. E. Hunter, and A. Hedayat. Cooperative communication in wireless networks. *IEEE Communications Magazine*, 42(10) :74–80, 2004.

[34] Y. Zhang, H.H. Chen, and M. Guizani. *Cooperative Wireless Communications*. CRC Press, 2009.

[35] E. Kranakis, D. Krizanc, and E. Williams. Directional versus omnidirectional antennas for energy consumption and k-connectivity of networks of sensors. In *Principles of Distributed Systems*, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[36] G. P. Joshi, S. Y. Nam, and S. W. Kim. Cognitive radio wireless sensor networks : Applications, challenges and research trends. *Sensors*, 13(9) :11196–11228, 2013.

[37] J. Mitola and G. Q. Maguire. Cognitive radio : making software radios more personal. *IEEE Personal Communications*, 6(4) :13–18, 1999.

[38] E. Orumwense, T. Afullo, and V. Srivastava. Achieving a better energy-efficient cognitive radio network. *International Journal of Computer Information Systems and Industrial Management Applications*, 8 :205–2013, 03 2016.

[39] P. Santi. *Topology control in wireless ad hoc and sensor networks*. Wiley, 2005.

[40] M. A. Labrador and P. M. Wightman. *Topology Control in Wireless Sensor Networks : With a Companion Simulation Tool for Teaching and Research*. Springer Publishing Company, Incorporated, 2010.

[41] S. Randhawa and S. Jain. Data aggregation in wireless sensor networks : Previous research, current status and future directions. *Wireless Personal Communications*, 97(3) :3355–3425, 2017.

[42] J. Kho, A. Rogers, and N. R. Jennings. Decentralized control of adaptive sampling in wireless sensor networks. *ACM Transactions on Sensor Networks*, 5(3) :19 :1–19 :35, 2009.

[43] C. Alippi, G. Anastasi, M. Di Francesco, and M. Roveri. An adaptive sampling algorithm for effective energy management in wireless sensor networks with energy-hungry sensors. *IEEE Transactions on Instrumentation and Measurement*, 59(2) :335–344, 2010.

[44] M. Z. Farooqi, S. M. Tabassum, M. H. Rehmani, and Y. Saleem. A survey on network coding : From traditional wireless networks to emerging cognitive radio networks. *Journal of Network and Computer Applications*, 46(C) :166 – 181, 2014.

[45] R. C. Carrano, D. Passos, L. C. S. Magalhaes, and C. V. N. Albuquerque. Survey and taxonomy of duty cycling mechanisms in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 16(1) :181–194, 2014.

[46] S. Misra, M. P. Kumar, and M. S. Obaidat. Connectivity preserving localized coverage algorithm for area monitoring using wireless sensor networks. *Computer Communications*, 34(12) :1484 – 1496, 2011.

[47] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella. Energy conservation in wireless sensor networks : A survey. *Ad Hoc Networks*, 7(3) :537 – 568, 2009.

[48] H. Ba, I. Demirkol, and W. Heinzelman. Passive wake-up radios : From devices to applications. *Ad Hoc Networks*, 11(8) :2605 – 2621, 2013.

[49] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks : a survey. *IEEE Wireless Communications*, 11(6) :6–28, 2004.

[50] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14) :2826 – 2841, 2007.

[51] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo. A survey of network lifetime maximization techniques in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 19(2) :828–854, 2017.

[52] M. A. Rahman, S. Anwar, M. I. Pramanik, and M. F. Rahman. A survey on energy efficient routing techniques in wireless sensor network. In *2013 15th International Conference on Advanced Communications Technology (ICACT)*, 2013.

[53] I. Sharma and K.R. Ramkumar. A survey on aco based multipath routing algorithms for ad hoc networks. *International Journal of Pervasive Computing and Communications*, 13(4) :370–385, 2017.

[54] X. Cheng, D.-Z. Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks. *Wireless Networks*, 14(3) :347–355, 2008.

[55] J. Tang, B. Hao, and A. Sen. Relay node placement in large scale wireless sensor networks. *Computer Communications*, 29(4) :490 – 501, 2006.

[56] F. K. Shaikh and S. Zeadally. Energy harvesting in wireless sensor networks : A comprehensive review. *Renewable and Sustainable Energy Reviews*, 55(C) :1041 – 1054, 2016.

[57] M. Welsh, S. Moulton, T. Fulford-Jones, and D. Malan. Codeblue : An ad hoc sensor network infrastructure for emergency medical care. In *MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004)*, 2004.

[58] N.K. Suryadevara, A. Gaddam, R.K. Rayudu, and S.C. Mukhopadhyay. Wireless sensors network based safe home to care elderly people : Behaviour detection. *Sensors and Actuators A : Physical*, 186 :277 – 283, 2012.

[59] R. Jouini, K. Maalaoui, and L. A. Saidane. Designing smart homes for dependent persons assistance. *Journal of Machine to Machine Communications*, 1 :161–176, 2014.

[60] H. Alemdar and C. Ersoy. Wireless sensor networks for healthcare : A survey. *Computer Networks*, 54(15) :2688 – 2710, 2010.

[61] A. Hadjidj, M. Souil, A. Bouabdallah, Y. Challal, and H. Owen. Wireless sensor networks for rehabilitation applications : Challenges and opportunities. *Journal of Network and Computer Applications*, 36(1) :1 – 15, 2013.

[62] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*. ACM, 2002.

[63] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of the ACM*, 47(6) :34–40, 2004.

[64] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking : Design tradeoffs and early experiences with zebranet. *ACM SIGARCH Computer Architecture News*, 30(5) :96–107, 2002.

[65] P. J. Marron, C. Shih, R. Figura, S. Fu, and R. Soleymani. Challenges in the planning, deployment, maintenance and operation of large-scale networked heterogeneous cooperating objects. In *The Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM'11*, 2011.

[66] A Nanda, A. Rath, and S. Rout. Wireless sensor network for coastal erosion : A national perspective. *International Journal of Computer Science & Communication*, 1 :1–5, 01 2010.

[67] K. Khedo, P. Rajiv, and M. Avinash. A wireless sensor network air pollution monitoring system. *CoRR*, 2, 2010.

[68] T. Ojha, S. Misra, and N. S. Raghuwanshi. Wireless sensor networks for agriculture : The state-of-the-art in practice and future challenges. *Computers and Electronics in Agriculture*, 118(C) :66 – 84, 2015.

[69] K. Sha, W. Shi, and O. Watkins. Using wireless sensor networks for fire rescue applications : Requirements and challenges. In *2006 IEEE International Conference on Electro/Information Technology*, 2006.

[70] N. Javaid, S. Faisal, Z. A. Khan, D. Nayab, and M. Zahid. Measuring fatigue of soldiers in wireless body area sensor networks. In *2013 Eighth International Conference on Broadband and Wireless Computing, Communication and Applications*, 2013.

[71] X. Niu, X. Huang, Z. Zhao, Y. Zhang, C. Huang, and L. Cui. The design and evaluation of a wireless sensor network for mine safety monitoring. In *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, 2007.

[72] É. L. Souza, E. F.Nakamura, and R.W. Pazzi. Target tracking for sensor networks : A survey. *ACM Computing Surveys*, 49(2) :30 :1–30 :31, 2016.

[73] V. Kumar, D. Rus, and S. Singh. Robot and sensor networks for first responders. *IEEE Pervasive Computing*, 3(4) :24–33, 2004.

[74] A. A. Kumar S., K. Ovsthus, and L. M. Kristensen. An industrial perspective on wireless sensor networks — a survey of requirements, protocols, and challenges. *IEEE Communications Surveys Tutorials*, 16(3) :1391–1412, 2014.

[75] J. R. Srivastava and T. S. B. Sudarshan. Intelligent traffic management with wireless sensor networks. In *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, 2013.

[76] K. Faez and M. Khanjary. Utospf : A distributed dynamic route guidance system based on wireless sensor networks and open shortest path first protocol. In *2008 IEEE International Symposium on Wireless Communication Systems*, 2008.

[77] Y. Wang, G. Zhou, and T. Li. Design of a wireless sensor network for detecting occupancy of vehicle berth in car park. In *2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)*, 2006.

[78] M. Tubaishat, P. Q. Qi. Zhuang, and Y. Shang. Wireless sensor networks in intelligent transportation systems. *Wireless Communications and Mobile Computing*, 9(3) :287–302, 2009.

[79] L. A. Villas, A. Boukerche, H. S. Ramos, H. A. B. F. de Oliveira, R. B. de Araujo, and A. A. F. Loureiro. Drina : A lightweight and reliable routing approach for in-network aggregation in wireless sensor networks. *IEEE Transactions on Computers*, 62(4) :676–689, 2013.

[80] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi. In-network aggregation techniques for wireless sensor networks : a survey. *IEEE Wireless Communications*, 14(2) :70–87, 2007.

[81] I. Solis and K. Obraczka. In-network aggregation trade-offs for data collection in wireless sensor networks. *International Journal of Sensor Networks*, 1(3/4) :200–212, 2006.

[82] O. Younis and S. Fahmy. An experimental study of routing and data aggregation in sensor networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005.*, 2005.

[83] T. H. Lin and P. Huang. Sensor data aggregation for resource inventory applications. In *IEEE Wireless Communications and Networking Conference, 2005*, volume 4, pages 2369–2374, 2005.

[84] M. Nesa Sudha, M.L. Valarmathi, and A. S. Babu. Energy efficient data transmission in automatic irrigation system using wireless sensor networks. *Computers and Electronics in Agriculture*, 78(2) :215 – 221, 2011.

[85] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. Vigilnet : An integrated sensor network system for energy-efficient surveillance. *ACM Transactions on Sensor Networks*, 2(1) :1–38, 2006.

[86] Y. J. Zhao, R. Govindan, and D. Estrin. Residual energy scan for monitoring sensor networks. In *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002 (Cat. No.02TH8609)*, 2002.

[87] K. Akkaya and I. Ari. *In-Network Data Aggregation in Wireless Sensor Networks*, pages 1131–1146. John Wiley & Sons, Ltd, 2011.

[88] E. F. Nakamura, H. S. Ramos, L. A. Villas, H. A.B.F. de Oliveira, A. L.L. de Aquino, and A. A.F. Loureiro. A reactive role assignment for data routing in event-based wireless sensor networks. *Computer Networks*, 53(12) :1980 – 1996, 2009.

[89] K. Fan, S. Liu, and P. Sinha. Structure-free data aggregation in sensor networks. *IEEE Transactions on Mobile Computing*, 6(8) :929–942, 2007.

[90] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion : A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 56–67, 2000.

[91] J. Y. Choi, J. W. Lee, K. Lee, S. Choi, W. H. Kwon, and H. S. Park. Aggregation time control algorithm for time constrained data delivery in wireless sensor networks. In *2006 IEEE 63rd Vehicular Technology Conference*, pages 563–567, 2006.

[92] H. Li, H. Yu, B. Yang, and A. Liu. Timing control for delay-constrained data aggregation in wireless sensor networks. *International Journal of Communication Systems*, 20(7) :875–887, 2007.

[93] S. C. H. Huang, P. J. Wan, C. T. Vu, Y. Li, and F. Yao. Nearly constant approximation for data aggregation scheduling in wireless sensor networks. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 366–372, 2007.

[94] O. D. Incel and B. Krishnamachari. Enhancing the data collection rate of tree-based aggregation in wireless sensor networks. In *2008 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 569–577, 2008.

[95] Chalermek Intanagonwiwat. *Directed diffusion : an-application-specific and data-centric communication paradigm for wireless sensor networks.* Theses, University Of Southern California, 2002.

[96] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag : A tiny aggregation service for ad-hoc sensor networks. *SIGOPS Operating Systems Review*, 36(SI) :131–146, 2002.

[97] L. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, pages 575–578, 2002.

[98] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8) :1333–1344, 1999.

[99] H. Ö. Tan and I. Körpeoğlu. Power efficient data gathering and aggregation in wireless sensor networks. *ACM SIGMOD Record*, 32(4) :66–71, 2003.

[100] H. O. Tan, I. Korpeoglu, and I. Stojmenovi. Computing localized power-efficient data aggregation trees for sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(3) :489–500, 2011.

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

[102] S. Lindsey, C. Raghavendra, and K. M. Sivalingam. Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9) :924–935, 2002.

[103] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, 1(4) :660–670, 2002.

[104] E. F. Nakamura, H. A. B. F. de Oliveira, L. F. Pontello, and A. A. F. Loureiro. On demand role assignment for event-detection in sensor networks. In *11th IEEE Symposium on Computers and Communications (ISCC'06)*, pages 941–947, 2006.

[105] L. A.Villas, A. Boukerche, R. B. Araujo, and A. A. F. Loureiro. A reliable and data aggregation aware routing protocol for wireless sensor networks. In *Proceedings of the 12th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 245–252, 2009.

[106] L. A. Villas, A. Boukerche, H. A.B.F. de Oliveira, R. B. de Araujo, and A. A.F. Loureiro. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Networks*, 12 :69 – 85, 2014.

[107] O. A. Mahdi, A. W. A. Wahab, M. Y. I. Idris, A. M. A. Abu znaid, Y. R. B. Al-Mayouf, and S. Khan. Wdars : A weighted data aggregation routing strategy with minimum link cost in event-driven wsns. *Journal of Sensors*, 2016 :3428730 :1–3428730 :12, 2016.

[108] C.-M. Chao and T. Y. Hsiao. Design of structure-free and energy-balanced data aggregation in wireless sensor networks. *Journal of Network and Computer Applications*, 37 :229 – 239, 2014.

[109] J. Zhang, Q. Wu, F. Ren, T. He, and C. Lin. Effective data aggregation supported by dynamic routing in wireless sensor networks. In *2010 IEEE International Conference on Communications*, pages 1–6, 2010.

[110] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicast routing for multimedia communication. *IEEE/ACM Transactions on Networking*, 1(3) :286–292, 1993.

[111] J. A. Stankovic T. He, T. F. Abdelzaher, and C. Lu. A spatiotemporal communication protocol for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 16(10) :995–1006, 2005.

[112] E. Felemban, Chang-Gun Lee, and E. Ekici. Mmspeed : multipath multi-speed protocol for qos guarantee of reliability and. timeliness in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 5(6) :738–754, 2006.

[113] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher. Real-time power-aware routing in sensor networks. In *200614th IEEE International Workshop on Quality of Service*, pages 83–92, 2006.

[114] K. Liu, N. Abu-Ghazaleh, and K. . Kang. Jits : just-in-time scheduling for real-time sensor data dissemination. In *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM'06)*, pages 5 pp.–46, 2006.

[115] H. Yousefi, M. H. Yeganeh, N. Alinaghipour, and A. Movaghar. Structure-free real-time data aggregation in wireless sensor networks. *Computer Communications*, 35(9) :1132 – 1140, 2012.

[116] T. Melodia, D. Pompili, V. C. Gungor, and I. F. Akyildiz. Communication and coordination in wireless sensor and actor networks. *IEEE Transactions on Mobile Computing*, 6(10) :1116–1129, 2007.

[117] M. F. Munir and F. Filali. A novel self organizing framework for sanets. In *12th European Wireless Conference 2006 - Enabling Technologies for Wireless Multimedia Communications*, pages 1–7, 2006.

[118] M. F. Munir and F. Filali. Analyzing the performance of a self organizing framework for wireless sensor-actuator networks. In *CNS 2007, 10th ACM/SIGSIM Communications and Networking Simulation Symposium, March 25-29, 2007, Norfolk, USA*, 2007.

[119] G. A. Shah, M. Bozyigit, and F. B. Hussain. Cluster-based coordination and routing framework for wireless sensor and actor networks. *Wireless Communications and Mobile Computing*, 11(8) :1140–1154, 2011.

[120] G. Di Caro, F. Ducatelle, and L. M. Gambardella. *Theory and Practice of Ant-Based Routing in Dynamic Telecommunication Networks*, pages 185–216. 01 2008.

[121] G. Michel and M. Michel. *Graphes et algorithmes (4e ed.)*. Lavoisier, 2009.

[122] Falko Dressler. A study of self-organization mechanisms in ad hoc and sensor networks. *Computer Communications*, 31(13) :3018 – 3029, 2008.

[123] P. Siarry J. Dreo, A. Petrowski and E. Taillard. *Métaheuristiques pour l'optimisation difficile*. EYROLLES, 2003.

[124] M. Weiser. The computer for the 21st century. *ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3) :3–11, 1999.

[125] B. Atakan and O. B. Akan. Immune system based distributed node and rate selection in wireless sensor networks. In *2006 1st Bio-Inspired Models of Network, Information and Computing Systems*, pages 1–8, 2006.

[126] J. Y. Le Boudec and S. Sarafijanović. An artificial immune system approach to misbehavior detection in mobile ad hoc networks. In *Biologically Inspired Approaches to Advanced Information Technology*, pages 396–411. Springer Berlin Heidelberg, 2004.

[127] W. Vogels, R. van Renesse, and K. Birman. The power of epidemics : Robust communication for large-scale distributed systems. *ACM SIGCOMM Computer Communication Review*, 33(1) :131–135, 2003.

[128] T. Tsuchiya and T. Kikuno. An adaptive mechanism for epidemic communication. In *Biologically Inspired Approaches to Advanced Information Technology*, pages 306–316. Springer Berlin Heidelberg, 2004.

[129] A. Tyrrell and G. Auer. Imposing a reference timing onto firefly synchronization in wireless networks. In *2007 IEEE 65th Vehicular Technology Conference - VTC2007-Spring*, pages 222–226, 2007.

[130] N. Wakamiya and M. Murata. Synchronization-based data gathering scheme for sensor networks. *IEICE TRANSACTIONS on Communications*, E88-B(3) :873–881, 2005.

[131] F. Dressler and O. B. Akan. A survey on bio-inspired networking. *Computer Networks*, 54(6) :881 – 900, 2010.

[132] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system : optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1) :29–41, 1996.

[133] M. Adamu, K. P. Seng, L. M. Ang, and W. C. Chia. Energy efficiency performance improvements for ant-based routing algorithm in wireless sensor networks. *Journal of Sensors*, 2013 :759654 :1–759654 :17, 02 2013.

[134] T. Stützle and H. H. Hoos. Max–min ant system. *Future Generation Computer Systems*, 16(8) :889 – 914, 2000.

[135] T. Stutzle and H. Hoos. Max-min ant system and local search for the traveling salesman problem. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, pages 309–314, 1997.

[136] M. Dorigo and L. M. Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2) :73 – 81, 1997.

[137] M. Dorigo and L. M. Gambardella. Ant colony system : a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1) :53–66, 1997.

[138] M. Dorigo and T Stützle. *Ant Colony Optimization*. Bradford Company, 2004.

[139] M. Saleem, G. A. Di Caro, and M. Farooq. Swarm intelligence based routing protocol for wireless sensor networks : Survey and future directions. *Information Sciences*, 181(20) :4597 – 4624, 2011.

[140] T. Camilo, C. Carreto, J. Silva, and F. Boavida. An energy-efficient ant-based routing algorithm for wireless sensor networks. In *Ant Colony Optimization and Swarm Intelligence*, pages 49–59. Springer Berlin Heidelberg, 2006.

[141] Y. Sun, H. Ma, L. Liu, and Y. Zheng. Asar : An ant-based service-aware routing algorithm for multimedia sensor networks. *Frontiers of Electrical and Electronic Engineering in China*, 3(1) :25–33, 01 2008.

[142] D. Niannian, P. X. Liu, and C. Hu. Data gathering communication in wireless sensor networks using ant colony optimization. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 697–702, 2005.

[143] W. Cai, Y. Zhang X. Jin, K. Chen, and R.Wang. Aco based qos routing algorithm for wireless sensor networks. In *Ubiquitous Intelligence and Computing*, pages 419–428. Springer Berlin Heidelberg, 2006.

[144] Y. Zhang, L. D. Kuhn, and M. P. J. Fromherz. Improvements on ant routing for sensor networks. In *Ant Colony Optimization and Swarm Intelligence*, pages 154–165. Springer Berlin Heidelberg, 2004.

[145] W. M. Chen, C. S. Li, F. Y. Chiang, and H. C. Chao. Jumping ant routing algorithm for sensor networks. *Computer Communications*, 30(14) :2892 – 2903, 2007.

[146] D. Li, W. Liu, and L. Cui. Easidesign : An improved ant colony algorithm for sensor deployment in real sensor network system. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5, 2010.

[147] X. Liu and D. He. Ant colony optimization with greedy migration mechanism for node deployment in wireless sensor networks. *Journal of Network and Computer Applications*, 39 :310 – 318, 2014.

[148] X. Liu. A novel transmission range adjustment strategy for energy hole avoiding in wireless sensor networks. *Journal of Network and Computer Applications*, 67(C) :43 – 52, 2016.

[149] M. Liu and C. Song. Ant-based transmission range assignment scheme for energy hole problem in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2012, 2012.

[150] A. M. Zungeru, L. M. Ang, and K. P. Seng. Classical and swarm intelligence based routing protocols for wireless sensor networks : A survey and comparison. *Journal of Network and Computer Applications*, 35(5) :1508 – 1536, 2012.

[151] R. Schoonderwoerd, J. L. Bruten, O. E. Holland, and L. J. M. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2) :169–207, 1996.

[152] E. Sigel, B. Denby, and S. Le Hégarat-Mascle. Application of ant colony optimization to adaptive routing in aleo telecomunications satellite network. *Annales Des Télécommunications*, 57(5) :520–539, May 2002.

[153] G. Di Caro and M. Dorigo. Antnet : Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9(1) :317–365, 1998.

[154] G. Di Caro, F. Ducatelle, and L. M. Gambardella. Anthocnet : an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5) :443–455, 2005.

[155] G. Singh, S. Das, S. V. Gosavi, and S. Pujar. *Recent Developments in Biologically Inspired Computing*, chapter Ant Colony Algorithms for Steiner Trees : An Application to Routing in Sensor Networks, pages 181–206. IGI Global, 2005.

[156] R. Misra and C. Mandal. Ant-aggregation : ant colony algorithm for optimal data aggregation in wireless sensor networks. In *2006 IFIP International Conference on Wireless and Optical Communications Networks*, pages 5 pp.–5, 2006.

[157] L. Villas, A. Boukerche, R. B. de Araujo, and A. A. F. Loureiro. Highly dynamic routing protocol for data aggregation in sensor networks. In *The IEEE symposium on Computers and Communications*, 2010.

[158] I. Korpeoglu H. Bagci and A. Yazici. A distributed fault-tolerant topology control algorithm for heterogeneous wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(4) :914–923, April 2015.

[159] A. Sobeih, J. C. Hou, L. C. Kung, N. Li, H. Zhang, W. P. Chen, H. Y. Tyan, and H. Lim. J-sim : a simulation and emulation environment for wireless sensor networks. *IEEE Wireless Communications*, 13(4) :104–119, 2006.