N° d'ordre : 03/2020-D/INF

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE
USTHB / ALGER
FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE

Thèse de doctorat en sciences

En vue de l'obtention de grade de Docteur en

INFORMATIQUE

Spécialité : Informatique Mobile

Par : IABBASSEN Dalila

SUJET

Dissémination des données à base d'agents mobiles dans les réseaux de capteurs sans fil

Soutenue publiquement, le 07 mars 2020, devant le jury composé de :

M.	Y.AKLOUF	Professeur à l'USTHB/FEI	Président
Mme	S.MOUSSAOUI	Professeur à l'USTHB/FEI	Directrice de thèse
M.	T.AHMED	Professeur à l'Université de Bordeaux	Examinateur
M.	A.BACHIR	Professeur à l'Université de Biskra	Examinateur
Mme	N.NOUALI	Directrice de recherche au CERIST	Examinatrice
M.	M.GUERROUMI	Maitre de conférences/A à l'USTHB/FEI	Examinateur

Résumé

En raison des diverses propriétés des réseaux de capteurs sans fil -RCSF- (contraintes physiques, topologie de déploiement, paradigmes de communication…), il est difficile de concevoir une solution qui répond à toutes les exigences de ce type de réseaux. Pour cela, plusieurs protocoles ont été proposés pour une dissémination efficace des données dans les RCSFs. Ces protocoles sont divisés en deux grandes catégories : ceux basés sur le paradigme classique client/serveur et ceux utilisant des agents mobiles (AM).

Dans les architectures traditionnelles client/serveur, les données de différentes sources sont transférées à une destination; alors que dans le paradigme agents mobiles, un code exécutable spécifique à une tache traverse les sources pertinentes pour collecter les données. Les agents mobiles peuvent être utilisés pour réduire considérablement le coût de communication, spécialement à travers de faibles bandes passantes, en déplaçant la fonction de traitement vers la donnée plutôt qu'apporter la donnée vers un processeur central.

Dans le cadre de ces travaux de thèse, nous avons classifié les différentes solutions de dissémination des données à base d'AMs proposées pour les RCSFs en deux catégories :

- Les protocoles de dissémination des données basés sur un agent mobile : Cette catégorie a pour but de combler les inconvénients du modèle centralisé. C'est-à-dire améliorer la consommation énergétique en déplaçant le code de traitement vers les nœuds capteurs plutôt que d'apporter toutes les données vers le puits. Cependant, cette approche s'est avérée couteuse en temps de réponse puisque un seul agent mobile devrait parcourir tous les nœuds sources du réseau séquentiellement.
- Les protocoles de dissémination des données basés sur de multiples agents mobiles : cette catégorie emploie plusieurs agents mobiles opérants simultanément pour effectuer la tâche d'agrégation et fusion de données, qui mène à un gain considérable en temps tout en diminuant la consommation énergétique.

La collecte des données dans les applications query-driven est initiée par des interrogations envoyées par le puits. Pour cette raison, la plupart des protocoles de dissémination des données à base de multiples agents mobiles utilisent des itinéraires statiques allant du nœud puits et retournant vers lui. En outre, chaque AM devrait transporter le code de traitement afin de collecter les données sensorielles et les ramener vers le puits. Cependant, cette hypothèse génère beaucoup de trafic pour les AMs; notamment entre le puits et la région cible; résultant ainsi une consommation supplémentaire en énergie et latence. De plus, les itinéraires statiques des AMs ne sont pas tolérants aux pannes.

Pour pallier ces problèmes, nous proposons une solution coopérative et hybride à base de multiples agents mobiles avec des itinéraires dynamiques, nommée CHEESE (Cooperative and hybrid energy efficient scheme for wireless sensor networks). CHEESE utilise une architecture basée sur des secteurs centrés équiangles, et s'appuie sur l'utilisation d'un puits virtuel qui est un nœud situé au centre de la région cible et qui représente le sommet de tous les secteurs. Le nœud puits envoie donc un seul AM principal vers la région cible qui sera

hébergé dans le puits virtuel. L'AM principal va générer et dispatcher de multiples AMs dans la région cible afin de collecter et agréger les données en parallèle. Dans les approches existantes à base d'AM, ce dernier passe à travers tous les nœuds sources qui lui sont affectés pour rassembler toutes les données. Ceci produit un itinéraire long qui consomme plus d'énergie et de temps de parcours. Cependant, dans CHEESE, nous optons pour un itinéraire optimisé qui fait transiter l'AM par la moitié des nœuds sources seulement. Les autres nœuds sources non visités par l'AM envoient leurs données aux nœuds sources voisins appartenant au chemin de l'AM. Ceci accélère la vitesse de parcours de chaque AM et diminue le risque de perte des données.

A la fin de l'opération de tous les AMs, et contrairement aux travaux connexes qui expédient chaque AM en retour vers le puits individuellement, le puits virtuel dans CHEESE rassemble toutes les données des différents AMs. Il les agrège et les fusionne pour enfin n'envoyer que l'information pertinente au puits. Le rôle du puits virtuel est d'éliminer le trafic inutile entre chaque AM et le puits. Il réduit davantage la quantité des données transmises au puits en rassemblant toutes les données de tous les AMs dans un seul paquet. Effectivement, les résultats de simulation de notre protocole ont confirmé son efficacité dans plusieurs scénarios par rapport aux travaux similaires existants.

Abstract

Due to the various properties of wireless sensor networks -WSN- (physical constraints, deployment topology, communication paradigms, etc.), it is not easy to design a solution that meets all the requirements of this type of network. For this reason, several protocols have been proposed for efficient data dissemination in WSNs. These protocols are divided into two main categories: those based on the classical client / server paradigm, and those using mobile agents (MA).

In the client/server traditional architectures, the data of various sources are transferred to a destination, whereas in mobile agent (MA) paradigms, an achievable code specific to a task crosses the relevant sources to assemble the data. Mobile agents can be used to reduce considerably the cost of communication, especially through weak band-widths, by moving the processing function towards the data rather than to bring the data towards a main processor.

In this thesis, we classified various solutions of MAs based data dissemination which were proposed for wireless sensor networks (WSNs) into two categories:

- Single Itinerary Planning (SIP): The purpose of this category is to fill the disadvantages of the centralized model, to improve the power consumption by moving the processing code towards the sensor nodes rather than bringing all the data in entirety towards the sink. However; this solution proved to be expensive in response times since only one mobile agent should traverse all the source nodes of the network sequentially.
- Multiple Itinerary Planning (MIP): This category employs several simultaneous mobile agents in order to gain in response times while decreasing the power consumption.

Data collection in query-driven applications is initiated by the sink queries. This is why the majority of MIP data dissemination protocols use static routes starting from the sink node and turning back towards him. Moreover, each MA should carry the processing code in order to collect the sensory data and bring them back towards the sink. However, this assumption generates much traffic for MAs; especially between the sink and the target region; thus resulting on an additional energy consumption and latency. In addition, the MAs static itineraries are not fault tolerant.

To mitigate these problems, we propose a cooperative and hybrid solution using multiple mobile agents named CHEESE (Cooperative and hybrid energy efficient scheme for wireless sensor networks). CHEESE uses architecture based on centered squared sectors, and relies on the use of a virtual sink which is a node located at the center of the target area. The sink node thus sends only one main MA towards the target area which will be hosted by the virtual sink. The main MA will generates and dispatch multiple MAs in the target area in order to collect and aggregate data simultaneously. In the existing MAs approaches, the MA transits through all source nodes to gather all the data. This produces a long itinerary which will consume more energy and time. However, in CHEESE, we use a reduced MA itinerary which will browse only half of the source nodes. The other source nodes not visited by the MA send their data to their neighbours source nodes belonging to the MA itinerary. This will accelerate the MA speed and decreases the risk of data loss.

At the end of all MAs operations; and differing to related works which individually send each MA in return towards the sink; the virtual sink in CHEESE gathers all the MAs data; it

aggregates and fuses them to finally send only pertinent data to the sink. The virtual sink goal is to eliminate the useless traffic between each MA and the sink. It reduces more the quantity of the transmitted data to the sink by gathering all the data of all MAs in only one package. Effectively, the simulation results of our protocol confirmed its effectiveness in several scenarios compared with other similar works.

<u>À mes deux sources de lumière :</u>

mon rayon de soleil : Samy

et mon clair de lune : Sarah

Remerciements

Je remercie avant tout ALLAH le tout puissant, de m‘avoir donnée la santé, le courage et la patience pour mener à bien ce projet de doctorat.

Je remercie énormément Pr. Samira MOUSSAOUI, ma directrice de thèse, pour m’avoir donnée la possibilité de travailler sur ce sujet de recherche très intéressant. Je la remercie pour son aide précieuse et ses conseils, qui m’ont permis de bien comprendre les différents aspects de ce sujet et de bien analyser les problèmes posés.

Je témoigne énormément ma gratitude envers tous les membres du jury et je remercie vivement M. Y/AKLOUF d‘avoir présidé mon jury de thèse. Je remercie également M. T/AHMED, M. A/BACHIR, Mme N/NOUALI ainsi que M. M/GUERROUMI d‘avoir accepté d’évaluer mon travail.

Je remercie aussi énormément ma famille, spécialement mes très chers parents, mes sœurs et frères ainsi que mon mari, qui m’ont soutenu vivement durant toutes mes études. Leur amour et encouragements m’ont été extrêmement bénéfiques durant toutes ces années.

Enfin, je tiens à exprimer mes sincères gratitudes aux personnes qui ont contribuées de près ou de loin à l’élaboration de la présente thèse de doctorat.

Sommaire

SOMMAIRE

Chapitre III. Protocoles de disséminations des données à multiples AMs dans les RCSFs

PARTIE 2 : Contribution

Chapitre IV. CHEESE : Un protocole coopératif et hybride

Chapitre VI. Evaluation des performances

LISTE DES FIGURES

LISTE DES tableaux

LISTE DES ACRONYMES

AbDD	Agent based Directed Diffusion
ADC	convertisseur analogique-numérique
AG-MIP	Angle Gap Multi agents Itinerary Planning
AM	Agent Mobile
BST-MIP	balanced spanning tree for multi-agent itinerary planning
CBIB	Clone-Based Design
CDDAM	clone-based dynamic and distributed agent migration
CH	Cluster Head
CHEESE	Cooperative and hybrid energy efficient schemE for wireless sensor networks
CL-MIP	Centre Location-based Multi agents Itinerary Planning
CMIP	clone mobile-agent itinerary planning approach
DD	Directed Diffusion
EBMADD	Energy Balanced Mobile Agent based Data Dissemination
GA-MIP	Genetic Algorithm-based Multi agents Itinerary Planning
GCF	Global Closest First
GIGM-MIP	a greatest information in the greater memory-based Multi agents Itinerary Planning
IEMA	Itinerary Energy Minimum Algorithm
IEMADI	Information Extraction protocol based on Multiple software Agents with Dynamic Itineraries
IEMF	Itinerary Energy Minimum for First-source-selection
ILS	iterated local search
LCF	Local Closest First
MADD	Mobile Agent based Directed Diffusion
MAEF	Multi-mobile Agent itinerary planning-based Energy and Fault aware
MANET	Mobile Ad hoc Networks
MAWSN	Mobile Agent based Wireless Sensor Network
MIP	Multiple Itinerary Planning
MMADIDD	Multiple mobile agents with dynamic itinerary based data dissemination
NOID	Near-Optimal Itinerary Design
OMIP	Optimal Multi-Agents Itinerary Planning
RCSF	Réseau de capteurs sans fil
SIP	Single Itinerary Planning
SMIP	a spawn multi-mobile agent itinerary planning
SNOID	second near-optimal itinerary design
TBID	Tree-Based Design Directions
TSG	Totally connected graph
VCL	Visiting Central Location
WSN	Wireless Sensor Network

Introduction générale

Introduction générale

La technologie des Réseaux de Capteurs Sans Fil (RCSF) joue un rôle important dans le domaine de la télécommunication. Elle permet d'apercevoir une nouvelle façon de recueillir les données. La grande puissance de ce type de réseaux se repère dans la facilité de leur déploiement vu leur petite taille et leur coût réduit. Cependant, les RCSFs sont principalement limités en termes de ressources (capacités insuffisante de stockage, de traitement et d'autonomie) parce qu'ils sont généralement alimenté par des piles rarement remplaçables. Ainsi, le principal problème des RCSFs est de savoir comment fournir les économies d'énergie pour prolonger la durée de vie du réseau. Parmi tous les aspects de la consommation d'énergie, la communication radio est considérée comme le principal consommateur d'énergie.

La dissémination des données est considérée comme une des phases principales de consommation d'énergie dans la communication des RCSFs [3]. D'où l'élimination du trafic des données redondantes et la réduction des coûts de communication sont les principaux défis dans la dissémination des données. Il est connu que l'inefficacité de la dissémination (comme les diffusions aveugles) cause des congestions et bloquent toutes les communications dans le réseau. Par conséquent, l'objectif d'une dissémination efficace des données est d'envoyer n'importe quels types d'informations (données ou requêtes) aux nœuds concernés, tout en minimisant le nombre de nœuds de transmission et le coût énergétique [1][2]. Concernant les applications des RCSFs telles que la surveillance de l'environnement, la dissémination efficace des données y joue un rôle très important pour prolonger la durée de vie du réseau.

La technologie des Agents Mobiles "AMs" représente une tendance dans l'informatique distribuée, qui répond aux problèmes de flexibilité et d'évolutivité des modèles centralisés. Un AM est un type particulier de programme autonome qui migre entre les nœuds d'un réseau pour effectuer une ou plusieurs tâches. Il répond aux conditions changeantes de l'environnement du réseau, afin de réaliser les objectifs de son répartiteur [5]. Les AMs sont donc bien adaptés pour réduire considérablement le coût de communication, spécialement à travers de faibles bandes passantes, en déplaçant la fonction de traitement vers la donnée plutôt qu'apporter la donnée vers un processeur central.

Dans ces travaux de thèse, nous nous intéressons à l'utilisation des AMs pour déployer dynamiquement de nouvelles applications dans les réseaux de capteurs. En effet, les AMs se révèlent être une méthode efficace pour relever les problèmes liés à la consommation d'énergie. Ils ont aussi été jugés particulièrement utiles pour faciliter la fusion et la dissémination des données dans les RCSFs [6].

Nous distinguons deux modèles d'algorithmes à base d'AMs dans les RCSFs : les solutions basées SIP (Single Itinerary Planning) employant un seul AM pour collecter toutes les données séquentiellement, et celles basées MIP (Multiple Itinerary Planning) employant plusieurs AMs simultanés qui opèrent en parallèle.

Le premier modèle à un seul AM a pour but de combler les inconvénients du mode client/serveur (C/S). Ce qui dit améliorer la consommation énergétique en déplaçant le code

de traitement vers les nœuds capteurs plutôt qu'apporter toutes les données en entier vers le puits. Cependant, cette solution s'est avérée coûteuse en temps de réponse notamment dans les réseaux à grande échelle, puisque un seul AM devrait parcourir tout le réseau séquentiellement pour enfin envoyer les données collectées au puits [74].

Le deuxième modèle disperse plusieurs AMs en parallèle afin de faire face au problème de l'échelle et de gagner en temps de réponse tout en diminuant la consommation d'énergie. Bien que les MIP ont surmonté les faiblesses des algorithmes SIP, leur conception est compliquée en raison de plusieurs multiples défis qui entrent en considération [7], comme :

- la détermination du nombre optimal d'agents mobiles,
- la conception de l'itinéraire optimal de chaque agent mobile,
- la division des nœuds sources dans des sous-ensembles de groupes afin d'assigner chaque agent mobile à un groupe spécifique.

La majorité des protocoles de dissémination des données à base de multiples AMs [33][47][49][73] utilisent des itinéraires allant du nœud puits et retournant vers lui. En outre, chaque AM devrait transporter le code de traitement afin de collecter les données sensorielles et les apporter vers le puits. Cependant, cette hypothèse augmente le nombre de sauts pour chaque AM, résultant ainsi une consommation supplémentaire en énergie. Pour pallier ce problème, Nous avons proposé une solution hybride nommée CHEESE (Cooperative and hybrid energy efficient scheme for wireless sensor networks) [55] à base de multiples AMs. CHEESE s'appuie sur les opérations d'agrégation effectuées par plusieurs AMs parallèles. Il utilise aussi un puits virtuel dans la région cible dans le but d'alléger la consommation d'énergie et la latence utilisées pendant le processus de collecte des données via multiples AMs. Pour cela, le nœud puits envoie un seul AM principal vers la région cible du réseau. Ensuite, cet AM sera hébergé chez le puits virtuel qui va générer et dispatcher de multiples AMs pour collecter et agréger les données en parallèle dans la zone cible. A la fin de l'opération de tous les AMs, le puits virtuel rassemble toutes les données, il les agrège et les fusionne pour enfin n'envoyer que l'information pertinente au puits.

Contrairement aux solutions existantes, qui font transiter l'AM par tous les nœuds sources pour collecter les données, notre solution utilise un itinéraire réduit permettant à l'AM de visiter seulement la moitié des nœuds sources.

Aussi, nous permettons un maximum de niveaux d'agrégations pour minimiser la taille des données utiles et gagner davantage en énergie et en temps de transfert. En effet, nous proposons un niveau d'agrégation supplémentaire qui élimine les redondances temporelles entres les données cycliques collectées par les AMs.

Afin de tester l'efficacité de notre contribution, nous avons effectué des simulations et des comparaisons avec les 3 modèles existants : C/S, SIP et MIP. En effet, notre proposition donne des résultats intéressants dans plusieurs scénarios.

Cette thèse est organisée en deux grandes parties : la première partie comporte l'état de l'art, et la deuxième contient notre contribution. La première partie comprend trois chapitres.

Le premier chapitre nous fait découvrir les RCSFs et leur intérêt ainsi que la notion de dissémination des données dans ce type de réseaux. Le second chapitre est consacré aux notions générales sur les AMs et leur utilisation adaptée à la collecte des données dans les RCSFs illustré avec quelques protocoles SIP. Le troisième chapitre s'étale sur plusieurs protocoles proposés dans la littérature employant de multiples AMs pour la collecte des données dans les RCSFs. Ils ont été classifiés en plusieurs catégories. La deuxième partie comportant notre contribution, qui est détaillée dans le quatrième chapitre et illustrée avec des schémas explicatifs et des organigrammes des différentes étapes du protocole. Le cinquième et dernier chapitre comporte les détails de simulations et les résultats des différents scénarios effectués pour comparer les performances de notre protocole avec les travaux connexes. Enfin, cette thèse sera conclue par des perspectives et des pistes potentielles pour de futures recherches.

Partie 1

État de l'art

Chapitre I

La dissémination des données dans les RCSFs

I.1. Introduction

Un capteur est un petit appareil autonome qui peut détecter et collecter des événements environnementaux, tels que la température, le son et le mouvement d'objets. Le capteur a la capacité de communiquer sans fil (radio) avec d'autres capteurs à proximité, ainsi que des modules de traitement (processeur) et de mémoire. Par conséquent, il est possible de former un réseau de capteurs sans fil coopérant (RCSF) sur une portée considérable. Cette technologie est considérée comme la fusion de deux technologies informatiques principales: les systèmes embarqués et les réseaux de communication sans fil [10].

Au cours des dernières années, l'émergence des RCSFs a inspiré de nombreux sujets de recherche. Beaucoup de travaux ont été consacrés à l'organisation, la communication, la gestion de l'alimentation et le routage des données dans ce type de réseaux. Dans ce chapitre, nous nous concentrerons sur le sujet de la dissémination des données dans les RCSFs qui a connu une grande évolution dans le temps.

I.2. Généralités sur les RCSFs

I.2.1. Définition d'un réseau de capteurs sans fil

Comme son nom l'indique, un réseau de capteurs sans fil (RCSF) est un réseau dont les nœuds sont des capteurs (ou des actionneurs) [70], soit de minuscules appareils possédant des ressources très limitées, mais qui leur permettent d'acquérir des données de leur environnement immédiat.

Un capteur est un dispositif électronique qui convertit un phénomène physique en un signal électrique. Il représente une partie de l'interface entre le monde physique et les appareils électriques capables d'interpréter ces signaux, tels que des ordinateurs. Un capteur possède également le matériel nécessaire pour effectuer des communications sans fil [69].

Contrairement aux réseaux traditionnels qui sont généralement conçus pour transférer des données entre les nœuds, un réseau de capteurs est étendu pour réaliser différentes fonctions. Plusieurs nœuds dans un réseau de capteurs combinent les données capturées localement, et les communiquent à l'utilisateur. L'utilisateur ne s'intéresse pas à l'envoie d'une requête à un nœud particulier, mais plutôt, à la récupération des informations à partir d'une région ou d'un ensemble de nœuds [67].

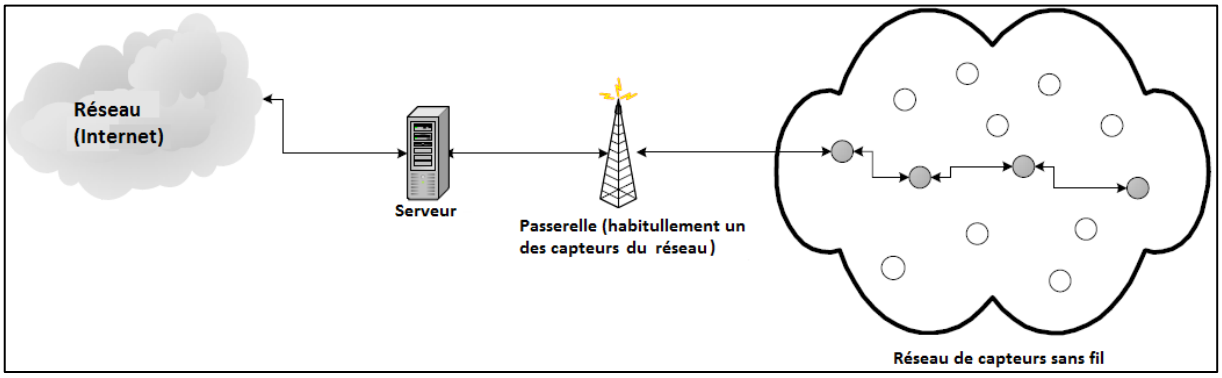


Figure I.1: Exemple d'un réseau de capteurs sans fil.

Ce type de réseau n'a pas d'adresse IP et est conçu pour se fondre dans l'environnement et travailler avec les réseaux traditionnels. Le nombre de nœuds qui composent un réseau de capteurs varie entre quelques dizaines de capteurs et plusieurs millions, selon les besoins de l'application. La figure I.1 montre un exemple de déploiement d'un réseau de capteurs sans fil (WSN-Wireless Sensor Network).

I.2.2. Composition du réseau de capteurs sans fil

Le déploiement d'un réseau de capteurs sans fil requiert 3 types d'éléments (voir figure I.1):

I.2.2.1. Le capteur : C'est l'élément de base pour l'acquisition des données. Il est caractérisé par sa petite taille et son faible coût mais avec des ressources matérielles très limitées (énergie, bande passante, mémoire et puissance de calcul).

Les différentes parties d'un nœud capteur peuvent être classées en six principales unités (voir figure 1.2) : 1.L'unité de transmission, 2. L'unité de traitement, 3. L'unité d'acquisition, 4. Les convertisseurs du signal Analogique/Numérique, 5. L'unité de contrôle d'énergie (batterie), 6. L'unité de stockage temporaire (mémoire).

Parfois, Un nœud capteur peut contenir d'autres unités dépendantes de l'application du réseau. En effet, la plupart des opérations de captage et algorithmes de routage dans les réseaux de capteurs sans fil requièrent la connaissance de la localisation géographique des nœuds, car ces nœuds sont souvent déployés aléatoirement et fonctionnent d'une manière autonome, ceci rend l'intégration d'une unité, consacrée au système de localisation, très utile dans un nœud capteur.

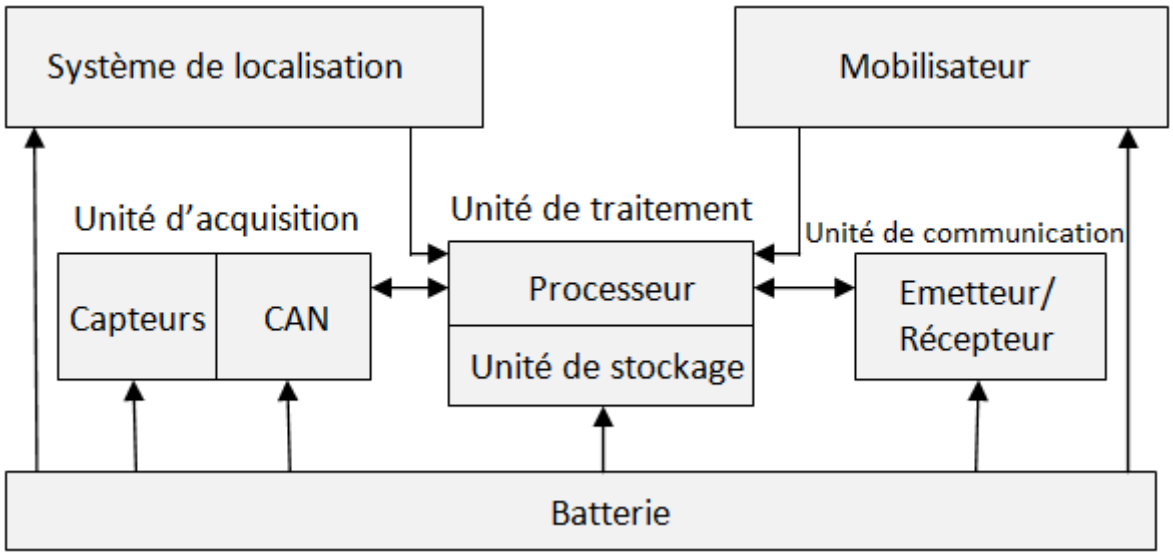


Figure I.2: Anatomie d'un nœud capteur [66].

D'où, il est souvent supposé que ces nœuds possèdent un système de localisation GPS avec une précision au moins égale à 5m [8]. La conception des nœuds capteurs peut aller jusqu'à prévoir un système de mobilisation du capteur pour le déplacer en cas de nécessité.

Toutes ces unités peuvent exiger leur intégration dans un boîtier de taille minimale inférieure à un centimètre cube, et avec un poids très léger qui permet aux nœuds de rester suspendus dans l'air, si l'application l'exige [10].

I.2.2.2. Le collecteur (puits ou sink): Les collecteurs sont un autre type de nœuds du réseau qui permettent de rassembler les données provenant de plusieurs capteurs et qui possèdent également des fonctions de passerelle, reliant ainsi les capteurs avec un réseau externe (Internet par exemple). Ces derniers possèdent beaucoup plus de capacités que les nœuds capteurs tant au niveau de la mémoire, de la vitesse de traitement et des réserves en énergie [9].

Ce nœud est responsable, en plus de la collecte des rapports, de la diffusion des demandes sur les types de données requise aux capteurs via des messages de requêtes. Un réseau de capteurs peut contenir plusieurs nœuds puits diffusant des intérêts différents. Par exemple, un nœud puits peut demander à tous les capteurs se trouvant dans la région sud du champ de captage d'envoyer un rapport de température chaque x minutes, pendant qu'un autre peut être intéressé seulement par les hautes températures (> 50° C) dans la région nord. Par conséquent, un capteur doit pouvoir stocker toutes les requêtes reçues, et les traiter séparément [10].

I.2.2.3. La station de base : C'est l'élément recueillant les données et/ou les analyses du réseau et servant également à son administration. Ça peut être un PC classique, portable, Palm, etc. Ainsi les données détectées vont parcourir le réseau de capteur en capteur jusqu'au nœud puits ensuite vers la station de base. En retour, celle-ci peut demander l'exécution d'une tâche particulière à un capteur donné [9].

I.2.3. Domaines d'applications des réseaux de capteurs sans fil

La diminution des coûts matériels, ainsi que l'élargissement de la gamme des capteurs disponibles, a permis d'étendre le champ d'application des réseaux de capteurs sans fil. Le domaine militaire a été un moteur initial dans le développement de ces technologies pour l'analyse de terrains dangereux ou la surveillance de mouvements. En effet, les armées souhaitent être en mesure d'espionner discrètement leurs ennemis. L'absence de câbles entre les nœuds, leur petite taille, le nombre élevé de nœuds déployables – pour couvrir une zone étendue – répondent bien à ses critères.

Puis, la diminution des coûts de fabrication des capteurs ainsi que la réduction de leur taille a entraîné son utilisation dans plusieurs applications civiles. Ils peuvent par exemple être utilisés à des fins de surveillance environnementale, pour la détection de feux de forêts, la surveillance d'activité volcanique ou sismique, ou encore le suivi du déplacement d'animaux. Par exemple, des capteurs peuvent être placés dans des régions glaciaires ou tropicales afin de suivre de manière précise les effets du réchauffement de la planète, les changements climatiques ou l'augmentation de la pollution. Dans les champs agricoles, les capteurs peuvent être semés avec les graines. Ainsi, les zones sèches seront facilement identifiées et l'irrigation sera donc plus efficace.

Sur les sites industriels, les centrales nucléaires ou dans les pétroliers, des capteurs peuvent être déployés pour détecter des fuites de produits toxiques (gaz, produits chimiques,

éléments radioactifs, pétrole, etc.) et alerter les utilisateurs dans un délai suffisamment court pour permettre une intervention efficace.

Les RCSFs peuvent également être employés pour la surveillance de l'habitat. Par exemple en montagne, des capteurs pourraient permettre de recenser les animaux fréquentant un territoire donné. Nous citons comme exemple, un biologiste voulant savoir l'existence d'un certain animal, et lorsque l'animal est détecté, il doit être suivi d'aussi près que possible. Dans ce cas, le réseau de capteurs est utilisé pour la reconnaissance automatique de cible et son suivi.

Pareillement, les réseaux de capteurs déployés dans les moyens de transport peuvent donner un mode de transport intelligent puisque les informations sur le trafic en temps réel sont collectées par les réseaux de capteurs pour améliorer plus tard des modèles de transport et signaler une alerte de la congestion et des problèmes de circulation aux conducteurs.

D'autre part, dans le domaine de la médecine, les réseaux de capteurs soutiennent des interfaces pour les personnes handicapées, la surveillance intégrée des patients, les diagnostics et l'administration de médicaments dans les hôpitaux, la télésurveillance des données physiologiques des malades ainsi que le suivi et la surveillance des médecins ou des patients à l'intérieur d'un hôpital. En effet, cela peut servir à la surveillance du niveau de glucose, le monitoring des organes vitaux ou la détection de cancers.

Aujourd'hui, plus d'applications spécifiques dans différents domaines se posent, et au lieu de déployer des réseaux de capteurs conçu uniquement pour des applications spécifiques, les réseaux de nos jours ont des nœuds avec différents capteurs physiques pour une grande variété de scénarios d'application et différents groupes d'utilisateurs (Les MICA motes contiennent déjà des capteurs de température, un magnétomètre, un accéléromètre, un microphone, et aussi plusieurs actionneurs).

Exemple de traitement d'une tâche dans un RCSF [11]

Dans certaines applications possibles, les gens peuvent interroger le groupe de capteurs dans leur propre intérêt, par exemple "L'administrateur est-il au bureau?" Ou "Y a-t-il des chaises libres dans la salle? Y a-t-il des réunions?"

Par exemple, un utilisateur du réseau de télésurveillance pourra contacter l'un des capteurs dans la zone de capture (en utilisant des liaisons radio longue distance) et effectuer les tâches suivantes: « Chaque I ms pour les prochaines T secondes, envoyez-moi une estimation de localisation de n'importe quel animal à quatre pattes dans la sous-région R du champ de capture ».

En général, le réseau doit prendre en charge plusieurs tâches.

En utilisant une communication sans fil multi sauts, cette demande est transmise aux nœuds dans la sous-région R du champ de capture. Chaque nœud utilise ses capteurs pour collecter des échantillons et les assembler pour les comparer avec les formes d'onde stockées localement. Si le nœud détecte une forme d'onde similaire à un animal quadrupède, il générera une description de l'événement toutes les I millisecondes, qui contient les éléments suivants: son propre emplacement, la valeur de code correspondant à l'animal, la force du

signal, et un certain degré de fiabilité dans son estimation. Les capteurs de la zone R doivent être coordonnés pour sélectionner la meilleure estimation. Cette dernière sera ensuite dirigée vers le demandeur de la requête.

I.2.4. Caractéristiques et contraintes des réseaux de capteurs sans fil

Alors même qu'un grand nombre d'applications mettent en jeu des RCSFs, ceux-ci ont plusieurs restrictions que ces applications doivent contourner. Par exemple, ils ont une faible puissance de calcul, une réserve d'énergie limitée et une bande passante réduite. Nous détaillerons quelques facteurs des RCSFs dans ce qui suit :

I.2.4.1. L'environnement : Les capteurs sont déployés soit à un endroit très proche du phénomène observé ou alors directement dans le phénomène lui-même. En effet, les capteurs sont souvent déployés en masse dans des endroits hostiles tels que des champs de bataille au-delà des lignes ennemies, à l'intérieur de grandes machines, au fond d'un océan, dans des champs biologiquement ou chimiquement souillés, etc. Par conséquent, ils doivent pouvoir fonctionner sans surveillance dans des régions géographiquement éloignés ou inaccessibles.

I.2.4.2. Passage à l'échelle : La surveillance d'un phénomène peut nécessiter le déploiement d'un nombre de nœuds qui est de l'ordre de plusieurs milliers de capteurs. Suivant l'application, ce nombre peut encore augmenter jusqu'à des millions de capteurs, ce qui entraîne des congestions et des erreurs de communication. Un tel déploiement, nécessite que le protocole utilisé pour la communication soit capable de détecter les erreurs et de contrôler le flux, il doit aussi exploiter la nature fortement dense des réseaux de capteurs.

La densité des nœuds dépend également de l'application pour laquelle le réseau de capteurs est employé. Une application de diagnostic de machines nécessite par exemple une densité proche de 300 nœuds par région de 25 m², tandis que la densité nécessaire pour le contrôle des véhicules ne peut pas dépasser 10 capteurs par une région de même taille [12].

I.2.4.3. Topologie du réseau : Les caractéristiques de déploiement aléatoire, fonctionnement autonome, et fréquence élevée de pannes rendent la maintenance de la topologie d'un réseau de capteurs une tâche complexe. En effet plusieurs centaines de capteurs sont déployés avec une densité pouvant être supérieur à 20 nœuds par m^3, ceci exige une bonne gestion de la maintenance de la topologie du réseau déployé. Cette maintenance de topologie se fait en trois phases :

- **Déploiement :** Les nœuds capteurs peuvent être éparpillés sur le champ de captage en masse ou placés d'une manière individuelle et ceci par le biais de plusieurs moyen tels que : les jeter d'un avion, utiliser une artillerie, roquette ou missile, ou les placer nœud par nœud d'une façon manuelle ou en utilisant des robots.
- **Post-déploiement :** Après la phase de déploiement, la topologie du réseau peut subir des changements dus aux : changement de position des nœuds, non accessibilité à cause du brouillage ou des obstacles en mouvements, épuisement d'énergie, mal fonctionnement des nœuds, ou pour des besoins de l'application.

En effet, Bien que les nœuds d'un réseau de capteurs peuvent être déployés d'une manière statique, la panne matérielle constitue un évènement très commun à cause de l'épuisement d'énergie ou la destruction. Il est possible également d'avoir un réseau de capteur avec des nœuds mobiles qui ont une mobilité très élevée. Par conséquent, la topologie du réseau de capteurs est fréquemment exposée aux changements après la phase de déploiement.

- **Redéploiement de nœuds additionnels :** Des nœuds capteurs additionnels peuvent être installés pour remplacer ceux qui sont en panne ou bien pour répondre aux besoins des tâches assignées au réseau. Cette addition entraîne la réorganisation du réseau et le changement de sa topologie. Ainsi, Une bonne gestion du réseau, faisant face au facteur de changement fréquent de la topologie d'un réseau ad hoc caractérisé par une contrainte exigeante de consommation d'énergie doit passer obligatoirement par la conception de protocoles de dissémination et de routage spéciaux pour ce type de réseau.

I.2.4.4. Tolérance aux pannes : Certains nœuds capteurs peuvent générer des erreurs ou ne plus fonctionner à cause d'un manque d'énergie, d'une défaillance matérielle ou d'une interférence [66]. Ceci ne doit pas affecter la globalité de la tâche du réseau de capteurs. En cas de défaillance, de nouveaux liens et routes doivent être établis pour assurer la continuité de la tâche de collecte des données. La réplication peut également être utilisée, tout en veillant à conserver une faible consommation d'énergie.

Le critère de tolérance aux fautes dépend essentiellement de l'environnement de déploiement du réseau. En effet, si le réseau de capteurs est destiné aux environnements à faible degré d'interférences (tel que ceux utilisés dans les bâtiments pour surveiller le taux d'humidité et le degré de température), les protocoles utilisés ne doivent pas cibler une grande tolérance aux pannes. En effet, dans ce type de réseau, il n'existe pas une grande interférence avec l'environnement, et ses nœuds ne sont pas exposés au risque d'endommagement.

Par contre, si le réseau est destiné aux applications militaires telles que la surveillance et le contrôle d'un champ de bataille, le niveau de tolérance aux pannes visé par les protocoles employés doit être très élevé. En effet, ici les nœuds sont exposés à un grand risque d'endommagement par des actions hostiles, et les informations captées sont très critiques.

Par conséquent, le niveau de tolérance aux pannes requis dépend de l'application du réseau de capteurs conçu, et les schémas de conception doivent prendre en charge ce paramètre.

I.2.4.5. Les contraintes matérielles : Malgré leur petite taille et leur faible coût, les capteurs subissent de fortes contraintes notamment d'énergie, de calcul et de stockage.

- **Dimension :** La taille réduite des capteurs peut présenter de nombreux avantages en fonction de l'utilisation prévue du système, et elle permet un déploiement flexible et simple du réseau. Cependant, la puissance des batteries utilisées pour alimenter les nœuds capteurs est limitée, par la petite taille de ces derniers.

- **Énergie :** Comme les nœuds capteurs sont des composants micro-électroniques, ils ne peuvent être équipés que par des sources limitées d'énergie (<0.5 Ah, 1.2 V). De plus, dans certaines applications, ces nœuds ne peuvent pas être dotés de mécanismes de rechargement d'énergie, par conséquent, la durée de vie d'un nœud capteur dépend fortement de la durée de vie de la batterie associée.

  Sachant que les réseaux de capteurs sont basés sur la communication multi-sauts, chaque nœud joue à la fois un rôle d'initiateur de données et de routeur également. La baisse d'énergie d'un certain nombre de nœuds entraîne un changement significatif sur la topologie globale du réseau, et peut nécessiter un routage de paquets différent et une réorganisation totale du réseau. C'est pour cela que le facteur de consommation d'énergie est d'une importance primordiale dans les réseaux de capteurs. La majorité des travaux de recherche menés actuellement se concentrent sur ce problème afin de concevoir des algorithmes et protocoles spécifiques à ce genre de réseau qui consomment le minimum d'énergie.

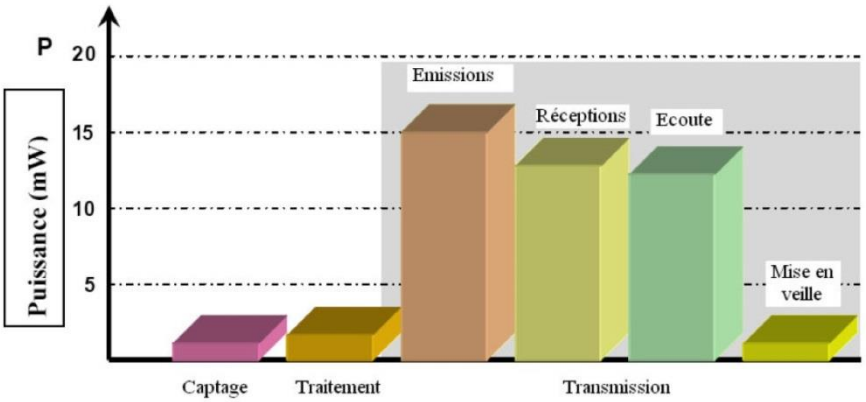


Figure I.3 : Consommation d'énergie dans un nœud capteur [71].

En effet, dans les réseaux ad hoc classiques, la consommation d'énergie est un facteur important mais ne constitue pas la première considération pour les concepteurs, car les batteries sont supposées toujours rechargeables ou remplaçables par l'utilisateur. Les chercheurs ont cependant concentré leurs efforts sur les facteurs de qualité de service dans ce type de réseau, tel que le débit de transmission et la tolérance aux pannes.

Par contre, l'énergie est considérée comme la principale et fondamentale contrainte dans les réseaux de capteurs sans fil [4]. Elle influe directement sur la durée de vie du réseau en entier. Pour cela, les concepteurs peuvent au moment du développement de protocoles négliger les autres métriques de performance telle que la durée de transmission et le débit, au détriment du facteur de consommation d'énergie.

L'énergie du nœud capteur est consommée par trois éléments : le captage, le traitement des données et la communication. Le taux d'énergie consommée dans chacun de ces trois éléments dépend de l'application du réseau de capteurs. Mais

généralement, c'est la communication qui consomme beaucoup plus d'énergie que les autres tâches du nœud capteur (voir figure I.3).

- **Puissance de traitement :** Malgré la construction des processeurs de plus en plus petits avec une puissance de calcul plus élevée, les unités de calcul et de stockage mémoire restent toujours une ressource faible pour les nœuds capteurs. Par exemple les nœuds Smart-Dust possèdent une unité de traitement avec une fréquence de 4MHZ et 512 octets de mémoire en plus d'une EEPROM de 512 octets également. Le système d'exploitation exécuté sur ces nœuds est le TinyOS, ce dernier possède 3500 octets pour le code du système, en plus d'un espace de 4500 octets disponibles pour d'autres programmes.

I.2.5. Différences entre les réseaux de capteurs et les réseaux AdHoc classiques

Les MANETs et les RCSFs partagent beaucoup de points communs mais quelques fois avec un poids différent. Parmi lesquels, nous trouvons: L'utilisation de médium de communications sans fil et de sources d'énergie limitées, la possibilité de défaillance des liaisons, une faible bande passante, une communication multi-saut, la possibilité d'auto-organisation et sécurité physique limitée.

Dans ce qui suit, nous présentons quelques points caractérisant mieux, les réseaux de capteurs par apport aux réseaux ad hoc:

- Un grand nombre de nœuds: en tenant compte de l'avantage de la petite taille des capteurs et leur faible coût dû à l'évolution de la technologie MEMS, un réseau de capteurs peut contenir des milliers de nœuds. La gestion de cet énorme nombre de nœuds est un problème majeur qui devrait être pris en considération pendant la conception des réseaux de capteurs.

- Traitement collaboratif des signaux: un autre facteur qui distingue ce type de réseaux des réseaux ad hoc traditionnels est que l'objectif final est de détecter ou d'estimer un certain nombre d'évènements de l'objet intéressé, et pas seulement la communication.

- Une utilisation efficace de la mémoire: à cause de leur petite taille, les ressources mémoires des capteurs sont restreintes. Donc à la conception des capteurs, les problèmes des tables de routage et la réplication des données doivent être pris en considération.

- Une gestion efficace de l'énergie: La contrainte d'énergie est un aspect critique dans les réseaux de capteurs. Généralement, le réseau est déployé dans une région difficile d'accès, ce qui rend presque impossible de changer ou recharger les batteries des nœuds. La durée de vie d'un réseau de capteurs est liée à sa réserve d'énergie. Pour augmenter cette durée, une meilleure gestion de la consommation d'énergie doit être instaurée.

- Auto-organisation: un réseau de capteurs doit avoir un haut niveau de tolérance aux pannes. Les nœuds sont dans la plus part des cas sujets aux pannes, ou à l'épuisement de leurs réserves d'énergie. Le coût du remplacement d'une batterie étant très élevé dans les réseaux de capteurs, les nœuds sont à éliminer du réseau. Donc le seul moyen d'assurer les fonctionnalités du réseau est d'ajouter d'autres nœuds pour couvrir les régions épuisées ce qui exige une réorganisation des nœuds du réseau.

- Communications centrées données (data centric) : contrairement aux réseaux ad hoc traditionnels qui ont un mode de communication "any to any", la destination dans les réseaux de capteurs est connue et les communications se font généralement de plusieurs sources de données vers une station de bases (many to one). L'objectif n'est donc pas de maximiser le taux de données communiquées, mais de minimiser la quantité d'informations utiles transmises vers les destinations, ce qui ne nécessite pas une large bande passante.

- Agrégation des données: Un grand nombre de nœuds peut encombrer le réseau avec les informations communiquées. Pour résoudre ce problème, quelques nœuds peuvent effectuer l'agrégation des données avant de les rediffuser vers les autres nœuds.

- Réseaux orientés applications: contrairement aux réseaux traditionnels (filaires ou sans fil) qui sont attendus de couvrir une grande variété d'applications, les réseaux de capteurs sont généralement déployés pour accomplir des tâches spécifiques. Le type d'applications a donc une grande influence sur les caractéristiques du capteur, la topologie du réseau, et les protocoles utilisés.

I.3. Dissémination des données dans les RCSFs

I.3.1. Définition de la dissémination des données

L'objectif de la dissémination des données est d'acheminer n'importe quels types d'informations (données ou requêtes) à tous les nœuds concernés par ces informations, tout en minimisant le nombre de nœuds de transmission et le coût d'énergie [1][2]. C'est un processus vital pour un bon nombre de services dans les RCSFs tel que la reprogrammation (transport de code), et la mise à jour des paramètres du système. La dissémination des données est considérée comme une des phases principales de consommation d'énergie dans la communication des RCSFs [3]. D'où la réduction des coûts de communication ainsi que l'élimination du trafic des données redondantes sont les principaux défis de la dissémination des données. Concernant les applications des RCSFs telles que la surveillance de l'environnement, la dissémination des données est très importante. En effet, Il est connu que l'inefficacité de la dissémination des données (comme les diffusions aveugles) causent des tempêtes de diffusion et bloquent toutes les communications des données dans le réseau.

I.3.2. Contraintes de la dissémination des données

En général, afin d'assurer la qualité de service (QoS) lors de la dissémination de données, quatre principales contraintes doivent être assurées [78] :

- La fiabilité : afin d'assurer un réseau fiable à 100%, deux contraintes doivent etre respectées : 1) la couverture de tous les nœuds du réseau ; 2) chaque nœud doit recevoir le bloc complet de données. Dans le cas contraire, l'inconsistance des données et la défaillance du réseau peuvent être causées.

- Un délai d'acheminement réduit : lors de la dissémination de données, un nombre important de paquets occupent le canal de transmission, ce qui peut bloquer le trafic ou rendre le réseau inefficace. De ce fait, le délai d'acheminement de données doit être le plus court possible.

- Une meilleure utilisation des ressources (énergie): l'énergie consommée lors de la dissémination de données est usée lord de la lecture/écriture des mémoires flash, la transmission radio, ainsi que l'écoute du signal lors de l'état « idle » du capteur. Comme l'opération de lecture/écriture est inévitable, la transmission radio doit être minimisée et contrôlée.

- La mise à l'échelle (scalability) : le nombre ainsi que la densité des nœuds dans un RCSF ne doivent en aucun cas affecter le processus de dissémination de données. La mise à l'échelle d'un protocole de dissémination de données est dite satisfaisante si le temps de l'accomplissement de cette opération est linéaire par rapport au nombre et à la densité des nœuds.

I.3.3. Agrégation et fusion des données

Dans un RCSF, plusieurs sources peuvent envoyer des données similaires. De plus, selon l'application, l'information captée par un seul nœud peut ne pas être très significatives. Par exemple, pour un utilisateur désirant obtenir la température d'une région donnée, l'information fournie par un seul nœud n'est pas très représentative. La température globale de la région nécessite la collecte de toutes les données captées par les nœuds de cette région. Soulignons que dans un RCSF, le traitement de données est beaucoup moins couteux en énergie que la communication. En effet, il a été montré dans plusieurs travaux scientifiques que la transmission d'un bit est équivalente, en terme d'énergie, à l'exécution d'environ 1000 instructions. Il est donc plus avantageux de privilégier le traitement local des données en les agrégeant de sorte à réduire le nombre de transmissions et avoir des données plus significatives. Les données captées sont souvent agrégées si elles atteignent le même nœud de relai au cours de leur acheminement vers le nœud puits.

L'agrégation des données peut être divisée en deux catégories comme suit :

I.3.3.1. Les opérations d'agrégation : Dans la plupart des scénarios d'applications, la façon typique d'extraire les informations d'un RCSF est de disséminer une requête d'agrégation déclarative dans le réseau, en demandant aux nœuds de surveiller périodiquement l'environnement, et de retourner le résultat agrégé vers le puits d'une façon périodique [17]. Cela peut se faire, par exemple, à travers des requêtes SQL (Structured Query Language) de base pour la fusion des données : Count, Min, Max, Sum et Average [18]. De plus, la majorité des systèmes de bases de données permettent de définir des fonctions propres à l'utilisateur (UDF) qui peuvent spécifier des agrégations plus complexes.

Exemple : calculer la température moyenne (average) dans une certaine région du champ de capture chaque 10 min. Un exemple d'une telle requête est : « *select avg (temperature) from Sensors where location in Region every 10 min* » [17].

I.3.3.2. Élimination de la redondance : Les lectures effectuées par un même capteur à différents instants ou à travers plusieurs capteurs voisins peuvent avoir de grande corrélation entre elles, et contiennent des informations redondantes. Au lieu de transmettre toute l'information corrélée vers le demandeur, il est plus efficace pour certains nœuds intermédiaires d'abréger l'information reçue pour en résulter des informations agrégées, dans

le but de réduire la quantité des données à transmettre (et diminuer donc l'énergie consommée et la bande passante utilisée pendant la transmission).

Comme exemple de capteurs corrélés, on a les capteurs de température et d'humidité dans une même zone géographique, ou des capteurs de surveillance de mouvement des véhicules. Un autre exemple intéressant de données sensorielles corrélées sont les capteurs audio (microphones) qui captent un événement commun comme un concert ou les cris de baleines [19]. La corrélation existe au niveau de la donnée captée sous deux formes :

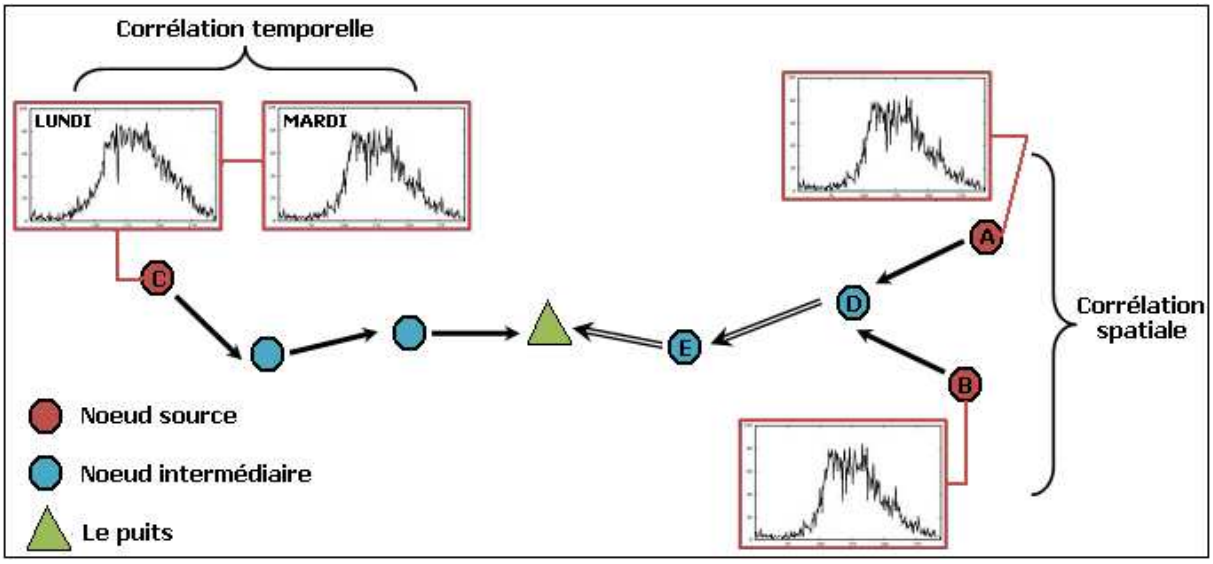


Figure I.4 : Corrélation temporelle et spatiale dans un RCSF [20].

- **Corrélation temporelle :** Le trafic des données fait preuve de fortes corrélations temporelles. Par exemple, un nœud capteur risque de trouver dans sa mémoire un ancien jour avec un trafic très similaire et il est capable de rapprocher les lectures de ce jour comme une fonction linéaire aux lectures de l'ancien jour. Cela lui permet de propager seulement peu de paramètres de dérivée vers le puits, plutôt que les séries en temps complètes. Dans la figure I.4, le nœud capteur C détecte que la série en temps du jour courant (Mardi) est fortement corrélée avec la série en temps du Lundi que le puits connaît déjà (figure I.4). Dans ce cas, le capteur expédie seulement un ensemble de paramètres de façon à ce que le puits puisse dériver la série en temps courante en se basant sur les données du Lundi, avec des bornes maximales d'erreurs définies par l'utilisateur [20].Cela démontre la force de la corrélation temporelle dans le trafic des données, et révèle une opportunité unique pour l'exploitation de cette corrélation afin d'achever des économies en communications [21].

- **Corrélation spatiale :** Le niveau de corrélation spatiale des données dépend directement de la distance séparant les différents capteurs [21]. De ce fait, des économies de communication similaires peuvent être achevées en exploitant les corrélations spatiales ; les nœuds capteurs A et B dans la figure I.4 envoient leur données en séries en temps vers le nœud D, le prochain saut dans le chemin vers le puits. Si une forte corrélation entre les deux signaux existe, le nœud D va exprimer la série en temps du nœud A comme une fonction linéaire de la série en temps du nœud B ; il expédie ensuite seulement les séries en

temps des données du nœud B et les paramètres de dérivée pour le nœud A vers le puits via le prochain saut E [20].

I.3.4. Approches de dissémination des données

Selon la façon de récolter les données et de les traiter, nous pouvons distinguer deux grandes approches (voir figure 1.5) de dissémination des données dans les RCSFs [75][68]:

- **Dissémination basée client/serveur :** ici, le puits envoie des requêtes aux nœuds capteurs. Ensuite, chaque nœud capteur va traiter la requête du puits pour lui renvoyer individuellement les données demandées. Ces données seront ensuite traitées et fusionnées au niveau du puits.

  Il est à noter que les premiers protocoles de dissémination basés client/serveur ne sont plus appropriés aux réseaux de capteurs, puisque c'est une opération très couteuse en énergie. En effet, les échanges fréquents de données épuisent rapidement les batteries des capteurs qui sont souvent non rechargeables.

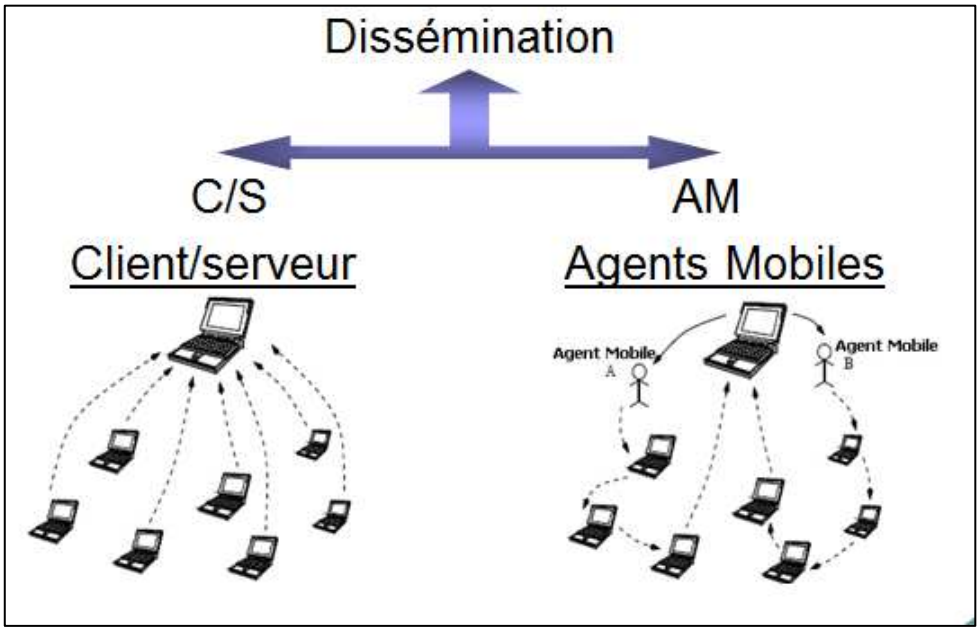


Figure I.5 : Approches de dissémination des données dans les RCSFs.

- **Dissémination basée agent mobile :** le puits expédie un ou plusieurs agents mobiles vers les nœuds capteurs. Cet agent va transporter le code de traitement des données (code d'agrégation). De cette façon, les données seront traitées et fusionnées localement au niveau des nœuds capteurs, de sorte à minimiser la taille des paquets transférés dans le réseau. Puis, l'agent va collecter ces données déjà traitées et agrégées pour les faire parvenir au puits. Ceci se fait dans le but de réduire la communication dans le réseau afin de minimiser l'énergie consommée et prolonger donc la durée de vie du réseau.

I.4. Conclusion

A travers ce premier chapitre, nous avons introduit les réseaux de capteurs sans fil en définissant leurs principes de fonctionnement, leurs caractéristiques intrinsèques, ainsi que quelques concepts liés à la dissémination des données dans ces réseaux.

De nos jours, les travaux de recherche dans le domaine de la dissémination des données dans les RCSFs se basent beaucoup plus sur l'utilisation des agents mobiles. En effet, les agents mobiles sont connus pour leur efficacité en économie d'énergie.

Nous abordons en détails le paradigme agent mobile dans les deux prochains chapitres, en l'illustrant avec plusieurs solutions pertinentes.

Chapitre II

Protocoles de dissémination des données à base d'un AM dans les RCSFs

II.1. Introduction

Dans les réseaux distribués, le paradigme classique client/serveur exige une connexion permanente entre le client et le serveur, ce qui n'est pas le cas des terminaux mobiles qui sont exposés à la perte de la connexion. Le concept d'agent mobile apparaît dans ce contexte comme une solution facilitant la mise en œuvre d'applications dynamiquement adaptables, et il offre un cadre générique pour le développement des applications réparties sur des réseaux de grande taille qui recouvrent des domaines multiples. L'envoi du code sur le serveur (nœuds sources) permet d'adapter les services distants aux exigences du client (puits) et de ne lui envoyer que les informations utiles [65].

Nous allons voir dans ce chapitre les motivations qui ont mené à l'utilisation des agents mobiles dans les réseaux de capteurs sans fil, en illustrant quelques travaux pertinents dans ce chapitre et le chapitre suivant.

II.2. Les agents mobiles

II.2.1. Définition d'un agent mobile

Un agent mobile (AM) est un processus autonome qui peut migrer entre les nœuds d'un réseau tout en maintenant son état, et il est capable de communiquer et coordonner avec d'autres agents [72].

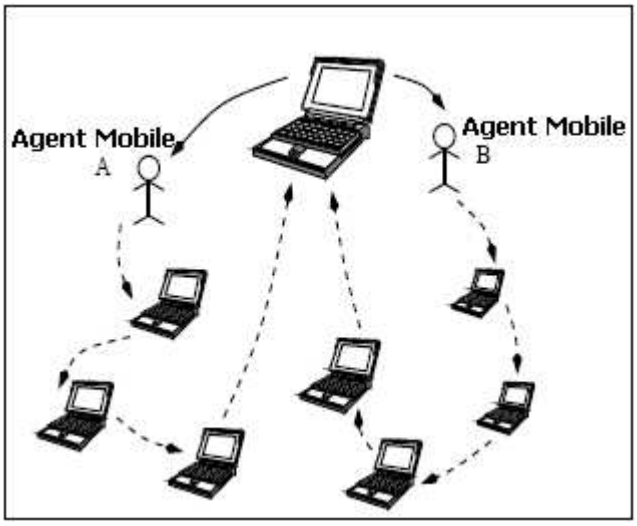


Figure II.1. Paradigme Agent Mobile.

Initialement, un AM réside dans la machine qui l'a créé. Il est ensuite expédié pour s'exécuter sur une machine distante appelée généralement hôte d'AM ou serveur. Lorsqu'un AM est expédié, le code entier de l'AM et son état d'exécution sont transférés vers l'hôte.

L'hôte fournit à l'AM un environnement d'exécution convenable. L'AM va utiliser les ressources (CPU, mémoire…etc.) de l'hôte pour exécuter sa tâche. Après avoir accompli sa tâche au niveau de l'hôte, l'AM va migrer vers une autre machine. Puisque l'état d'exécution est aussi transféré vers l'hôte, l'AM peut reprendre son exécution au point où elle s'est arrêtée au niveau de l'ancien hôte. De cette manière, l'AM va continuer son parcours jusqu'à la dernière machine de son itinéraire pour enfin revenir vers la machine qui l'a produit.

II.2.2. Caractéristiques d'un agent mobile

La mobilité des agents est considérée comme étant une des caractéristiques des agents. En effet, un agent mobile est un logiciel autonome et auto-adaptable qui est capable de se déplacer avec ses données propres, son code et son état d'exécution au cours de son exécution dans le réseau. L'agent donc a la possibilité de migrer d'un site à un autre et de suspendre ou changer son comportement selon les évènements et les circonstances qu'il rencontre [25].

En plus de la mobilité, l'AM possède les caractéristiques suivantes [65]:

- **Comportement** : Un agent est pourvu d'un programme qui lui dicte la tâche à accomplir. On dit qu'il agit par délégation pour le client. Un agent peut ainsi venir avec une compétence particulière sur une machine hôte.
- **Autonomie** : Un agent agit indépendamment du client. Il décide lui-même là où il va se déplacer et ce qu'il doit y faire, en fonction du comportement qui lui a été donné. Cela implique que l'on ne peut pas toujours prévoir l'itinéraire des AMs.
- **Mémoire** : Un agent mobile dispose d'une capacité de mémorisation lui permettant de récolter des informations sur les sites visités. Les informations mémorisées seront livrées par l'agent après son retour au client.
- **Environnement** : L'environnement dans lequel l'agent évolue est constitué :
  - par les machines qui vont accueillir l'agent lors de son exécution ; ainsi l'agent utilise les ressources (unité de calcul, mémoire, etc...) disponibles sur la machine afin d'accomplir la tâche demandée par le client,
  - par les liens réseau que l'agent utilise pour se déplacer entre les différents sites,
  - par les autres agents fixes et mobiles avec qui l'agent interagit afin de réaliser sa tâche.

II.2.3. Fonctionnement d'un agent mobile

Lors de sa création, l'agent mobile peut être initialisé avec des informations nécessaires à son exécution telles qu'un itinéraire, des préférences de son utilisateur, etc. Par ailleurs, un agent est initialisé par le système avec des informations nécessaires à son interaction avec son environnement, comme par exemple l'identité de son utilisateur.

Un Agent qui reçoit l'ordre ou décide de migrer, se déplace dans le réseau de machines et accédera localement aux différentes ressources et services offerts par ces hôtes. Pour réaliser ces opérations nous pouvons distinguer trois phases [23]:

- L'Agent mobile décide de migrer, il établit la description de sa mission,

- L'exécution de la mission par l'Agent. Ce dernier se déplace pour accéder aux différents services et aux ressources offertes par le site hôte,

- Le retour de l'Agent mobile au site de départ et la récupération éventuelle des résultats de sa migration.

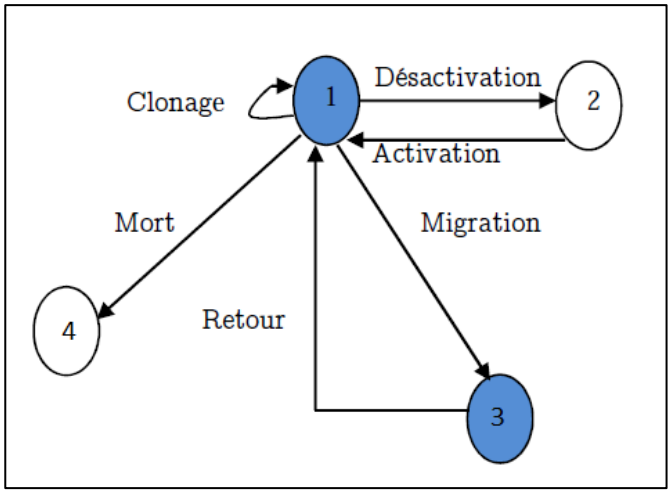


Figure II.2. Cycle de vie d'un agent mobile [23].

Ces quatre étapes nous permettent de décrire le cycle de vie d'un agent mobile (Figure II.2) expliqué dans le tableau suivant :

Etat / Transition	Explication
1	État initial : création de l'AM. C'est au cours de cette étape que l'AM aura un nom, des objectifs à atteindre et des capacités.
2	État de repos. : l'AM peut, pour des raisons diverses, être désactivé (attente des ressources par exemple). Il sera donc stocké mais pourra reprendre une activité normale quand c'est possible.
3	Migration de l'AM : C'est l'étape la plus importante. Pour satisfaire ses besoins, l'AM décide de quitter le lieu dans lequel il se trouve pour aller vers un autre site.
4	Mort de l'AM : l'agent est supprimé soit à la fin de l'exécution de ses tâches, soit c'est le site d'accueil qui décide ainsi, pour des raisons de sécurité entre autres.
Clonage	C'est une action héritée des concepts d'agent classique. L'agent aura une copie avec un nom et un identificateur différent (la copie sera à l'état initial même si son « père » ne l'est pas).
Migration	L'action de migration est un processus délicat et complexe. En fait, c'est le passage d'un environnement à un autre.
Retour	C'est une migration dans le sens inverse. Une fois l'AM a terminé l'exécution de sa tâche pour laquelle il a migré, il devra retourner à son site d'origine et continuer son activité localement.
Désactivation	Un événement s'est produit et l'agent doit être stocké et mis au repos momentanément.
Activation	Un événement s'est produit, il faut relancer l'AM à l'état où il s'est arrêté au moment de son stockage.

Table II.1 Description du cycle de vie d'un Agent Mobile.

II.2.4. Avantages des agents mobiles

Les agents mobiles sont un paradigme pour l'informatique distribuée, particulièrement convenable pour supporter le développement d'applications distribuées, des services et des protocoles dans les systèmes distribuées conventionnels de même que dans les environnements distribués très dynamiques [24].

Plusieurs motivations ont mené à l'utilisation des agents mobiles dans les réseaux distribués; nous les éclaircissons en proposant des exemples de contextes dans les RCSFs [26] :

II.2.4.1. Réduction de la charge du réseau

Les agents mobiles peuvent accéder à des ressources éloignées (ex : BD, capteurs) ou communiquer avec d'autres agents éloignés ou avec d'autres composants, en se déplaçant vers leur site et en interagissant avec eux localement, et économiser ainsi les ressources du réseau comme la bande passante et l'énergie durant la phase d'interaction. Comme exemple dans un RCSF, un nœud capteur transmet périodiquement la données captées vers le puits, qui à son tour traite ces données afin d'extraire les informations agrégées. La transmission des données captées consomme de la bande passante durant son acheminement du nœud capteur vers le puits. Un AM incorporant l'algorithme de traitement des données peut migrer vers le nœud capteur et appliquer localement cet algorithme en réduisant ainsi la consommation de bande passante. En effet, l'AM doit aussi transmettre la donnée agrégée vers le puits, mais dans ce cas, la bande passante consommée est beaucoup plus petite que celle consommée pour la transmission de la donnée brute. Cela implique aussi une réduction de la consommation d'énergie durant la communication, vu que la taille des données transportées par l'AM est plus petite que celles des données brutes.

II.2.4.2. Surmonter la latence du réseau

Les agents mobiles peuvent se déplacer vers des nœuds éloignés et appliquer localement les contrôles en temps réel pour la régulation physique/logique des dispositifs/composants en évitant ainsi le contrôle à distance qui est lourdement affecté par la latence du réseau. Comme exemple de scénario ; un capteur/actionneur éloigné qui doit être calibré pour l'exécution d'une tâche donnée, et après contrôlé en temps réel pour compléter cette tâche avec succès. Un AM équipé avec le code de calibration et de contrôle peut se déplacer vers le nœud capteur/actionneur et le calibrer localement. Dans ce sens, la latence du réseau n'affectera pas l'opération de control en temps réel qui peut être exécutée aussi dans le cas de panne de la connectivité du réseau avec la station de base.

II.2.4.3. Encapsulation de protocoles

Les AMs peuvent encapsuler des protocoles et les déployer dynamiquement et surmonter ainsi la standardisation ou les enjeux de mise à niveau. Les protocoles peuvent alors évoluer dynamiquement ou être modifiés selon la demande pour être plus efficaces. Un exemple de cela dans le routage ; un protocole de routage spécifique supportant des chemins multi sauts qui devrait être déployé dans une zone donnée du RCSF. Une tâche de coopération d'AMs implémentant le protocole de routage peut être créée et migrée vers les nœuds capteurs de la

zone ciblée dans le RCSF. Lorsqu'une nouvelle version du protocole est définie, une nouvelle tâche implémentant le nouveau protocole sera lancée pour remplacer la version antérieure du protocole.

II.2.4.4. Exécution asynchrone et autonome

L'asynchronisation et l'autonomie sont des propriétés distinctives des AMs. Une fois injectés dans le réseau, les AMs peuvent exécuter d'une façon autonome la tâche programmée et supporter les opérations déconnectées en opérant d'une façon asynchrone tout en respectant le processus ou le composant qui l'a généré. Ces propriétés sont très importantes dans les environnements dynamiques où la topologie du réseau change rapidement et les connections sont non stables d'une façon à ce qu'elles peuvent être établies seulement pour de courtes périodes de temps. Un exemple de tâche asynchrone dans un RCSF : périodiquement, l'utilisateur du RCSF demande aux nœuds capteurs de lui envoyer leur taux d'énergie courant. Pour cela, un AM créé par l'utilisateur, peut se balader dans le réseau, nœud par nœud, d'une façon autonome pour collecter l'information désirée et finalement rapporter cette information d'une façon asynchrone vers le demandeur.

II.2.4.5. Adaptation dynamique

Les AMs peuvent capter leur environnement d'exécution et réagir automatiquement aux changements. D'ailleurs les tâches transportées par l'AM peuvent se distribuer à travers les nœuds du réseau pour régler un problème particulier d'une façon coopérative. L'adaptation dynamique peut supporter le comportement autonome d'un RCSF ; en fait, les RCSFs sont des systèmes à longue exécution dans lesquels les pannes des dispositifs sont accablées. L'adaptation concernant les pannes des dispositifs est déjà devenue un aspect important dans ce domaine. Un exemple est donné dans le point 7.

II.2.4.6. Orientation à l'hétérogénéité

Les AMs peuvent être utilisés comme des intégrateurs des systèmes hétérogènes. Ils peuvent agir comme des couvertures ou des médiateurs à travers les systèmes basés sur différents matériaux et logiciels. Ils peuvent aussi être convertis pour franchir le bord à travers des systèmes hétérogènes. Un scénario intéressant est : l'intégration d'un RCSF avec un réseau IP. Un AM peut être envoyé vers la passerelle entre le RCSF et le réseau IP pour agir comme couverture du RCSF. N'importe quelle information à collecter du RCSF est demandée à l'AM de la part des composants résidants dans le réseau IP. L'AM, à son tour, va traduire une telle requête en une requête spécifique et la soumettre au RCSF, et une fois qu'il reçoit la réponse, il va la renvoyer au composant l'ayant demandée.

II.2.4.7. Robustesse et tolérance aux fautes

Comme les AMs traitent l'habilité à réagir dynamiquement aux situations non favorables et aux événements inattendus, ils peuvent supporter la construction d'un système distribué robuste et tolérant aux fautes. Un scénario fréquent de tolérance aux fautes dans les RCSFs est le suivant : Un nœud capteur qui va s'éteindre dû au déchargement de ses batteries, alors l'activité de captage devrait être activée dans un nœud capteur voisin en temps réel. Tous les

AMs s'exécutant sur des nœuds capteurs non chargés vont migrer vers un autre nœud équivalent de façon à continuer leur activité après la migration.

Cependant, le principal inconvénient des AMs est le risque en sécurité. Par exemple, un AM malicieux peut endommager un hôte particulier, ou alors un hôte malicieux peut modifier le fonctionnement de l'agent…etc.

II.2.5. Comparaison entre les paradigmes "Client/Serveur" et "Agent mobile"

Dans le paradigme client/serveur, un client demande un service auprès du serveur d'une façon interactive et synchrone, ceci signifie que le client est bloqué tant que le serveur n'a pas répondu. Tandis qu'au niveau du modèle agent mobile, l'agent se déplace d'une façon autonome entre les machines d'un réseau pour réaliser certaines tâches localement. Ces agents s'exécutent d'une façon asynchrone ce qui permet de dire que ce paradigme est mieux adapté que le client/serveur à des traitements longs nécessitant des interrogations fréquentes du serveur [29]. En effet, le principal but de l'utilisation d'un agent mobile est la réduction du trafic, ainsi en envoyant l'AM là où les tâches s'effectuent, les messages échangés deviennent locaux et libèrent d'autant la charge du réseau. Ceci convient parfaitement aux caractéristiques et exigences des RCSFs, dont les applications visent à minimiser le trafic pour conserver l'énergie et prolonger ainsi la durée de vie du réseau.

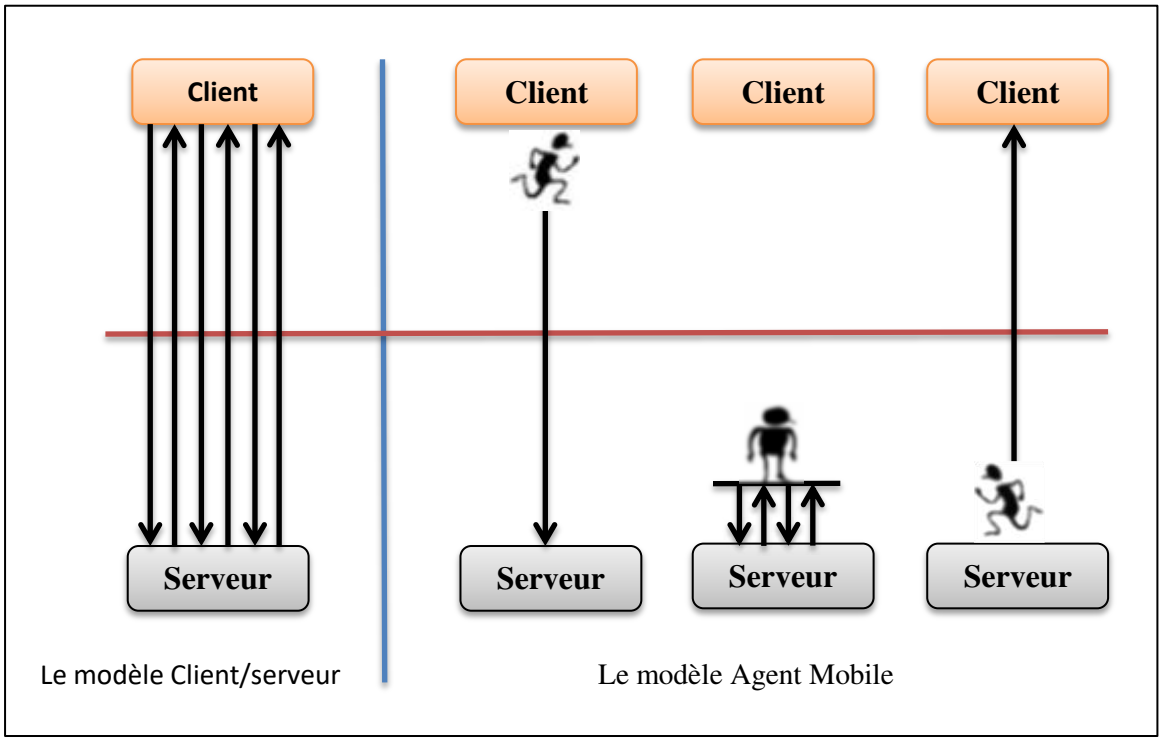


Figure II.3. Paradigme Client/serveur Versus Agent Mobile [29].

II.2.6. Critères de conception des agents mobiles

Les agents mobiles impliquent aussi des coûts et des risques. Les plus évidents sont ceux abordés à la reprise à distance: la migration et l'exécution de l'agent doivent être portatives à travers des plateformes hétérogènes. En plus, l'utilisation des agents mobiles impliquent des

contraintes additionnelles concernant la communication (agent-à-agent et agent-à-hôte) et la sécurité [5].

II.2.6.1. La migration de l'agent

La migration de l'agent dans un système de machines hétérogènes pose un problème car la représentation du code et des données du processus est dépendante de la machine sur laquelle il s'exécute. Dans le cas de machines avec des architectures différentes, l'état d'un agent risque de ne pas être compris. Les solutions proposées sont :

- la traduction de l'état de l'agent de la représentation source à la représentation finale, ce qui implique autant de traducteurs que de couples d'architectures différentes.
- la représentation de l'état dans un format standard intermédiaire, indépendante de la machine et compris par toutes les machines.

II.2.6.2. La communication

Le maintien des communications entre agents est un autre problème rencontré lors de la migration de l'agent. En effet, que se passe-t-il lorsqu'un message arrive à un agent en cours de migration? Dans ce cas, on doit considérer que l'état d'agent en cours de migration est un état particulier dans lequel il ne peut recevoir de messages. Dans ce cas il devra prévenir les autres agents.

II.2.6.3. La sécurité

La sécurité est un sérieux problème qui présente un centre d'intérêt de multiples travaux de recherches à l'heure actuelle. Un agent doit se protéger des attaques extérieures par rapport aux autres agents, à ses environnements d'exécution et pendant son déplacement sur le réseau. Son intégrité et la confidentialité des données qu'il transporte doivent être assurées. Cette problématique reste encore ouverte nécessitant ainsi plus d'attention de la part de la communauté de chercheurs et des connaisseurs du domaine.

II.2.7. Planification d'itinéraire des agents mobiles

La majorité des travaux de recherche dans le domaine de déploiement de système d'agents mobiles dans les RCSF se basent sur l'optimisation des performances en terme de latence et consommation d'énergie des nœuds à travers la planification efficace d'itinéraire d'agent mobile. L'itinéraire suivi lors de la migration de l'agent peut avoir un impact significatif sur la consommation d'énergie et le temps de réponse. Trouver une séquence optimale de nœuds sources à visiter par les AMs est un problème difficile à résoudre (NP-hard problem) [27].

Un itinéraire est défini comme étant le chemin suivi par un agent mobile lors de sa migration [50]. La planification d'itinéraire peut être établie par la station de base (Sink) et/ou par l'agent mobile. De ce fait, la planification d'itinéraire peut être classifiée en trois principales catégories comme suit:

II.2.7.1. Planification statique

Les nœuds sources à visiter ainsi que l'itinéraire de l'agent mobile sont totalement déterminés par la station de base (Sink) avant que l'agent mobile ne soit expédié [28]. Le

problème avec cette planification, est que l'agent peut ne pas être en mesure de se déplacer selon son itinéraire prédéterminé en raison de défaillances de liaison ou de l'épuisement des nœuds. Aussi, lors de la collecte périodique d'informations de topologie du réseau qui subit un coût supplémentaire [50].

II.2.7.2.Planification dynamique

L'agent mobile détermine de manière autonome les nœuds sources à visiter ainsi que le chemin de sa migration au niveau de chaque saut selon l'état actuel du réseau. Ceci offre une grande flexibilité qui permet d'éviter les liens et nœuds défaillants.

II.2.7.3. Planification hybride

Dans ce cas, la décision de l'ensemble des nœuds sources à visiter est fournie par la station de base (Sink). Tandis que, l'ordre de visite est déterminé dynamiquement par l'agent mobile.

II.2.8. Implémentation des agents mobiles dans les RCSFs

Comme illustré dans les sections précédentes, l'utilisation d'AMs dans les réseaux informatiques a des avantages mais aussi des inconvénients, tels que le code de mise en cache et la sécurité dans certains scénarios. Malgré tout, ils sont utilisés avec succès dans différentes applications telles que la programmation parallèle, la collecte des données, le e-commerce et l'informatique mobile. Tel que décrit dans [30], de nombreux avantages inhérents (par exemple la scalabilité et la sensibilisation d'énergie) de l'architecture AM la rend plus adaptée aux RCSFs que l'architecture client/serveur. En effet, les agents mobiles peuvent être utilisés pour réduire considérablement le coût de communication, spécialement à travers de faibles bandes passantes, en déplaçant la fonction de traitement vers la donnée plutôt qu'apporter la donnée vers un processeur central.

Les agents mobiles peuvent supporter la programmation des RCSFs au niveau de différentes couches [26]:

II.2.8.1. Couche application

Des agents mobiles peuvent être utilisés pour la conception et la programmation des abstractions par lesquelles les applications du RCSF peuvent être effectivement développées ;

II.2.8.2. Couche middleware

Des agents mobiles peuvent être utilisés pour l'implémentation des noyaux des services du RCSF comme l'agrégation des données, la fusion et la dissémination, ainsi que le déploiement dynamique de nouveaux services via la dissémination efficace du code ;

II.2.8.3. Couche réseau

Des agents mobiles peuvent être utilisés comme les capsules dans les réseaux actifs [31] pour le routage multi sauts et d'autres services du réseau.

II.2.9. Modèles d'algorithmes à base d'agents mobiles dans les RCSFs

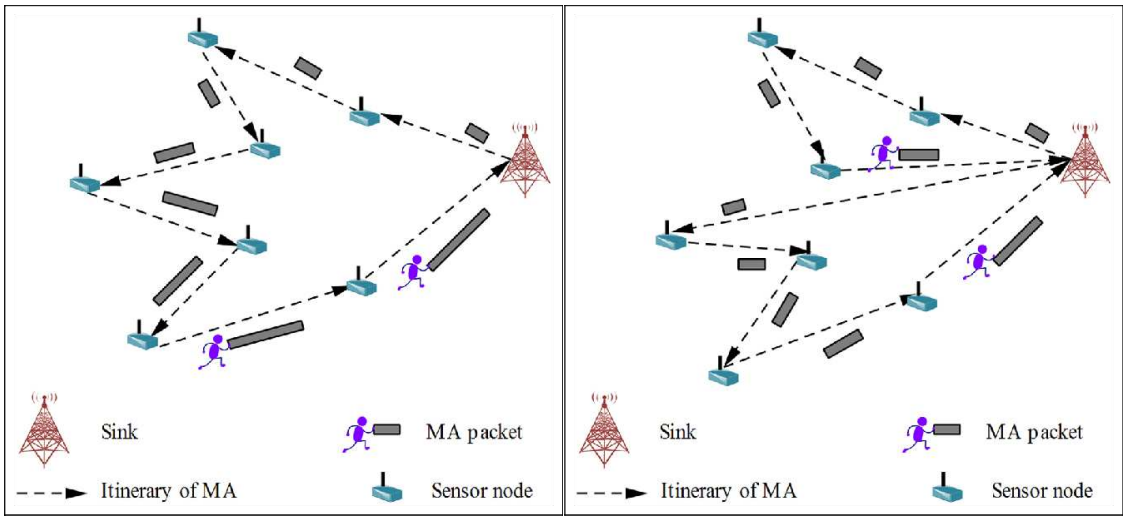
II.2.9.1. Modèle à un seul AM

Appelé souvent SIP (Single itinerary planning), La figure II.4 (a) représente un modèle de réseau de capteurs qui se base sur un seul agent mobile. Ici, la récupération des informations détectées par tous les capteurs sources, est effectué à travers l'envoie et le passage d'un seul agent mobile [74], ce qui conduit à :

- La réduction de la consommation de la bande passante.
- L'introduction d'une flexibilité supplémentaire dans le réseau.

Cependant, l'utilisation d'un seul agent mobile dans les RCSFs ne s'applique qu'aux réseaux de petite taille. L'augmentation de la taille du réseau conduit à l'augmentation du nombre de nœuds sources. La distribution de ces derniers deviendra plus compliquée. Cela provoque les problèmes suivants [50]:

- Grande latence causée par la grande échelle du réseau de capteurs, dans lequel de nombreux nœuds sources doivent être visités par un seul agent.
- Grand risque de perdre l'agent mobile à cause de sa migration vers plusieurs nœuds sources.
- Les nœuds capteurs dans l'itinéraire de l'agent s'épuisent en énergie plus rapidement que les autres nœuds.
- La taille de l'agent mobile augmente continuellement au cours des visites de tous les nœuds sources.



(a) SIP (b) MIP

Figure II.4 Modèles d'algorithmes à base d'agents mobiles [74]

II.2.9.2. Modèle à plusieurs AMs

Pour résoudre les problèmes du modèle SIP cités ci-dessus, les chercheurs ont proposé l'utilisation de plusieurs agents mobiles au lieu d'un seul agent mobile.

L'idée principale d'un modèle qui se base sur de multiples agents mobiles (MIP-Multiple Itinerary Planning) est de partager la tâche de récolte de données entre plusieurs AMs. Il est à noter qu'un protocole MIP est composé de deux ou plusieurs protocoles SIP travaillant simultanément pour visiter des groupes de nœuds sources [74].

La figure II.4 (b) donne un exemple de ce modèle avec deux agents dans lequel, les nœuds sont divisés en deux groupes, et les nœuds dans chaque groupe sont visités par un agent particulier qui est expédié à partir de la station de base.

La performance de ce modèle peut dépasser celle du modèle à base d'un seul agent mobile si les nœuds sont bien regroupés et les itinéraires sont bien conçus. Cependant, un système basé sur de multiple AMs peut, si la solution est mal adaptée, ne pas améliorer les performances et introduire encore plus de trafic de transmission dans le réseau. Par conséquence, le système à plusieurs AMs devrait découvrir le meilleur équilibre sur le compromis entre la performance et l'efficacité.

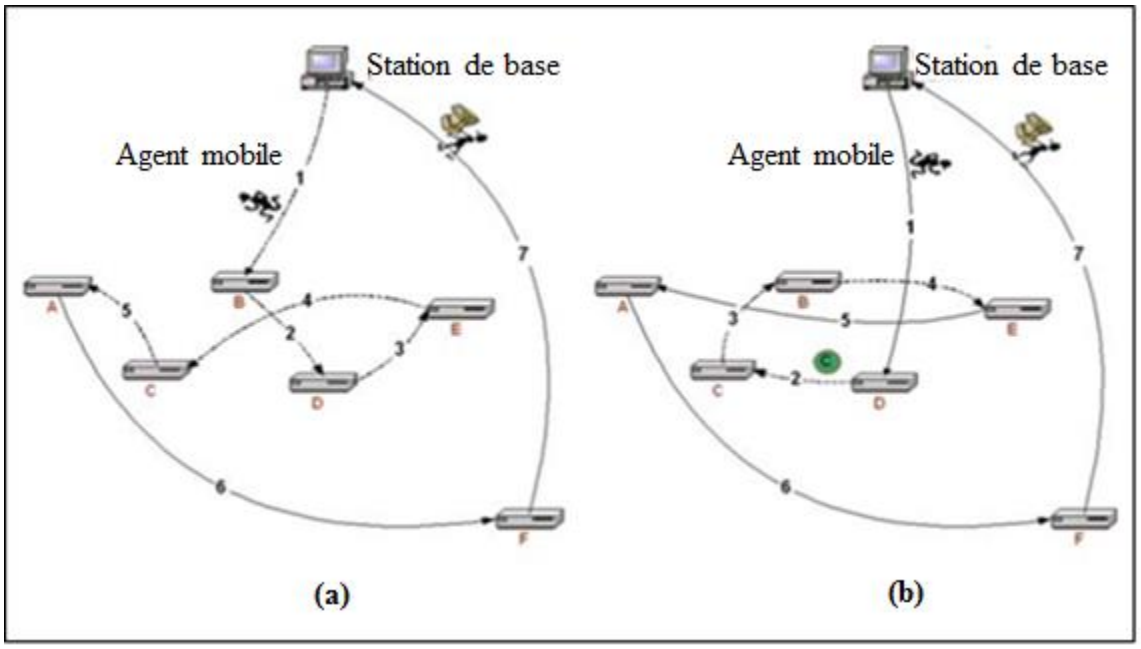
II.3. Les protocoles de dissémination des données basés SIP

II.3.1. LCF (Local Closest First) et GCF (Global Closest First)

Dans l'article [34], deux algorithmes heuristiques sont proposées:

1) LCF algorithm (Local Closest First) qui cherche le prochain nœud ayant la distance la plus courte par rapport au nœud actuel, et

2) GCF algorithm (Global Closest First) qui cherche le nœud le plus proche du centre du réseau.



(a) LCF (b) GCF

Figure II.5 : Itinéraire des algorithmes GCF et LCF.

Le principal atout de ces deux algorithmes est qu'ils sont associés à une faible complexité de calcul. Cependant, l'itinéraire de l'algorithme LCF dépend fortement de l'emplacement actuel de l'agent mobile. Donc la recherche de la prochaine destination se fait parmi les nœuds sources adjacents à l'emplacement actuel de l'agent mobile, au lieu de regarder la matrice des distances du réseau global. Par conséquent, les nœuds à visiter sont généralement associés à un coût de migration élevé (voir, par exemple, les deux derniers sauts, 6 et 7 dans la figure II.5(a). D'autre part, GCF produit dans la plupart des cas un itinéraire plus désordonné que LCF et les déplacements de l'agent mobile répétitifs autour du centre de la région, entraînant des chemins longs et de mauvaise performance.

II.3.2 MADD (Mobile Agent based Directed Diffusion)

Le protocole MADD [36] (Mobile Agent based Directed Diffusion) est une combinaison du l'approche MAWSN (Mobile Agent based Wireless Sensor Network) [35] avec DD [11] (Directed Diffusion).

L'approche MAWSN (Mobile Agent based Wireless Sensor Network) [35] propose d'utiliser le paradigme agents mobiles (AM) pour réduire et agréger les données dans un réseau de capteurs à architecture plate. Elle accomplie plusieurs tâches concurrentes associées avec de petites quantités de données par un seul paquet transportant plusieurs requêtes, et fusionne après leurs résultats dans un seul paquet afin de diminuer le coût de communication.

Dans MADD, dès qu'une nouvelle tâche est reçue en demande par une application, le puits diffuse initialement un paquet d'intérêt afin de localiser les sources qui vont traiter la tâche et établir les gradients.

La diffusion initiale de l'intérêt permet à chaque nœud capteur (nœud intermédiaire ou nœud source) de mettre en place des gradients d'exploration qui seront utilisés pour délivrer les messages d'exploration destinés à la mise en place des routes et leur maintenance. Lorsque les nœuds sources dans la région cible reçoivent les paquets d'intérêt, ils diffuseront des données d'exploration vers le puits individuellement.

Le nœud puits décide donc de la liste des nœuds sources qui seront visités par l'AM sans avoir à déterminer leur séquence de visite, celle-ci sera décidée dynamiquement au fur et à mesure que l'AM se déplace entre les nœuds. Dans cette liste, il y a seulement deux nœuds sources dont les positions sont importantes, à savoir le premier et le dernier nœud source que l'AM devrait visiter. En effet, la source qui est la dernière (respectivement la première) à envoyer les messages d'exploration, au puits est choisie comme FirstSource (respectivement LastSource).

Parmi tous les voisins d'un nœud capteur, seul le voisin qui a transmis en premier le message d'exploration d'une source spécifique va être choisi comme prochain saut dans une structure locale ToSourceEntry. Par exemple, lorsqu'un AM arrive au nœud A, le nœud va choisir le prochain nœud le plus proche dans sa ToSourceEntry montré dans la première colonne de la figure II.6. Puisque la plus basse latence du nœud B est la plus petite, il en conclu que le nœud B est le nœud source le plus proche du nœud A et il est choisi comme NextSrc.

MADD divise la période de la tâche entière en plusieurs cycles séquentiels, où chaque cycle requiert l'AM pour visiter tous les nœuds ciblés et retourner le résultat des données collectées vers le puits.

ToSourceEntry (exploratory message SeqNum = 5)					
	Source	*A*	*B*	*C*	*D*
A	*NextHop*	—	*B*	*B*	14
	Lowest Latency (ms)	—	4.46	8.24	16.32
B	*NextHop*	*A*	—	*C*	*C*
	Lowest Latency (ms)	4.47	—	4.43	12.89
C	*NextHop*	*B*	*B*	—	16
	Lowest Latency (ms)	8.16	4.32	—	8.52
16	*NextHop*	15	*C*	*C*	*D*
	Lowest Latency (ms)	9.65	7.56	4.86	5.08
D	*NextHop*	10	16	16	—
	Lowest Latency (ms)	14.15	12.67	8.73	—

Figure II.6 La structure *ToSourceEntry* dans MADD.

Dans le premier cycle, en plus que l'AM se déplace d'une source à l'autre pour collecter et agréger l'information, il copie aussi le code de traitement dans la mémoire locale de chaque nœud source. Au début de chaque cycle, le premier nœud source va construire un autre AM à partir de sa mémoire et l'expédie pour initialiser un nouveau cycle. Puisque le code de traitement a déjà résidé dans la mémoire de chaque nœud source après le premier cycle, l'AM ne va plus transporter ce code de traitement dans les prochains cycles. Lorsque la tâche entière sera terminée, tous les nœuds sources vont se débarrasser du code de traitement.

II.3.3. AbDD (Agent based Directed Diffusion)

Afin d'assurer qu'un chemin de routage ne sera pas épuisé très tôt, AbDD (Agent based Directed Diffusion) adopte un schéma combiné [38], qui prend en considération et le coût du routage Ci (nombre de sauts vers le destinataire) et l'énergie restante Ei d'un nœud Ni. L'agent mobile au nœud Ni sélectionne le nœud voisin Nj, en se basant sur la valeur H, en utilisant la formule suivante :

$$H_1 = \underset{j \in FT_i}{MAX} \frac{E_j}{C_j}$$

Pour effectuer ces calculs, chaque nœud met en place une table d'expédition des données (FT) basée sur les informations locales obtenues dans la première phase. Afin que les nœuds mettent à jour leur table d'expédition, ils utilisent les messages Hello [38].

AbDD utilise deux types d'agents ; Agents Stationnaires (AS) et Agents Mobiles (AM).

L'AS réside dans chaque nœud source. Il surveille toutes les activités pour le nœud et met à jour ses connaissances en conséquence. Ces connaissances sont critiques et sont utilisées par l'AM pour prendre des décisions de sélection du prochain saut. L'AS maintient le cache d'intérêt, les modèles des autres, son modèle et les coûts d'initialisation du routage.

La séquence de visite des nœuds sources est déterminée par le puits avant d'expédier l'AM. L'AM va se déplacer à travers les nœuds capteurs pour atteindre les nœuds sources. À chaque saut, il consulte les modèles des autres nœuds et fait sa décision pour sélectionner le prochain saut (voisin), en se basant sur le niveau de batterie maximum et le coût de route minimum. Une fois la Durée de l'intérêt expire, l'AM supprime tous les codes d'applications spécifiées de tous les nœuds sources.

II.3.4. IEMF (Itinerary Energy Minimum for First-source-selection) et IEMA (Itinerary Energy Minimum Algorithm)

Une autre solution efficace en énergie est décrite dans [39], qui s'appelle IEMF (an Itinerary Energy Minimum for First-source-selection). IEMF adopte la méthode round robin, dans laquelle chaque nœud est provisoirement sélectionné comme premier nœud source. Ensuite, l'algorithme LCF est appliqué aux nœuds sources restants. Un tel processus génère différents itinéraires candidats, chaque itinéraire correspondant à un coût énergétique. Par la suite, un itinéraire avec le coût énergétique le plus bas est choisi par l'IEMF.

Une solution itérative d'IEMF appelée IEMA (Itinerary Energy Minimum Algorithm) est aussi proposée dans le même travail. Dans IEMA, les coûts énergétiques sont employés pour décider de chaque prochain nœud source à visiter.

II.3.5. Comparaison des performances des protocoles à un seul AM

Algorithme	Scalabilité	Robustesse	Planification d'itinéraire	Avantages	Inconvénients
LCF	limitée	faible	statique	Facile à mettre en œuvre	Coût de migration élevé
GCF	limitée	faible	statique	Complexité de calcul inférieure à LCF.	Itinéraire long
MADD	limitée	moyenne	hybride	Choix efficace de First-src et Last-src	Prend en considération seulement les positions géographiques
AbDD	limitée	moyenne	hybride	Approche distribuée tolérante aux pannes	Première phase coûteuse en énergie
IEMF	limitée	faible	statique	Minimum coût énergétique de l'itinéraire	Estimé à avoir n fois la complexité de LCF
IEMA	limitée	moyenne	statique	Estimation du coût minimum pour toutes les itérations.	Grande complexité de calcul

Table II.2 Comparaison des performances des protocoles SIP [45][50].

Le tableau ci-dessus compare les performances de plusieurs approches SIP dans les RCSFs. Nous remarquons que les protocoles qui prennent en considération la tolérance aux pannes (qui est une caractéristique principale des RCSFs) sont plus robustes que les autres.

Cependant, ils sont tous limités en scalabilité, puisqu'un seul AM devrait parcourir séquentiellement tous les nœuds sources du réseau.

II.4. Limites des protocoles à un seul AM

Les algorithmes avec un seul agent mobile ont une efficacité élevée lorsque les nœuds sources sont répartis géographiquement proches les uns des autres et que le nombre de nœuds sources n'est pas élevé. Néanmoins, les restrictions imposées à l'utilisation d'un seul AM pour effectuer l'intégralité de la tâche rendent l'algorithme peu fiable dans les applications où un grand nombre de nœuds sources doivent être visités.

Par conséquent, l'utilisation d'un seul agent mobile dans un réseau de capteurs à grande échelle souffre de quelques limitations:

- Une grande latence lorsqu'il y a un grand nombre de nœuds sources à visiter;
- Les nœuds capteurs dans l'itinéraire de l'AM s'épuisent en énergie plus rapidement que les autres nœuds;
- Au cours de la visite des nœuds sources, la taille de l'AM augmente continuellement;
- La transmission de l'AM consommera plus d'énergie lors de son retour vers le nœud puits;
- La quantité croissante de données accumulées par l'agent mobile pendant sa migration augmente le risque de perte des données;
- Ainsi, plus l'itinéraire est long, plus la migration par un seul agent est risquée.

II.5. Conclusion

En déplaçant le code de traitement vers la donnée, un AM peut éviter la transmission des données intermédiaires dans le réseau, continuer le travail même en présence de déconnexions dans le réseau, et compléter donc l'ensemble de la tâche plus efficacement que dans les solutions traditionnelles client/serveur. Ceci rend le paradigme AM plus adapté aux RCSFs qui ont plusieurs contraintes et limites en ressources [63] [64]. En effet, nous avons étudié, dans ce chapitre, quelques travaux pertinents qui ont proposé l'utilisation d'un agent mobile pour la collecte efficace des données dans les RCSFs. Cependant, l'unicité de l'AM peut engendrer des dégradations dans le fonctionnement des réseaux à grande échelle. Pour éviter cela, des algorithmes employant plusieurs AMs ont été proposés pour une collecte de données efficace à travers de grands réseaux de capteurs.

Toutefois, la conception des algorithmes à base de multiples agents mobiles dans les réseaux de capteurs sans fil s'avère un défi intéressant. Nous allons voir dans le chapitre suivant les travaux les plus pertinents des algorithmes à plusieurs AMs (MIP).

Chapitre III

Protocoles de dissémination des données à multiples AMs dans les RCSFs

III.1. Introduction

Les agents mobiles peuvent être utilisés pour réduire considérablement le coût de communication, spécialement à travers de faibles bandes passantes, en déplaçant la fonction de traitement vers la donnée plutôt qu'apporter la donnée vers un processeur central.

Dans le chapitre précèdent, nous avons vu quelques protocoles employant un seul agent mobile pour collecter toutes les données du réseau. Cependant, l'unicité de cet AM dans un réseau à grande échelle rend cette tâche très lourde et coûteuse, puisque un unique AM devrait collecter toutes les données du réseau séquentiellement, et trainer donc avec lui le paquet de données tout au long de son parcours. Ceci engendre une grande dégradation des performances du réseau (grande consommation énergétique) et une faible qualité de service (grande latence et risque de perte de données). Pour pallier ce problème dans les réseaux à grande échelle, plusieurs protocoles employant multiples agents mobiles ont été proposés dans le but de gagner en temps de réponse tout en diminuant la consommation énergétique et offrir une meilleure fiabilité.

Dans ce chapitre, nous allons voir quelques protocoles pertinents employant plusieurs agents mobiles dans les RCSFs, que nous classifions en cinq catégories.

III.2. Les défis de conception des protocoles à multiples agents mobiles

Bien que les algorithmes MIP (Multiple Itinerary Planning) ont pour objectif de surmonter les faiblesses des algorithmes SIP (Single Itinerary Planning), leur conception est complexe en raison de plusieurs défis qui entrent en considération [32] :

III.2.1. La détermination du nombre optimal d'agents mobiles

Le nombre d'agents mobiles est un facteur important dans la planification d'itinéraire. Un nombre élevé d'agents produit plus de flux de transmission. Par conséquent, plus d'énergie va être consommée. Cependant, l'utilisation d'un nombre réduit d'agents mobiles pose le problème de retard dans le temps de réponse. Donc, il faut toujours songer à chercher le nombre optimal d'agents mobiles selon l'application en question.

III.2.2. La division des nœuds sources dans des sous-ensembles de groupes

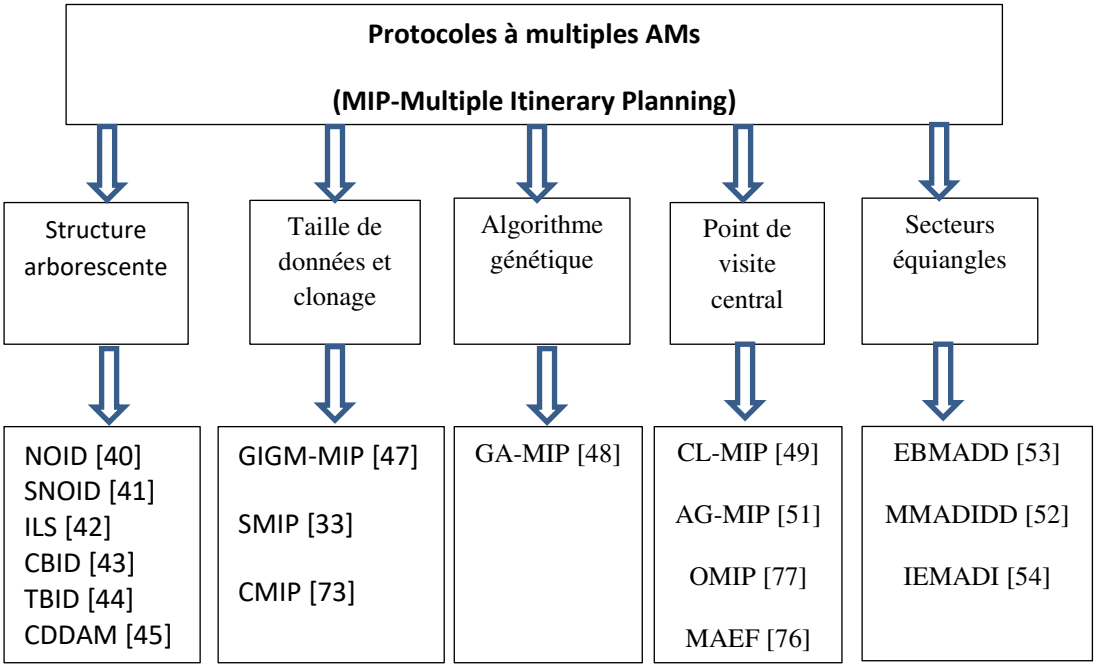
Il faut sélectionner un ensemble de nœuds sources à visiter par chaque agent mobile, donc ils doivent être regroupés et assignés à leurs agents correspondants de manière optimale. En outre, les nœuds-sources dans le même groupe doit être liées entre eux géographiquement (proches les uns des autres).

III.2.3. La conception d'itinéraire optimal de chaque agent mobile

Après avoir déterminé le nombre des agents mobiles avec leurs groupes de nœuds-sources correspondant, le chemin de chaque agent devrait être déterminé (l'ordre de visite des nœuds sources). Nous pouvons traiter indépendamment la définition de chemin comme un problème d'optimisation d'itinéraire [33].

III.3. Protocoles de dissémination des données basés MIP

Plusieurs protocoles ont été proposés dans la littérature utilisant de multiples agents mobiles pour collecter et agréger les données dans les réseaux de capteurs sans fil. Nous



avons étudié les plus pertinents parmi eux, que nous avons groupés en différents sous-groupes opérant sur des principes similaires.

Figure III.1 Classification des protocoles MIP.

La figure III.1 illustre une classification des protocoles de collecte et agrégation des données à base de multiples agents mobiles dans les réseaux de capteurs sans fil selon leur méthode de détermination du nombre d'agents mobiles et leur itinéraire.

III.3.1. Protocoles à base de structures arborescentes

L'algorithme NOID [40] a été proposé pour trouver le nombre optimal d'agents mobiles en MIP. Cet algorithme groupe itérativement les nœuds capteurs dans le réseau pour séparer les sous-arbres qui sont reliés progressivement au puits. Après que NOID finisse de construire les sous-arbres, le puits expédie un agent mobile à chaque sous-arbre. NOID surpasse les approches SIP (par exemple, LCF[34], GCF[34]), en termes de coût de fusion de données et temps de réponse global, mais il endure une complexité de traitement élevée en déterminant les itinéraires des agents.

[41] a proposé une version améliorée de l'algorithme NOID nommée SNOID (second near-optimal itinerary design). SNOID diffère de NOID en considérant le cout de communication entre les nœuds lors de la construction des itinéraires des agents mobiles. Le nombre d'agents mobiles dans SNOID est déterminé en divisant la région autour du puits en des zones concentriques. Les nœuds se trouvant en dessous du rayon de la première zone autour du puits seront les points de départ de chaque itinéraire. Le premier rayon de zone peut être obtenu par aRmax, où a est un paramètre d'entrée dans la plage [0, 1] et le Rmax

est la portée de transmission maximum de n'importe quel nœud capteur. Chaque itinéraire d'agent mobile commence à partir des zones près du puits et s'étend vers les zones externes.

De même, [42] a proposé une méthode méta-heuristique appelée ILS (iterated local search). Cet algorithme est semblable aux autres algorithmes MIP à base d'arbre (NOID et SNOID), mais il diffère en considérant l'augmentation de la taille du paquet de l'agent mobile, aussi bien que la consommation d'énergie due aux migrations entre les nœuds intermédiaires lors de la construction de l'itinéraire de l'AM.

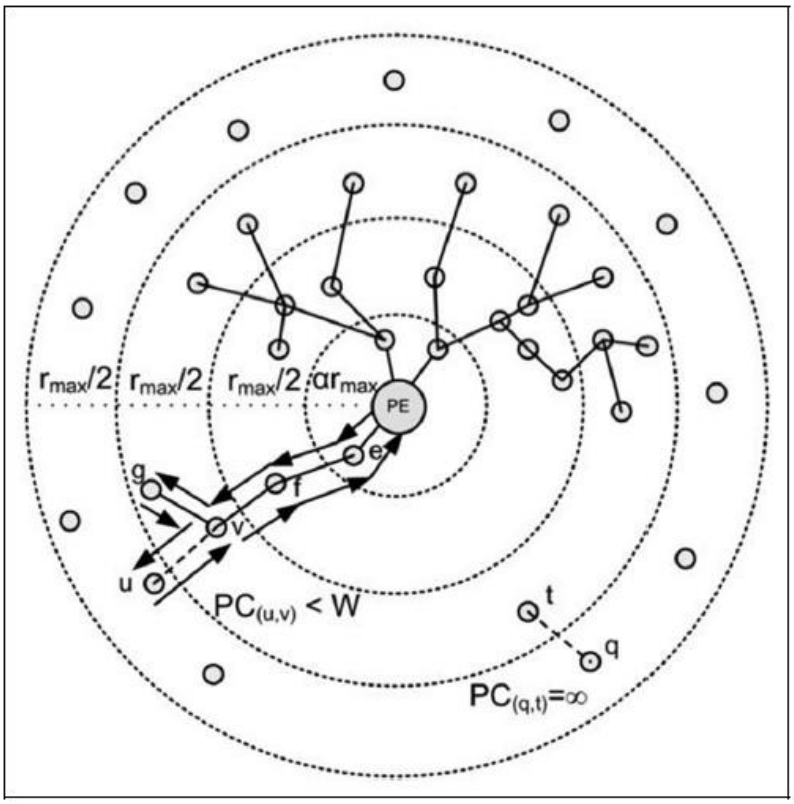


Figure III.2 Partitionnement de la zone cible en zone concentriques dans SNOID.

[43] a proposé un algorithme CBID où des agents sont expédiés en parallèle qui visitent séquentiellement des nœuds disposés en structures arborescentes. Après avoir visité un nœud avec deux ou plusieurs nœuds enfants, l'agent principal crée des copies (clones) de lui-même. Chaque agent visite une branche secondaire de l'arbre. Quand tous les agents esclaves retournent de nouveau à leur nœud parent, ils remettent leurs données rassemblées à l'agent principal. Dans CBID, le paquet de l'agent doit transporter l'information supplémentaire à propos du lieu de son clonage.

Dans [44], les auteurs ont proposé un algorithme de conception d'itinéraire glouton à base d'arbre (TBID) afin de trouver des itinéraires presque optimaux pour des agents multiples. Cet algorithme est exécuté centralement au niveau du puits et détermine statiquement le nombre d'agents qui devraient être utilisés ainsi que leurs itinéraires. L'idée principale de l'algorithme TBID est de diviser la région autour du puits en zones concentriques pour construire l'arbre avec un itinéraire presque optimal en allant des zones intérieures vers les zones externes. Il utilise un ordre postérieur de parcours avec un raccourcissement possible de l'itinéraire en arbre pour dériver un itinéraire pour chaque agent. Le principal inconvénient

de cet algorithme, est qu'il dérive des itinéraires statiques qui pourront être basés sur des anciennes observations de la topologie du réseau. L'agent sera alors incapable de compléter son itinéraire en cas de présence de nœuds défaillants.

CDDAM [45] est conçu pour l'agrégation et la collecte périodique des données dans un RCSF. Il emploie un arbre de parcours enraciné. Le nœud puits crée les agents mobiles, MAj pour chaque arbre enraciné Ti. Ensuite, le nœud puits expédie un agent à chaque racine de Ti pour des tâches d'agrégation et de collecte de données. L'agrégation et la collecte de données commence à partir de chaque nœud feuille de Ti.

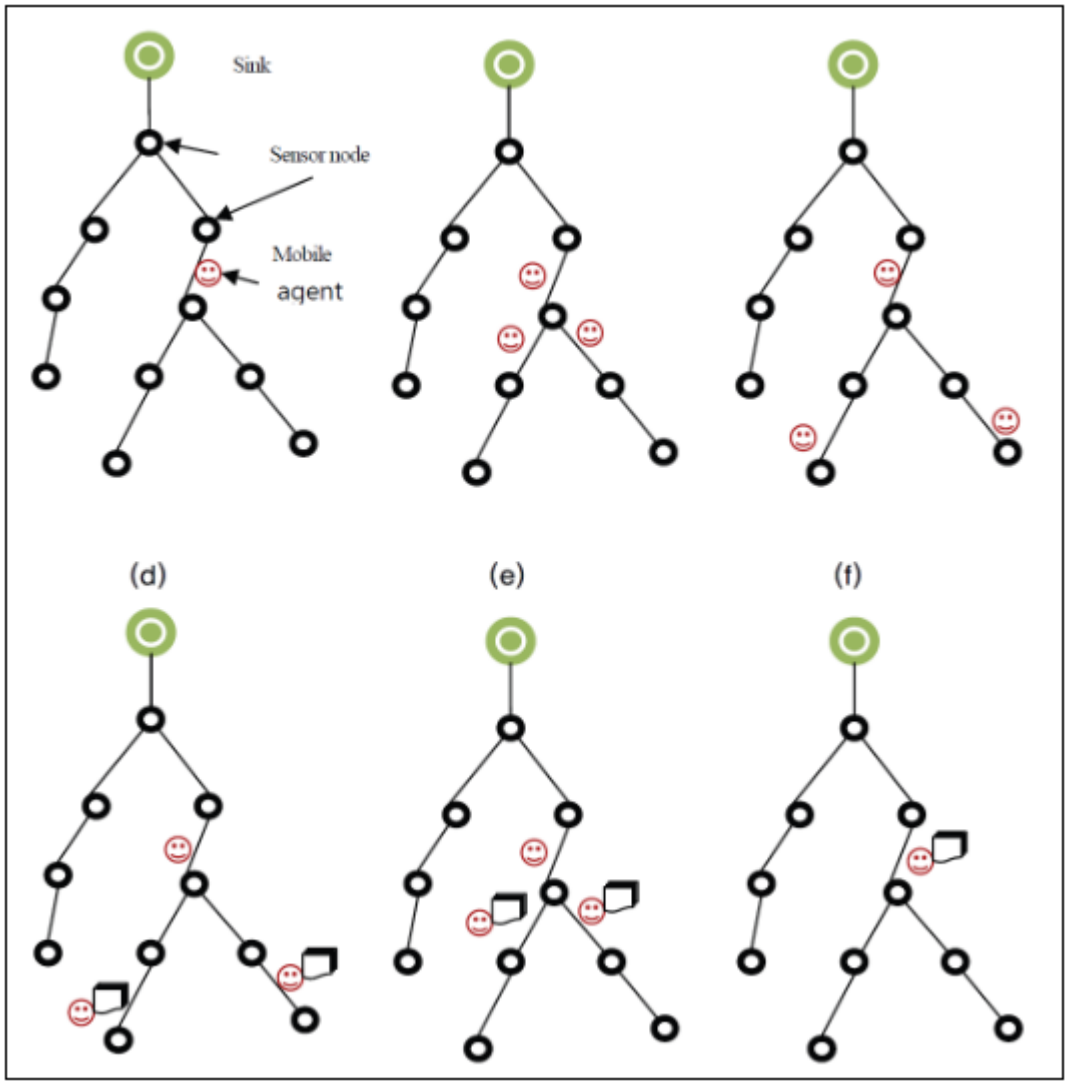


Figure III.3. Parcours des AMs dans CDDAM.

Chaque MAj commence sa migration à partir du nœud racine de l'arbre Ti. Lorsqu'il atteint un nœud SNi ayant deux nœuds enfants ou plus, il construit des copies de lui-même MAclonej, k , où k = 1, 2,..m et m est le nombre d'enfants du nœud visité SNi. Chaque agent clone MAclone j,k est expédié pour visiter le nœud enfant de la branche d'arbre. Cependant MAj reste sur le nœud parent SNi. Lorsque l'agent MAclone j,k atteint un nœud feuille, il commence à accomplir la tâche d'agrégation de données et retourne de nouveau au nœud parent SNi où toutes les données rassemblées par MAclone j,k sont remis à MAj. Ensuite, l'AMclone j,k s'autodétruit.

MST-MIP est employé dans [46] pour créer des groupes de nœuds et trouver la planification de l'itinéraire optimale des agents multiples. Tous les nœuds sources du réseau sont modélisés par un graphe totalement connecté (TSG-Totally Connected Graph) ; les sommets sont représentés par les nœuds sources et le poids de chaque chemin est égal au nombre de sauts entre les deux nœuds. Les nœuds sources dans la branche de l'arbre sont groupés et l'AM est expédié pour se déplacer entre eux avant de renvoyer les données recueillies au puits. Le nombre d'AMs dans le réseau est déterminé par le nombre de sommets directs reliés au puits. Plutard, Une version équilibrée de MST-MIP appelé BST-MIP (balanced spanning tree for multi-agent itinerary planning) est proposée. Cette dernière emploie un facteur d'équilibrage (alpha) afin de réaliser un compromis entre la consommation d'énergie et la latence.

Bien que les algorithmes MIP à base d'arbre opèrent mieux que les algorithmes SIP dans les réseaux à grande échelle, l'itinéraire de l'AM consomme de l'énergie supplémentaire due aux itinéraires renversés que l'AM prend, en particulier quand il y a une quantité énorme de branches.

III.3.2. Protocoles considérant la taille des données à collecter

Dans la plupart des algorithmes MIP, l'information géographique des nœuds capteurs est le paramètre principal employé pour déterminer le nombre optimal d'AMs et de leurs itinéraires. [47] a proposé GIGM-MIP (a greatest information in the greater memory-based MIP) qui considère non seulement l'information géographique, mais tient compte également de la taille des données dans chaque partition pour déduire le nombre optimal d'AMs et de leurs itinéraires. Dans GIGM-MIP, l'algorithme de partitionnement en k-moyennes est employé pour diviser le réseau en k clusters (Étant donnés des points et un entier k, le problème est de diviser les points en k groupes, appelés clusters, de façon à minimiser une certaine fonction (on considère la distance d'un point à la moyenne des points de son cluster ; la fonction à minimiser est la somme des carrés de ces distances). Après la division du réseau, GIGM-MIP calcule la taille des données des nœuds sources dans chaque cluster. Cette taille de données déterminera alors le nombre d'AMs qui serait assigné à cette partition tels que chaque partition peut avoir plus d'un AM. Cependant, l'algorithme GIGM-MIP a deux inconvénients principaux. Le premier est qu'avec l'algorithme de division des k-moyennes, le nombre de partitions doit être manuellement identifié par l'utilisateur. La deuxième limitation de l'algorithme GIGM-MIP est que chaque partition peut avoir plus d'un AM. La taille de données des nœuds sources dans chaque partition déterminera le nombre d'AMs qui sera expédié à cette partition. Bien que cette solution ait équilibré les données accumulées entre plusieurs AMs, l'utilisation de plus d'un AM dans une partition a comme conséquence une augmentation du nombre de sauts de migration des AMs due au plus grand nombre d'itinéraires dans chaque partition. D'ailleurs, tous les AMs utilisés dans une partition particulière doivent transporter le code d'agrégation (qui est identique pour tous les AMs) aux nœuds source afin d'exécuter la tâche d'agrégation. Multiple AMs portant le même code d'agrégation dans une seule partition aurait comme conséquence une augmentation de consommation d'énergie [33].

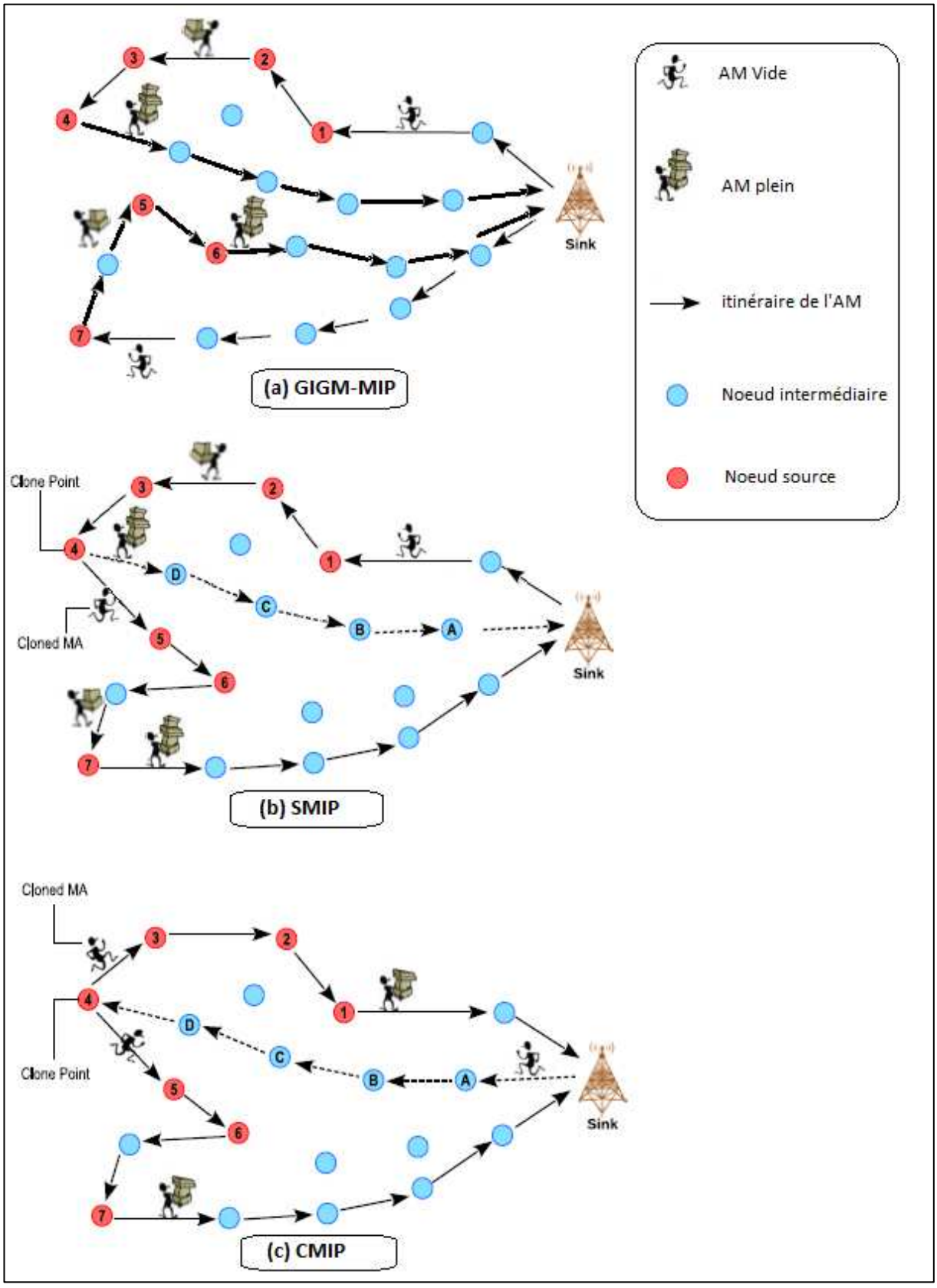


Figure III.4 Parcours des AMs dans GAGM-MIP, SMIP et CMIP.

L'approche SMIP [33] propose deux types d'AMs (un AM principal et un AM esclave (SMA)). Le puits devrait déterminer l'itinéraire pour chaque AM. Au début, après que le processus de partitionnement du réseau soit fait, le puits assigne un AM principal à chaque partition. Dans chaque partition, la séquence de visite des nœuds sources pour l'AM principal est statiquement déterminée au niveau du puits en adaptant l'algorithme LCF. Après avoir établi la liste de visite des nœuds sources, le puits détermine dans quel nœud source l'AM principal créera un agent mobile esclave (SMA). Dans SMIP, la taille de la charge utile des données de l'AM principal est employée pour déterminer à quel nœud source le SMA peut être créé. Spécifiquement, un seuil a été employé pour la charge utile des données de l'AM principal, ainsi une fois que les données accumulées excèdent le seuil, l'AM principal créera un nouveau SMA. Le but principal de l'approche SMIP est d'alléger le problème de la charge de travail dû au transport du même code de traitement par multiples AMs [33].

Une amélioration de SMIP est proposée dans l'algorithme CMIP (clone mobile-agent itinerary planning approach) [73], qui a pour but de minimiser la durée de la tâche, tout en collectant le maximum des données en employant aussi le concept de clonage de l'AM.

Dans CMIP, un seul AM est expédié par le puits vers la zone cible, cet AM va se cloner au niveau d'un certain nœud appelé « clone point » qui est considéré comme le nœud source le plus loin du puits. A partir de ce nœud, les nœuds sources à visiter seront donc divisés en deux groupes. Arrivé au point de clonage, l'AM va créer une copie de lui et l'envoyer pour collecter les données du premier groupe de nœuds sources, en même temps que lui-même se chargera de la collecte des données du 2eme groupe.

Le choix du point de clonage comme étant le nœud source le plus loin du puits, est justifié par le fait que le processus de collecte à partir du nœud le plus loin en allant vers le puits consomme moins d'énergie et de temps comparé au chemin allant du plus proche nœud source. L'itinéraire suivi par l'AM au premier groupe sera le chemin obtenu par l'algorithme LCF, alors que pour le deuxième groupe, l'itinéraire de l'AM sera le chemin inverse obtenu par LCF. En effet, cela permettrai d'avoir le nœud source le plus proche du puits comme étant le dernier nœud à visiter pour les deux AMs. Dans CMIP, l'emplacement du point de clonage est un paramètre clé du protocole et la méthode de son élection joue un rôle très important sur les performances du réseau. Cependant, définir ce point de clonage comme étant le nœud source le plus loin du puits n'est pas toujours la meilleure solution et peut parfois mener à de mauvais résultats.

III.3.3. Protocole à base d'algorithme génétique

Un algorithme génétique GA-MIP a été proposé dans [48] afin de déterminer également le nombre optimal d'AMs dans les MIP. Le but est de traiter le MIP comme problème simple.

GA-MIP est employé pour déterminer le nombre d'AMs ainsi que leurs nœuds sources attribuées par la méthode à deux niveaux de codage. Le codage représente un gène qui contient le source-ordering-code (vecteur d'ordre) et le source-grouping-code (vecteur de groupe). Un source-ordering-code est un vecteur qui inclut des segments, où chaque segment a un certain nombre de nœuds sources à visiter par un AM particulier. Le source-grouping-code est un vecteur de nombres, dont chaque nombre indiquant le nombre de nœuds source

de chaque segment dans le source-ordering-code. Les deux opérations de base de l'algorithme génétique (croisement et mutation) sont employées dans chaque itération, puis une fonction physique est adaptée pour choisir les meilleurs gènes à survivre. GA-MIP a une meilleure exécution que d'autres algorithmes MIP précédents en termes de délai et consommation d'énergie, mais c'est une approche avide qui produit une solution essentiellement suboptimale et une complexité de calcul élevée.

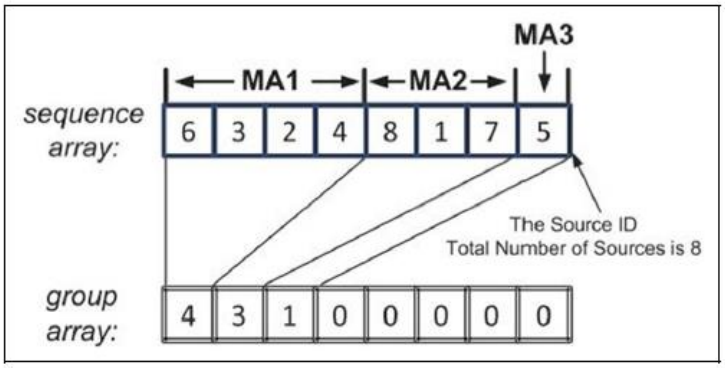


Figure III.5 Exemple de l'algorithme GA-MIP.

III.3.4. Protocoles à base de point de visite central

CL-MIP [49] est un protocole qui détermine le nombre optimal d'AMs en MIP en utilisant l'algorithme VCL. L'idée principale de l'algorithme VCL est de calculer la densité des nœuds source par la distribution d'un facteur d'impact parmi les nœuds sources. Ainsi, si n représente le nombre de nœuds sources, alors chaque nœud source recevra (n-1) facteurs d'impact de la part des autres, et un de lui-même. Puis, l'emplacement du nœud source avec le plus grand facteur d'impact accumulé sera choisi comme VCL. Après le choix du VCL, tous les nœuds sources dans le rayon de VCL sont groupés ensemble et assignés à un AM. En employant l'algorithme itératif, le processus ci-dessus est répété jusqu'à ce que tous les nœuds sources restants soient groupés et assignés aux AMs. Enfin, chaque itinéraire est déterminé en employant un des algorithmes SIP (GCF, LCF, IEMF et IEMA). Néanmoins, l'algorithme CL-MIP assume une approche à base de clusters, où les nœuds source sont arrangés géographiquement et distribués dans plusieurs clusters. Ceci limite l'utilisation de l'algorithme lorsque les nœuds capteurs sont déployés à faible densité. D'ailleurs, l'algorithme CL-MIP peut avoir comme conséquence l'isolement de nœuds source (peu de nœuds source sont localisés ensemble) et puis assigné à un nouvel itinéraire. Ceci mènerait à une augmentation de consommation d'énergie, particulièrement quand les nœuds sources isolés sont situés loin du puits.

OMIP (Optimal Multi-Agents Itinerary Planning) [76] est une autre solution similaire à CL-MIP, où les nœuds sources sont regroupés en clusters. OMIP adopte le protocole ECRP [56] (Efficient Clustering Routing Protocol) pour partitionner le réseau en clusters, sélectionner un nœud médoïde MNi dans chaque cluster, puis minimiser la distance moyenne entre MNi et les autres nœuds du cluster. L'approche OMIP souffre des mêmes inconvénients que CL-MIP, où la méthode de clustering dépend fortement de la distribution des nœuds sources. OMIP génère des clusters avec peu de nœuds sources à visiter par l'AM. En

conséquence, les itinéraires des AMs provoquent beaucoup l'overhead, ce qui affecte directement les performances du réseau en termes de consommation d'énergie et de latence.

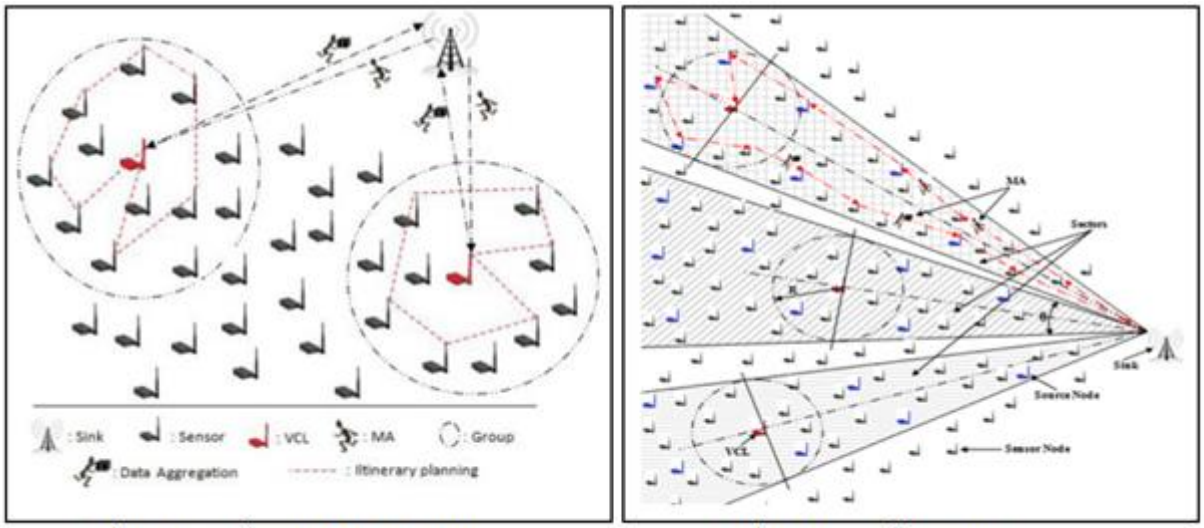


Figure III.6 Architecture de CL-MIP, OMIP et AG-MIP.

Dans [51], une nouvelle proposition AG-MIP (Angle Gap - MIP) offre une nouvelle vision de regroupement des nœuds sources qui n'utilise pas la forme du cercle. Dans AG-MIP les nœuds au sein d'un écart d'angle particulier Δθ autour d'un point central de visite (VCL), doivent être inclus dans le même groupe. L'idée principale d'AG-MIP est de connecter la station de base et tous les nœuds sources avec des lignes droites. Donc les écarts d'angle Δθ entre les lignes deviennent un facteur critique pour décrire le degré pertinent entre les nœuds sources. L'importance d'AG-MIP est son mode de groupement; il utilise l'écart d'angle pour diviser le réseau en secteurs. Par conséquence, il conduit à une contestation et des interférences potentiellement réduites entre les agents mobiles, cependant l'interrogation ouverte dans cette approche est de trouver un seuil optimal de l'écart d'angle.

MAEF (Multi-mobile Agent itinerary planning-based Energy and Fault aware) [77] est une solution qui combine entre les VCLs et les arbres. En effet, MAEF regroupe également les nœuds sources en clusters et chaque cluster a un cluster-head (CH). Il adopte la même idée présentée dans CL-MIP dans laquelle une distribution du facteur d'impact de densité est utilisée pour sélectionner les CH. Ensuite, tous les nœuds sources, qui se trouvent dans la plage de transmission maximale de chaque CH sont regroupés. MAEF diffère de CL-MIP et d'OMIP dans les itinéraires des AMs qui ne sont établis que parmi les CHs et déterminés à l'aide du protocole MST [46]. De plus, le nombre d'itinéraires est égal au nombre de nœuds situés dans la plage de transmission du puits. Dans MAEF, une fois que l'AM atteint le CH pour la première fois, il notifie aux nœuds source du CH d'envoyer leurs données au CH. Lorsque l'AM atteint le dernier CH de son itinéraire, il commence à collecter des données auprès des CH sur son chemin de retour vers le puits. Bien que l'algorithme MAEF déploie une nouvelle stratégie MIP, où les AMs distribués ne visitent que les CH, il présente deux inconvénients principaux. Premièrement, la plage de transmission des CH (qui regroupait auparavant les nœuds sources en groupes) peut générer un chevauchement entre les groupes, ce qui peut augmenter la surcharge. Deuxièmement, chaque AM doit parcourir son itinéraire

deux fois; pour informer les nœuds sources du CH d'envoyer des données et de recueillir des données du CH. Cela entraîne une augmentation de la consommation d'énergie et de la durée des tâches.

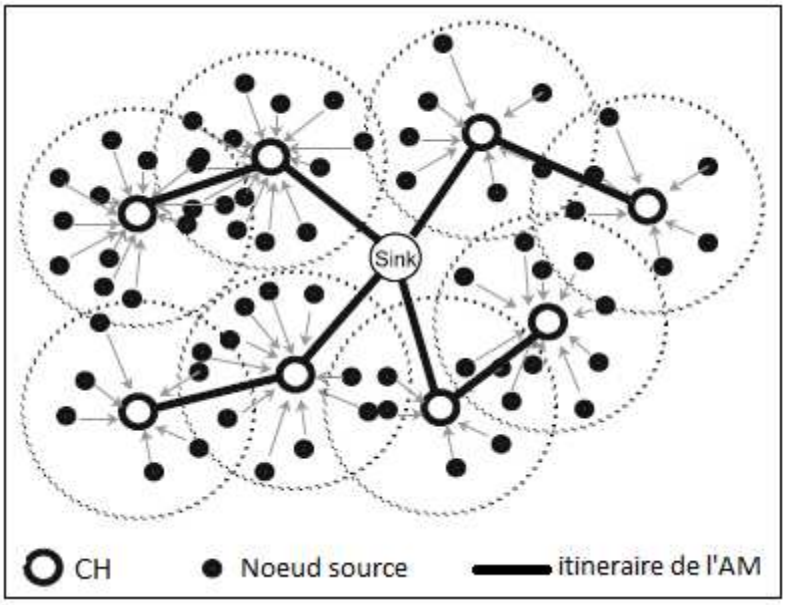


Figure III.7 Itinéraires des AMs dans MAEF [77].

III.3.5. Protocoles se basant sur des secteurs équiangles

Multiple mobile agents with dynamic itinerary based data dissemination [52] (MMADIDD) est un protocole de dissémination de données initialement décrit dans EBMADD [53]. Il considère un réseau avec plusieurs nœuds capteurs uniformément distribués dans une zone de surveillance en forme de cercle ou le puits est situé au centre de cette zone.

MMADIDD divise la zone de surveillance en 8 secteurs équiangles (45°) et en plusieurs anneaux de largeur rmax/2 (voir figure III.8). Chaque Secteur est attribué à un AM qui déterminera son itinéraire dynamiquement à chaque saut en utilisant des informations enregistrées localement.

Le puits crée et diffuse les agents mobiles au premier anneau de chaque secteur. Chaque AM traverse initialement les nœuds de bordure d'un secteur en allant de l'anneau intérieure vers l'anneau extérieure. Dans ce trajet, du puits vers la couronne extérieure, l'AM ne collecte pas les données.

Lors de l'arrivé à la dernière couronne, l'AM commence à colleter et agréger les données a fur et à mesure qu'il migre dynamiquement entre les nœuds sources dans un ordre en spirale à l'intérieur du secteur jusqu'à ce qu'il arrive au puits comme le montre la figure III.7. L'avantage de cette approche est d'éviter d'entraîner inutilement les données lors de la traversée initiale du puits vers l'anneau extérieure, ce qui permet de réduire l'overhead de communication.

Contrairement aux protocoles MIP vu précédemment, l'AM de MMADIDD utilise un mode de routage dynamique multi-sauts à moindre coût en migrant vers le nœud le plus proche ayant une grande énergie, ce qui le rend tolérant aux pannes

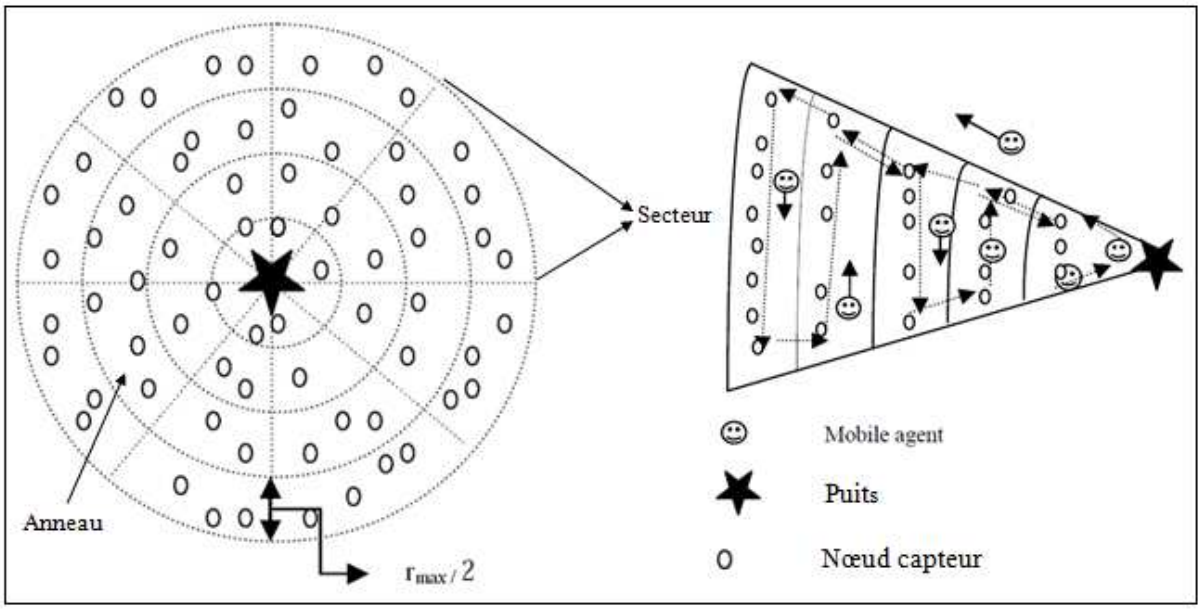


Figure III.8 Architecture de MMADIDD et IEMADI.

MMADIDD suppose que le puits est équipé d'une antenne directionnelle, qui peut atteindre tout le réseau afin de l'utiliser pour diviser la zone d'intérêt en 8 secteurs et plusieurs anneaux. De ce fait, la phase de groupement des nœuds engendre un grand overhead puisque le puits rediffuse le message de construction pour chaque anneau y compris ceux déjà construits. Aussi, la puissance d'émission du puits devrait atteindre tous les nœuds en un seul saut, ce qui n'est pas toujours évident notamment lors du passage à l'échelle.

Une amélioration de MMADIDD nommée IEMADI [54] a été proposée plutard. IEMADI utilise une méthode différente de MMADIDD pour le groupement des nœuds. En effet, IEMADI utilise une formule mathématique localement au niveau de chaque nœuds capteurs pour transformer les coordonnées cartésienne (x,y) en coordonnées polaire (r, θ) sans avoir recours à échanger des messages avec le puits.

II.4. Comparaison des performances des protocoles MIP

Afin de comparer les performances des protocoles MIP étudiés dans ce chapitre, nous avons établis le tableau suivant, qui regroupe les protocoles MIP étudiés sous différentes classes partageant les mêmes caractéristiques :

Classe	Protocole	Critère de détermination du nombre de partitions	Paramètre de partitionnement	Nombre d'AMs déterminé selon:	Planification d'itinéraire	Avantages	Inconvénients
Structure arborescente	NOID, ILS SNOID TBID CBID CDDAM	Nombre de nœuds dans la portée du puits	Portée des nœuds et /ou cout de communication	Nombre de partitions	statique	Minimise le coût d'agrégation	Consomme de l'énergie supplémentaire due aux itinéraires renversés
Taille de données	GIGM-MIP SMIP	Un nombre initial	Distance entre les nœuds	Taille des sonnées dans chaque partition	statique	Equilibre la charge des AMs Diminue le risque de perte des données.	Détermination du seuil de la charge des données
Clonage	CMIP	Un nombre initial	Distance entre les nœuds	Nombre de partitions X 2	statique	Améliore la latence	Deux AMs dans chaque partition pourrait être insuffisant pour de grandes partitions ou excessif pour les petites.
Algorithme génétique	GA-MIP	Un nombre initial	Distance entre les nœuds	Nombre de partitions	statique	Bons résultats de latence	Solution avide
Point de visite central	CL-MIP AG-MIP OMIP MAEF	Nombre de VCLs	Densité des nœuds	Nombre de partitions Nombre de nœuds dans la portée du puits pour MAEF	statique	Groupement de nœuds efficace	l'isolement de nœuds source à faible densité.
Secteurs équiangles	EBMADD MMADIDD IEMADI	Un nombre initial	position des nœuds	Nombre de partitions	dynamique	Tolérant aux pannes	Concentration de la charge au centre

Table III.1 Comparaison des classes de protocoles MIP

Toutes ces approches MIP ont pour principal but de définir des itinéraires optimaux pour de multiples AMs opérant en parallèle pour collecter et agréger les données d'un RCSF. Cependant, nous remarquons que la plupart de ces approches se partagent plusieurs points faibles communs, en effet :

- l'AM doit passer par tous les nœuds sources pour collecter leurs données. Ceci va retarder l'AM et provoquera donc une grande latence. Pour aller plus vite, il serait alors mieux de faire transiter l'AM par seulement une partie des nœuds sources.
- Aussi, chaque AM démarre du puits pour collecter les données et enfin revient aussi vers lui. Ceci engendre un grand overhead et une consommation supplémentaire en énergie, puisque chaque AM transporte le code de traitement avec lui. Pour diminuer cet overhead, il serait préférable que le puit envoi un seul AM principal qui sera ensuite éclaté en plusieurs AMs pour collecter les données en parallèle.
- Et enfin, presque tous ces protocoles utilisent des itinéraires statiques pour les acheminer les AMs, où c'est le nœud puits qui décide de la séquence de visite des AMs avant de les expédier. Ceci pourrait engendrer des défaillances du système lors de l'apparition de pannes dans l'itinéraire préétablis. La seule catégorie qui prend ce point en considération, et qui utilise donc des itinéraires dynamiques est la catégorie basée sur des secteurs équiangles. C'est pourquoi nous nous sommes basés dans notre contribution sur cette catégorie tout en essayant de régler les autres points faibles communs des approches MIP.

III.5. Conclusion

La performance des algorithmes conçus à base d'un seul agent mobile est relativement faible en particulier dans les réseaux de capteurs à grande échelle. En effet, un seul agent mobile est utilisé pour la tâche de fusion des données et par conséquent, il doit porter une lourde charge de données extraites par les nœuds sources. De ce fait, cette catégorie ne convient que pour les réseaux de capteurs à petite échelle.

D'autre part, la catégorie à base de multiples agents mobiles peut être efficacement utilisée dans les réseaux de capteurs à grande échelle. Dans ce cas, plusieurs agents mobiles travaillent simultanément pour effectuer la tâche d'agrégation et fusion de données qui mène à un gain considérable en temps et en énergie. Cependant, la complexité des algorithmes proposés dans cette catégorie (MIP) est considérable par rapport aux algorithmes à base d'un seul agent mobile (SIP). En effet, plusieurs problèmes restent difficiles à optimiser, comme la détermination du nombre optimal d'AMs et leur itinéraire. Effectivement, la plupart des solutions proposées supposent que l'itinéraire de chaque AM est déterminé au niveau du nœud puits, ce qui signifie que l'AM a un itinéraire statique. Dans ce cas, tout changement dans la topologie du réseau dû à la mobilité des nœuds ou à des pannes de nœuds (épuisement énergétique) pourrait affecter la migration de l'AM. La migration de l'AM doit donc être dynamique et plus intelligente, de sorte à ce que l'itinéraire de l'AM soit décidé au niveau de chaque nœud capteur visité.

Partie 2

Contribution

Chapitre IV

CHEESE: Un protocole coopératif et hybride

IV.1. Introduction

Dans les chapitres précédents, nous avons classifié les protocoles de dissémination des données dans les RCSFs en 3 catégories (C/S, single MA et multiples MAs). Il s'avère que le paradigme C/S soit trop gourmand en énergie malgré qu'il soit très efficace pour délivrer des données précises à temps. Alors que la solution à un seul AM prend beaucoup de temps pour délivrer les données avec un grand risque de perte des données. Les solutions à multiples AMs ont tendance à équilibrer entre la consommation énergétique et le délai de délivrance ainsi que la précision des données. Dans cet article, nous proposons une solution hybride (CHEESE [Cooperative and hybrid energy efficient scheme for wireless sensor networks]) [55] qui combine entre de multiples AMs en parallèle avec le paradigme C/S afin de bénéficier des avantages des deux paradigmes dans un seul algorithme et de répondre au mieux aux exigences de qualité de service des applications dans les RCSFs.

IV.2. Environnement et hypothèses

L'architecture de CHEESE s'intéresse aux applications lançant des requêtes pour la collecte cyclique des données dans une sous-région du réseau. Cette dernière est appelée zone cible. Ses coordonnées sont fixées par la requête du puits, et elle est supposée géographiquement éloignée du puits.

Exemple : *« Chaque I minutes pour les prochaines T heures, envoyez-moi une estimation de localisation de n'importe quel animal à deux pattes dans la sous-région R du champ de capture ».*

Notre réseau supporte une variété de types de tâches. C'est-à-dire que les nœuds sont équipés de différents types de dispositifs de capture. Par exemple : des capteurs de température, de mouvement, d'humidité…etc.

IV.2.1. Hypothèses principales

Le réseau utilisé dans notre protocole est un réseau qui répond aux hypothèses suivantes :

- Chaque nœud a un identificateur unique (ID).
- Tous les nœuds capteurs sont homogènes et ont donc la même capacité de communication, de traitement et d'énergie mais peuvent différer dans leurs types d'unités d'acquisition des données.
- Chaque nœud est statique et connaît ses coordonnées géographiques(x, y) dans un plan à deux dimensions (soit à l'aide d'un GPS ou autres algorithmes de localisation.
- Plusieurs tâches sont prises en considération par l'application. En effet, l'application envoie un seul paquet transportant plusieurs tâches. Puis, les

résultats sont fusionnés dans un seul paquet afin de diminuer le coût de communication.

- Les nœuds capteurs de la région cible sont géographiquement proches l'un à l'autre comparés à la distance les séparant du nœud puits.
- L'environnement est supposé sain ; c'est-à-dire qu'il n'y a pas lieu d'attaques malicieuses qui affecteront le déroulement du système conçu.

IV.2.2. Routage géographique

Un routage est dit géographique lorsque les décisions de routage sont basées sur les positions des nœuds [57]. Puisque notre solution utilise un GPS, ainsi les positions des nœuds sont disponibles et il est donc plus convenable et plus qualifié d'utiliser un routage géographique pour acheminer les données dans le réseau.

Dans un routage géographique, un nœud qui doit envoyer un paquet sélectionne comme prochain saut le nœud qui est « dans la direction » du destinataire et qui produit « le meilleur » avancement vers lui [58].

Le mécanisme de relais généralement suit le paradigme connu glouton (Greedy Forwarding) [59]: lorsqu'un nœud capteur a un paquet à transmettre, le nœud de relais le plus proche de la destination est choisi pour transférer le paquet. Ce principe glouton est appliqué jusqu'à ce que la destination finale soit atteinte.

Le protocole de routage sans fil utilise généralement un processus de découverte de voisinage qui est réalisé grâce à l'envoi périodique de messages Hello. Chacun de ces messages contient l'identifiant du nœud émetteur et éventuellement d'autres informations (position géographique, énergie restante, mémoire libre…). Cet échange périodique permet de construire et de maintenir une table de voisinage contenant la liste des nœuds à portée de communication.

IV.3. Principe de CHEESE

Notre contribution CHEESE se base sur une architecture divisée en des secteurs équiangles semblable à l'architecture d'IEMADI [54]. Sauf que dans CHEESE, nous utilisons cette architecture seulement dans la zone cible (voir figure IV.1). Nous utilisons un puits virtuel (PV) qui est un nœud choisi au centre de la zone cible afin de diminuer l'overhead de communication entre le puits et la zone cible. Le puits va donc envoyer un seul AM vers le puits virtuel, et ce dernier va se charger de dispatcher multiples AMs à travers toute la zone cible (un AM pour chaque secteur).

Ensuite, chaque AM s'occupera de collecter les données de son secteur qui est lui-même divisé en plusieurs anneaux centrés. La collecte se réalisera suivant un itinéraire en forme de spirale allant de l'anneau extérieur vers le centre ou se trouve le PV. Dans IEMADI, chaque AM devrait parcourir tous les anneaux pour effectuer la tache de collecte, alors que dans notre contribution CHEESE, l'AM parcourt seulement la

moitié des anneaux (un seul anneau est parcouru parmi chaque deux anneaux successifs - voir figure IV.4). L'autre moitié anneaux non visités par l'AM vont envoyer leurs données aux anneaux voisins appartenant au chemin de l'AM. Ceci va permettre d'améliorer le temps de réponse et de diminuer le risque de perte de l'AM puisque son chemin est rétréci. De plus, l'énergie consommée par l'AM sera minimisée puisqu'il ne transfère pas tout son paquet à travers tout le réseau.

Afin de diminuer la taille des paquets transmis par l'AM, et en plus des trois niveaux d'agrégations utilisées dans MADD [36] (agrégation d'application, agrégation spatiale et fusion), CHEESE propose un quatrième niveau d'agrégation correspondant à l'agrégation temporelle et qui sera effectuée grâce au PV. En effet, ce dernier va enregistrer une copie de chaque paquet de données cycliques envoyées au puits, pour pouvoir comparer ces données avec celles du cycle suivant et éliminer les redondances avant de les envoyer au puits (voir section IV.4.4).

IV.4. Fonctionnement de CHEESE

CHEESE est adapté aux applications de type hybride (query-driven et time-driven), où l'application demande au puits de collecter les données cycliquement durant une période de temps prédéfinie au niveau d'une zone géographique spécifique appelée zone cible (ZC).
Pour ce faire, CHEESE se base sur les quatre étapes suivantes :
- Phase 1 : Sélection des nœuds sources et du puits virtuel(PV) ;
- Phase 2 : Préparation des itinéraires des AMs ;
- Phase 3 : Collecte cyclique des données ;
- Phase 4 : Agrégation et fusion des données collectées.

IV.4.1. Etape 1 : Sélection des nœuds sources et du puits virtuel (PV)

IV.4.1.1. Sélection des nœuds sources

Dans cette étape, le puits diffuse un message d'intérêt qui spécifie les descriptions des données dont l'application a besoin afin d'identifier les nœuds sources d'événements. L'intérêt est ensuite propagé vers la zone cible en multi sauts à travers les capteurs du champ de captage via un routage géographique glouton.
Afin de permettre au puits de choisir un puits virtuel (PV) pour le remplacer dans la région cible, chaque nœud non source (les nœuds qui ne contiennent pas les données correspondantes à l'intérêt du puits) de la région cible qui recevra l'intérêt vérifie si son taux d'énergie atteint un seuil ***PV_Seuil***. Si c'est le cas, il se déclare comme candidat pour devenir, éventuellement, un PV et rajoute alors, dans la liste ***Liste_NSources***, ses informations (son identifiant, ses coordonnées cartésiennes et son taux d'énergie). Ensuite, il continue la diffusion de l'intérêt à ses voisins directs dans la zone cible. La liste ***Liste_NSources*** contient les nœuds intermédiaires non sources dans les zone cible.

A la réception d'un message d'intérêt par un nœud source qui appartient à la zone cible, ce dernier envoie ses données d'exploration (ses coordonnées cartésiennes et ***Liste_NSources*** vers le puits et continue la diffusion du message d'intérêt dans la zone cible.

A la réception des données exploratoires, le puits crée une liste ***Liste_Sources*** qui contient les coordonnées cartésiennes des nœuds sources à visiter plus tard par les AMs.

IV.4.1.2. Sélection du Puits Virtuel (PV)

Le PV est un nœud appartenant à la région cible qui est choisi par le puits pour le remplacer dans la région cible afin de minimiser le nombre de transferts des AMs de la région cible vers le puits.
Vu que le PV (qui est un nœud ordinaire dans le réseau) devrait prendre le rôle du puits (qui a généralement plus de puissance matérielle que les autres nœuds du réseau) pour accomplir sa tâche, il devrait être choisi sur des critères spécifiques comme suit:

- Le PV doit avoir assez d'énergie pour pouvoir jouer le rôle du puits et recevoir donc toutes les données demandées par l'application et les traiter localement pour diminuer la taille de la donnée utile et minimiser donc le coût de transfert vers le puits.
- Le PV ne devrait pas faire partie des nœuds sources : vu que les nœuds sources de la région cible vont participer à la tâche de collecte des données par les AMs, ils ne devraient donc pas être surchargés par les autres tâches du puits surtout que leur énergie est préservée pour assurer la tâche de collecte périodique des données.
- Le PV doit être assez proche du centre de la zone cible pour pouvoir atteindre à faible coût tous les nœuds sources de la zone cible.

Le puits désigne alors un PV par des calculs mathématiques en utilisant les coordonnées cartésiennes envoyées par les nœuds de la région cible. Le puits calcule les coordonnées du point C qui représentera le centre de la zone cible et définie un disque D de rayon ***r*** et de centre C qui est une région limite pour le choix du PV.

Afin de choisir le nœud qui va jouer le rôle d'un PV dans la zone cible, le puits classe les nœuds de ***Liste_NSources*** appartenant au disque D selon deux facteurs séparément comme suit :

- Il classe les nœuds dans un ordre croissant selon la distance entre les nœuds et le point C ;
- Il classe les nœuds par ordre décroissant selon leur quantité d'énergie restante.

Soit Di et Ei le classement du nœud i selon les facteurs distance et énergie respectivement. Un seul nœud de la liste *Liste_NSources* est choisi comme un PV tel que :

$$\alpha D_{pv} + \beta E_{pv} = Min\ (\alpha D_i + \beta E_i)\ \forall i\ \in Liste\_NSources$$

Où α et β sont des facteurs d'importance de la distance et de l'énergie respectivement.

Le puits envoie un message contenant le code de l'AM et le rayon R (R est le rayon de la zone cible définie par le puits) au nœud choisit comme PV via un routage géographique glouton.

IV.4.1.3. Réélection du PV

A un moment donné pendant le déroulement de la tâche de collecte des données par les AMs, le PV peut être épuisé en énergie et cela va immobiliser l'application et mettre fin à la tache bien avant l'écoulement de son délai. Pour éviter cela, à chaque fin de cycle, le PV va comparer son énergie restante avec un seuil énergétique ***PV_Seuil***. Si le PV atteint ce seuil, il va faire une réélection d'un autre nœud qui sera élu comme un nouveau PV, en choisissant un voisin qui a le maximum d'énergie pour le remplacer. Puis le PV informe tous ses voisins qui appartiennent au premier anneau du nouveau PV élu et transmet les données de l'application (le code d'agrégation temporelle et une copie des données du dernier cycle) au nouveau PV.

IV.4.2. Etape 2 : Préparation des itinéraires des AMs

CHEESE utilise une solution semblable à IEMADI [54] pour créer les itinéraires des AMs en divisant la zone cible en plusieurs secteurs et anneaux.

IV.4.2.1. La division de la zone cible en anneaux et secteurs

Dans cette étape la zone cible sera divisée en **k** secteurs et en plusieurs anneaux Ai (i=1, …, n) comme le montre la figure IV.1 tel que l'angle de chaque secteur est égale à α (α=360/k). La largeur de chaque anneau est rmax/2 pour que chaque nœud puisse communiquer avec les nœuds qui se trouvent dans l'anneau qui le précède Ai-1 et l'anneau qui le suit Ai+1 quel que soit son emplacement (rmax est la portée de transmission maximale d'un nœud capteur). Ainsi le nombre total d'anneaux (maxAnneau) est égale à *R/(rmax/2).*
A la réception du code de l'AM, le PV diffuse un paquet Div_pkt qui contient deux champs : R et PVcoordCar où R est le rayon de la zone cible circulaire et PVcoordCar représente les coordonnées cartésiennes du PV. Chaque nœud qui reçoit ce paquet calcule la distance qui le sépare du PV pour que ce paquet ne se propage pas au-delà de la zone cible. En plus, chaque nœud calcule ses coordonnées polaires (r et thêta) comme suit :

Conversion du système cartésien vers le système polaire

Les deux coordonnées cartésiennes *x* et *y* peuvent être converties en coordonnées polaires *r* et thêta (θ) en utilisant les formules suivantes [60] :

- $r = \sqrt{x^2 + y^2}$

- pour obtenir une unique valeur de θ, nous nous restreignons à l'intervalle [0, 2π[:

- $$\theta = \begin{cases} \arctan\left(\frac{y}{x}\right) & si\ x > 0\ et\ y \geq 0 \\ \arctan\left(\frac{y}{x}\right) + 2\pi & si\ x > 0\ et\ y < 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & si\ x < 0 \\ \frac{\pi}{2} & si\ x = 0\ et\ y > 0 \\ \frac{3\pi}{2} & si\ x = 0\ et\ y < 0 \end{cases}$$

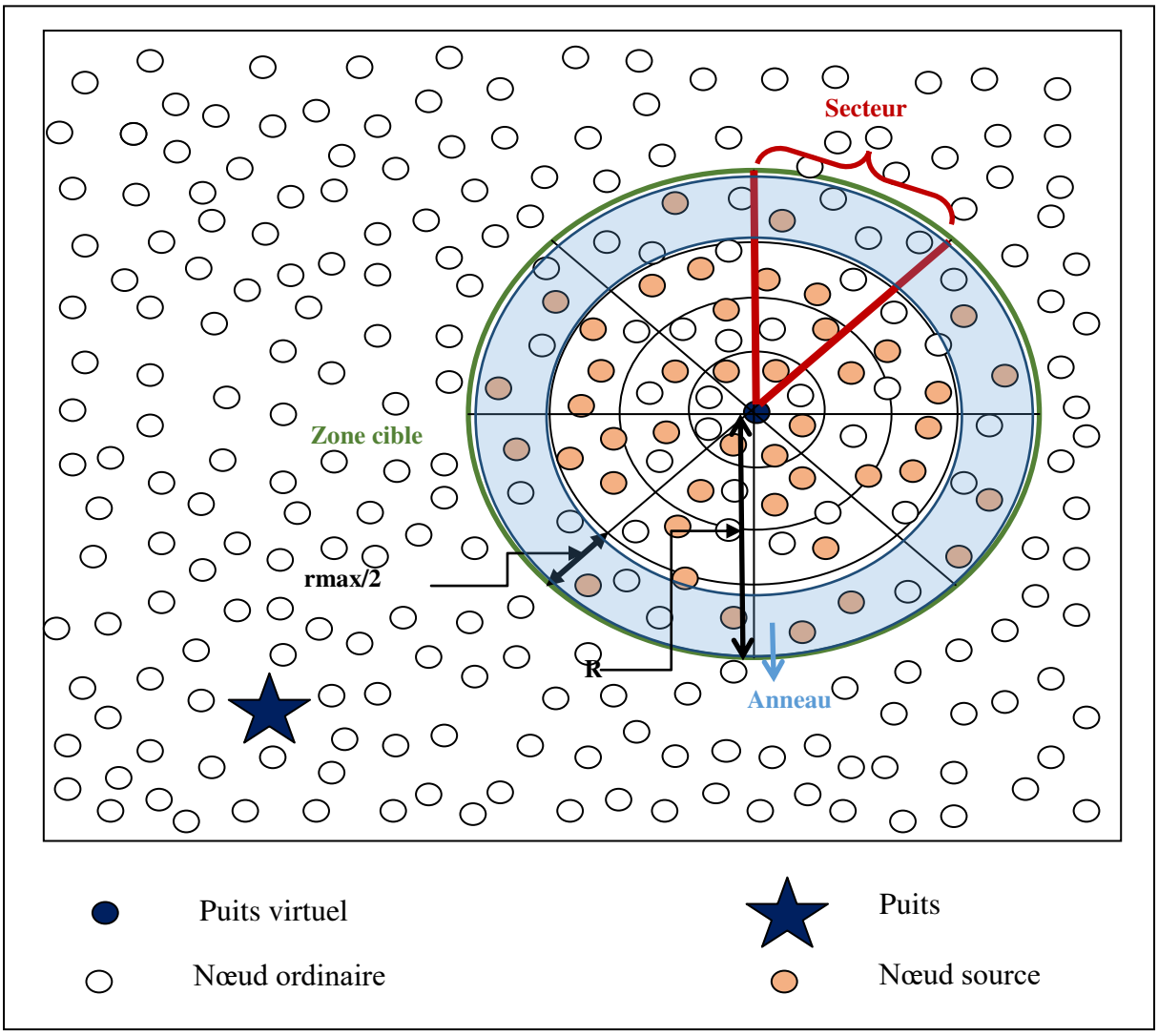


Figure IV.1. Architecture de CHEESE

Ensuite, chaque nœud appartenant à la zone cible calcule son numéro d'anneau (NumAnneau) et son numéro de secteur (NumSect) suivant l'algorithme ci-dessous.

Pour le PV :
Le PV crée Div_Pkt ;
Diffuser le paquet Div_pkt (double R, CoordCar PVcoordCar) ;

Pour les autres nœuds :

A la réception d'un paquet Div_pkt (double R, CoordCar PVcoordCar){
Si (distance (nœud.coordCar, PVcoordCar) <R) {
// Le nœud appartient à la zone cible
(nœud.r, nœud.thêta) =calculeCoordPolaire (PVcoordCar, nœud.coordCar) ;
nœud. NumAnneau =0; nœud. NumSect =0;i=1
maxAnneau = R/(rmax/2)+1;
si (nœud.NumAnneau == 0 &&nœud.NumSect == 0) {
nœud.NumSect = nœud.thêta/ α
tantque (i <=maxAnneau) {
si ($(i-1).\frac{r_{max}}{2} < n\text{œ}ud.r \leq (i).\frac{r_{max}}{2}$)){
nœud. NumAnneau = i;
break;
}
i++;
}
}
Diffuser le paquet Div_pkt (double R, CoordCar PVcoordCar) ;
}
}

IV.4.2.2. Construction de la table de voisinage

Chaque nœud de la zone cible crée et diffuse un paquet *Pkt_Voisin* dans le but de construire la table de voisinage. Le paquet *Pkt_Voisin* contient six champs (voir figure IV.2).

ID_nœud	nœud_E	NumAnneau	NumSect	CoordPolaire	SN

Figure IV.2 : Structure de la table de voisinage

Tels que :

- ID_nœud est l'identificateur pour le nœud,
- nœud_E est l'énergie restante du nœud,
- NumAnneau est l'identificateur de l'anneau auquel le nœud appartient,
- NumSect est l'identificateur du secteur auquel appartient le nœud,
- CoordPolaire représente les coordonnées polaires du nœud et
- SN indique si c'est un nœud source ou pas.

Si un nœud x reçoit un paquet *Pkt_Voisin* de son propre secteur, il met à jour sa table de voisinage Vx. La structure de la table de voisinage est représentée sur la figure IV.2.

À la fin de cette étape, chaque nœud connaît son NumAnneau et son NumSect ainsi que tous ses voisins dans sa région de transmission (à un saut) appartenant à son secteur.

IV.4.3. Etape 3 : Collecte cyclique des données

Le PV crée k messages STARTi(i= 1,…, k) (k est le nombre de secteurs) identiques et les expédie, chacun au voisin direct du premier anneau de chaque secteur qui a une valeur de thêta minimale (voir figure IV.3).

Le message START comporte deux paquets : un paquet AM_pkt (paquet Agent mobile) et un autre C/S_pkt (paquet Client/serveur).

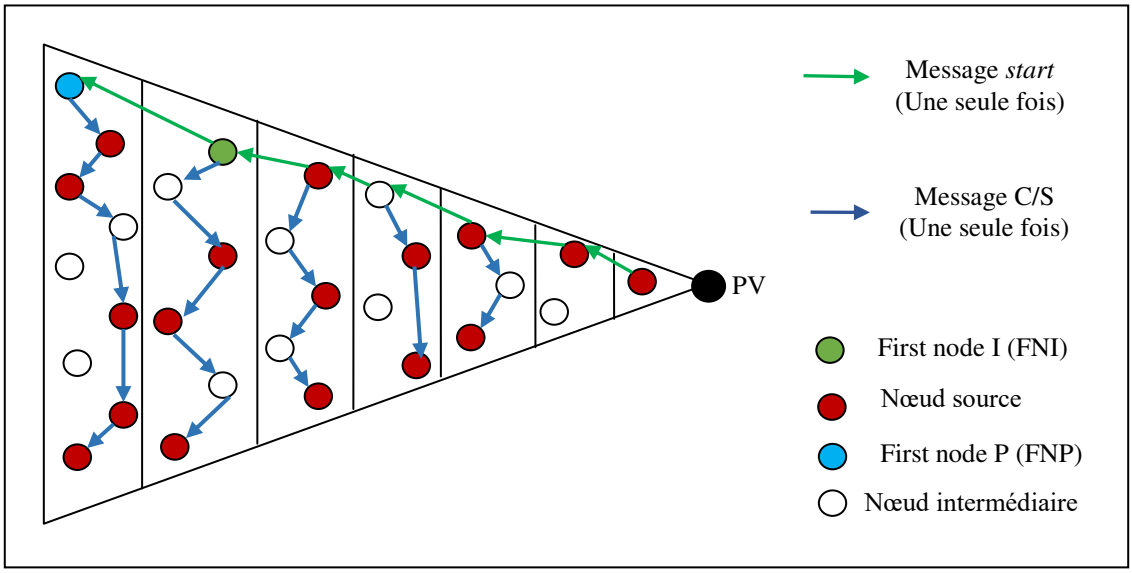


Figure IV.3 : Dissémination du message *start.*

IV.4.3.1. Phase Client / Serveur

Après avoir envoyé le message *Start* par le PV, chaque nœud qui le reçoit, sélectionne dans sa table de voisinage un voisin qui appartient au prochain anneau extérieur avec la valeur thêta la plus minimale.

Chaque nœud recevant le message *Start* va extraire le paquet C/S et va l'envoyer à tous les nœuds sources de son anneau en multi sauts en utilisant les tables de voisinage déjà construites en phase 2. En effet, chaque nœud recevant le paquet C/S vérifie dans sa table de voisinage s'il existe un nœud source qui appartient au même secteur et au même anneau avec thêta minimale. Si un tel nœud source n'existe pas, il choisit un nœud voisin (non source) le plus loin parmi ses voisins, et qui va jouer le rôle d'un nœud intermédiaire pour router le paquet C/S (voir figure IV.3).

Le paquet C/S contient l'ordre de générer des données périodiques (chaque *T_Delai*) et de les envoyer chaque *2xT_Delai* (pour alterner entre les anneaux pairs et impairs afin d'équilibrer la charge dans le réseau). Seulement, la moitié des données (soient

celles appartenant aux anneaux pairs ou celles appartenant aux anneaux impairs à tour de rôle pour chaque cycle) seront envoyées individuellement vers le plus proche nœud source de l'anneau voisin intérieur. Ceci va permettre de réduire la charge de la phase de l'AM pour gagner en énergie (puisque les données C/S seront directement transmises à l'anneau intérieur sans parcourir leur anneau courant) et diminuer donc le temps de parcours de l'AM.

En plus de transférer le paquet C/S, les nœuds sources copient leur code de traitement dans leur mémoire interne. Ainsi les AMs ne porteront plus le code de traitement à chaque cycle, puisqu'il existe déjà dans la mémoire de chaque nœud source. Le code de traitement est supprimé par les AMs lors du dernier cycle. Cela permet de diminuer la taille des AMs et de gagner donc en énergie et en temps de transfert de l'AM.

Cette phase se termine lorsqu'un nœud du dernier anneau reçoit le paquet *Start* et lance la phase de l'AM.

IV.4.3.2. Phase Agent Mobile

A la réception du message *Start* par le premier nœud de chaque dernier anneau, ce nœud va se nommer comme *FNP (First Node Pair)* et va extraire le paquet de l'AM *AM_pkt* contenant sept champs :

- ***AM_ID* :** est l'identifiant de l'AM.
- ***Direc_Flag* :** est une variable de drapeau, qui indique la direction de migration de l'agent mobile au sein de chaque anneau. Si la valeur de la variable est **true** alors l'AM se déplace de gauche à droite, sinon c'est de droite à gauche.
- ***Code_trait* :** le code de traitement (agrégation) de l'AM.
- ***Données* :** les données utiles collectées par l'AM.
- ***Premier_Cycle :*** Il indique qu'il s'agit du premier cycle.
- ***Nbr_Cycle :*** Il est initialisé au nombre maximum de cycles et il est décrémenté à chaque nouveau cycle.
- ***T_Delai :*** c'est la durée de la période de collecte cyclique des données.

Afin d'équilibrer la charge dans le réseau, nous échangeons les taches entre les anneaux pairs et impairs pour chaque cycle. Pour cela, nous allons avoir besoin d'un autre FN nommé *FNI (FirstNode Impair)* qui est le premier nœud de l'avant-dernier anneau. Le FNP va fixer la valeur de T_Delai à 2xT_Delai, et envoyer le MA_pkt au FNI (le nœud duquel il a reçu le message *Start*) après l'écoulement de T_Delai, dans le but d'alterner la collecte des données entres les anneaux pairs et impairs.

Le *FN* (qu'il soit FNP ou FNI) va garder une copie du MA_pkt qui sera stockée dans sa mémoire locale qui va servir pour le lancement des prochains cycles de l'AM.

Le *FN* (et chaque nœud qui recevra le MA_pkt) vérifie dans sa table de voisinage s'il existe un nœud source qui appartient au même secteur et au même anneau et répond aux exigences du seuil d'énergie, ce nœud doit avoir thêta minimale ou

maximale selon la valeur de **direc_Flag**. Si un tel nœud source n'existe pas, il choisit un nœud voisin (non source, pour jouer le rôle de routeur) le plus loin parmi ses voisins, qui répond aux exigences du seuil d'énergie et qui va jouer le rôle d'un nœud intermédiaire. La collecte et l'agrégation des données commencent alors à partir de l'anneau extérieur. L'AM migre dans son secteur en suivant un ordre en spirale allant de l'anneau extérieur vers le PV en parcourant seulement un anneau sur 2 pour chaque secteur (seulement les anneaux pairs ou les anneaux impairs en alternance pour chaque cycle – voir figure IV.4).

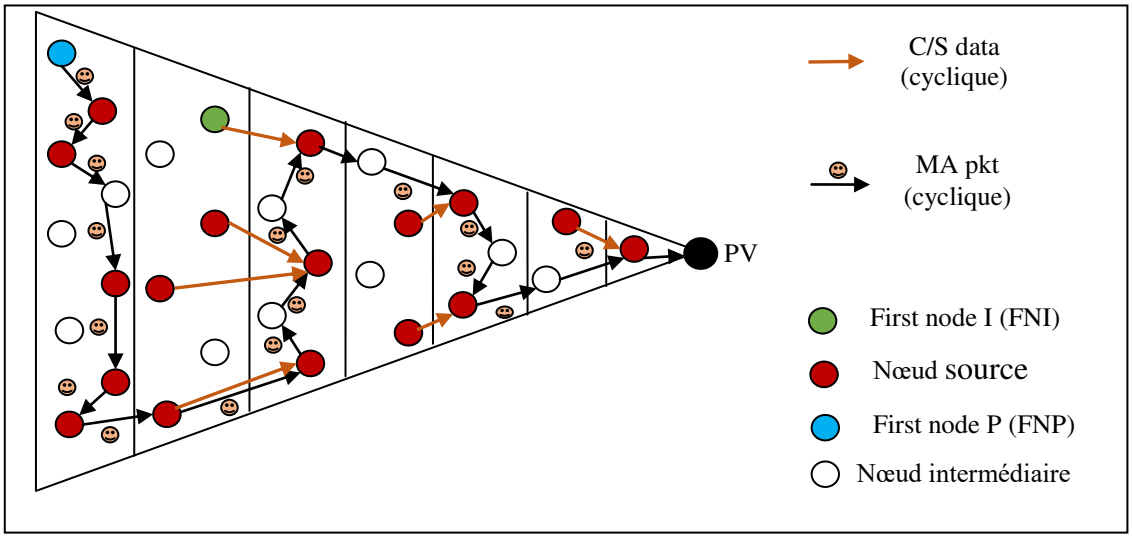
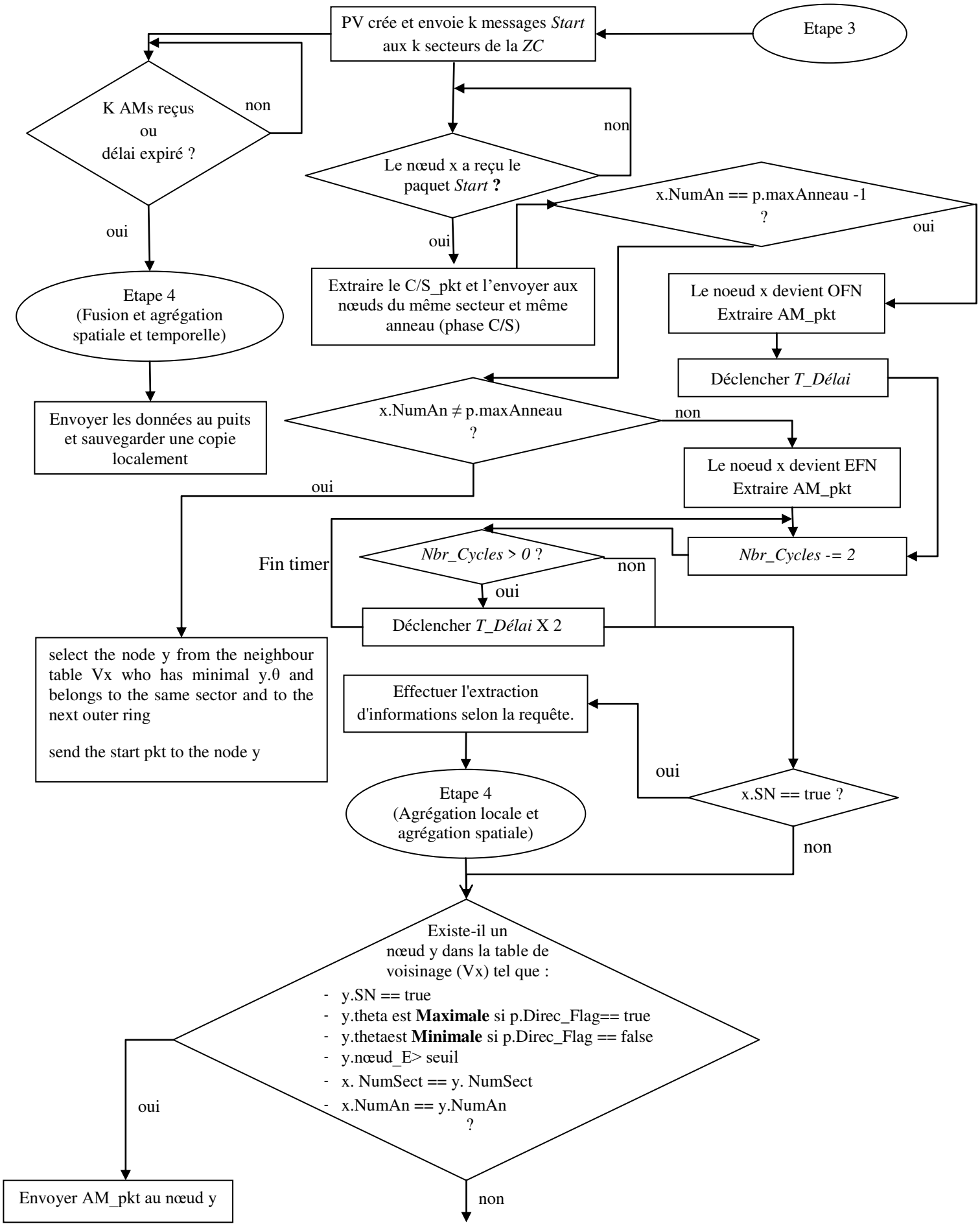


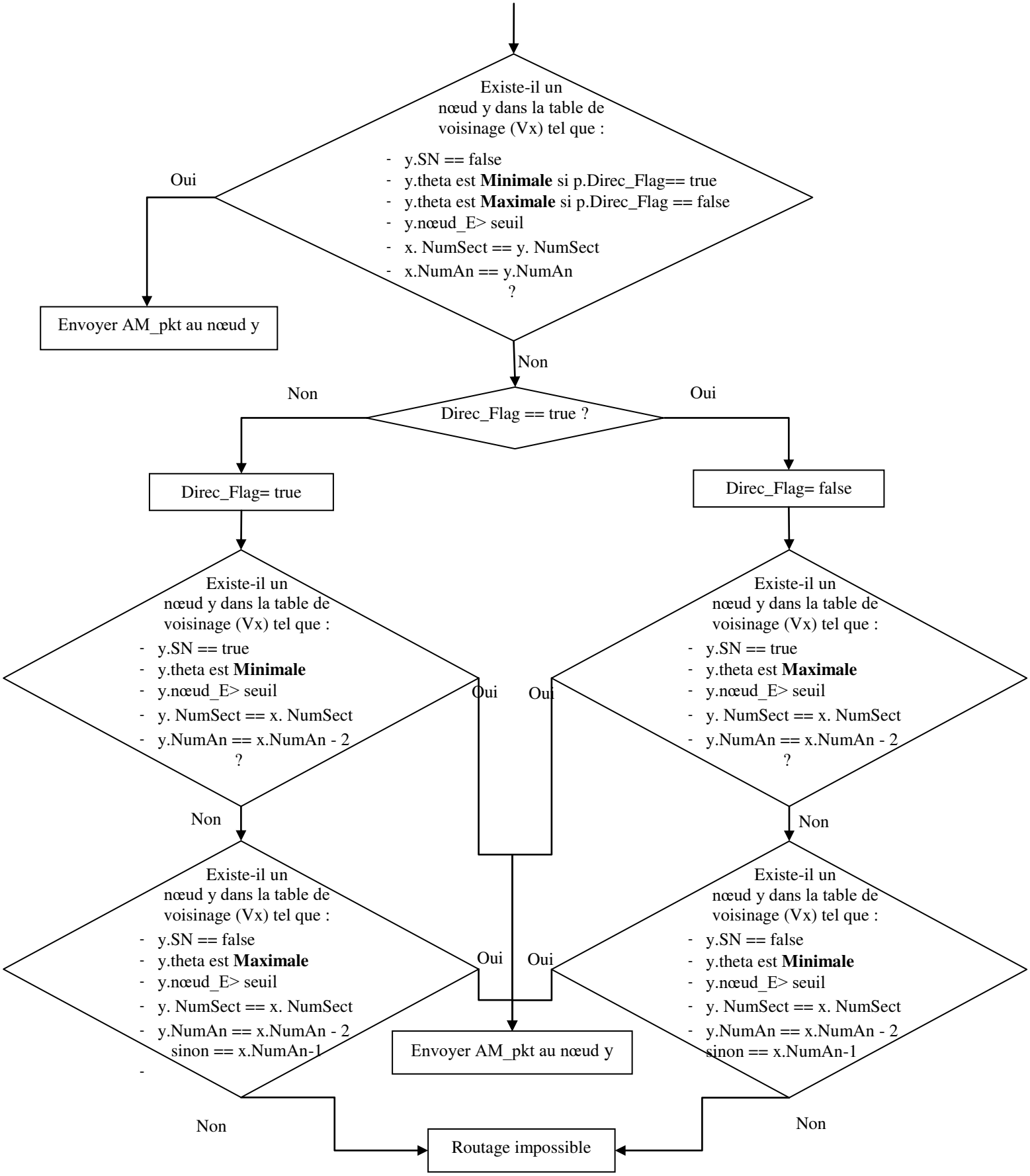
Figure IV.4 : Schéma de collecte des données cyclique au sein d'un secteur

La période de la tâche est divisée en plusieurs cycles périodiques, où chaque AM visite un sous ensemble des nœuds sources (seulement les nœuds sources appartenant aux anneaux impairs ou pairs de son secteur) et retourne les données collectées et agrégées au PV.

Dans IEMADI, c'est le puits qui relance les AMs à chaque nouveau cycle, et ceci va engendrer une perte de temps et d'énergie causée par le passage inutile de l'AM par les nœuds de bordures de tous les anneaux pour se déplacer du PV vers le *FN*. Pour éviter cette perte périodique de temps et d'énergie dans CHEESE, après l'expédition de chaque AM la première fois, chaque *FN* (et non le puits comme c'est le cas de IEMADI) déclenche *T_Delai* qui est utilisé pour lancer le prochain cycle et réexpédier les AMs afin de recueillir les données auprès de chaque secteur à nouveau.
A chaque expiration de *T_Delai*, chaque *FN* commence un nouveau cycle en envoyant chaque AM tout au long des capteurs ciblés, et réinitialise à nouveau *T_Delai* pour un prochain cycle jusqu'à la fin de la tâche.

Le diagramme suivant résume le fonctionnement de l'étape 3 :


Etape 3
PV crée et envoie k messages *Start* aux k secteurs de la *ZC*
K AMs reçus ou délai expiré ?
non
oui
Le nœud x a reçu le paquet *Start* ?
non
oui
x.NumAn == p.maxAnneau -1 ?
oui
Extraire le C/S_pkt et l'envoyer aux nœuds du même secteur et même anneau (phase C/S)
Le noeud x devient OFN
Extraire AM_pkt
Déclencher *T_Délai*
Etape 4 (Fusion et agrégation spatiale et temporelle)
Envoyer les données au puits et sauvegarder une copie localement
x.NumAn ≠ p.maxAnneau ?
non
oui
Le noeud x devient EFN
Extraire AM_pkt
Fin timer
Nbr_Cycles > 0 ?
non
oui
Nbr_Cycles -= 2
Déclencher *T_Délai* X 2
select the node y from the neighbour table Vx who has minimal y.θ and belongs to the same sector and to the next outer ring
send the start pkt to the node y
Effectuer l'extraction d'informations selon la requête.
oui
x.SN == true ?
non
Etape 4 (Agrégation locale et agrégation spatiale)
Existe-il un nœud y dans la table de voisinage (Vx) tel que :
- y.SN == true
- y.theta est **Maximale** si p.Direc_Flag== true
- y.thetaest **Minimale** si p.Direc_Flag == false
- y.nœud_E> seuil
- x. NumSect == y. NumSect
- x.NumAn == y.NumAn
?
oui
Envoyer AM_pkt au nœud y
non


Existe-il un nœud y dans la table de voisinage (Vx) tel que :
- y.SN == false
- y.theta est **Minimale** si p.Direc_Flag== true
- y.theta est **Maximale** si p.Direc_Flag == false
- y.nœud_E> seuil
- x. NumSect == y. NumSect
- x.NumAn == y.NumAn
?
Oui
Envoyer AM_pkt au nœud y
Non
Direc_Flag == true ?
Non
Oui
Direc_Flag= true
Direc_Flag= false
Existe-il un nœud y dans la table de voisinage (Vx) tel que :
- y.SN == true
- y.theta est **Minimale**
- y.nœud_E> seuil
- y. NumSect == x. NumSect
- y.NumAn == x.NumAn - 2
?
Oui
Existe-il un nœud y dans la table de voisinage (Vx) tel que :
- y.SN == true
- y.theta est **Maximale**
- y.nœud_E> seuil
- y. NumSect == x. NumSect
- y.NumAn == x.NumAn - 2
?
Oui
Non
Non
Existe-il un nœud y dans la table de voisinage (Vx) tel que :
- y.SN == false
- y.theta est **Maximale**
- y.nœud_E> seuil
- y. NumSect == x. NumSect
- y.NumAn == x.NumAn - 2 sinon == x.NumAn-1
-
Oui
Oui
Existe-il un nœud y dans la table de voisinage (Vx) tel que :
- y.SN == false
- y.theta est **Minimale**
- y.nœud_E> seuil
- y. NumSect == x. NumSect
- y.NumAn == x.NumAn - 2 sinon == x.NumAn-1
Envoyer AM_pkt au nœud y
Non
Non
Routage impossible

IV.4.4. Etape 4 : Agrégation et fusion des données collectées

CHEESE emploie l'agrégation à quatre niveaux : agrégation d'application, agrégation spatiale, agrégation temporelle et fusion des données captées.
Les 3 niveaux correspondants à l'agrégation d'application, l'agrégation spatiale et la fusion des données collectées sont similaires à ceux employés dans MADD [36]. De plus, pour une agrégation maximale, CHEESE permet d'éliminer les redondances temporelles à chaque fin de cycle de collecte des données au niveau du puits virtuel. Ceci permet de réduire davantage la taille des paquets envoyés au puits et de minimiser encore plus l'énergie consommée pendant le transfert des données entre la zone cible et le puits.

IV.4.4.1. Agrégation d'application

A l'aide du code de traitement de l'AM, chaque nœud source effectue des transformations locales sur les données brutes. Ainsi seules les données utiles seront extraites et transmises au puits (D_{xy} dans la figure IV.5). Ceci permet de réduire la quantité des données circulantes dans le réseau et de minimiser donc les coûts de communication en énergie et en bande passante.

IV.4.4.2. Agrégation spatiale

La région cible est divisée en plusieurs secteurs, où chaque secteur est dédié à réaliser plusieurs tâches simultanément. Chaque AM agrège pendant son parcours les données collectées de chaque tâche d'un secteur à part (D'_{xy} dans la figure IV.5). La séquence de résultat est fusionnée et sauvegardée au niveau du PV en attendant l'arrivée des données des autres AMs pour éliminer davantage les redondances spatiales entre les données des différents AMs.
En effet, à la réception du dernier AM, le PV va trier les paquets reçus des AMs par tâches, comme le montre le schéma de la figure IV.5. Ensuite, il effectue une autre agrégation spatiale des données de chaque tâche (D''_{xy} dans la figure IV.5).

IV.4.4.3. Fusion

La fusion des données permet de diminuer le coût de communication puisque le coût d'envoi d'un seul message long est généralement moins que celui d'envoi de plusieurs messages courts [36].
Après l'agrégation des tâches, au lieu d'envoyer un message pour chaque tâche, le PV fusionne les données agrégées de toutes les taches et de tous les AMs dans un seul paquet et l'envoi vers le puits. Comme le représente la figure IV.5. Le PV attend la réception de tous les AMs pour envoyer toutes les données fusionnées au puits. En cas de perte (panne, erreur de routage…) d'un agent mobile, le PV attend un délai ***TReponse*** et envoie seulement les paquets des AMs déjà reçus.

Contrairement aux solutions utilisant un seul AM, CHEESE permet de diminuer la perte des données collectées, puisque l'échec d'un AM n'entraine pas l'échec de toute la tâche. En effet, les autres AMs peuvent continuer leur taches indépendamment les

uns des autres. Alors que dans une solution à un seul AM, l'échec de l'AM entraine forcément la perte de toutes les données et donc l'échec de la tache complète.

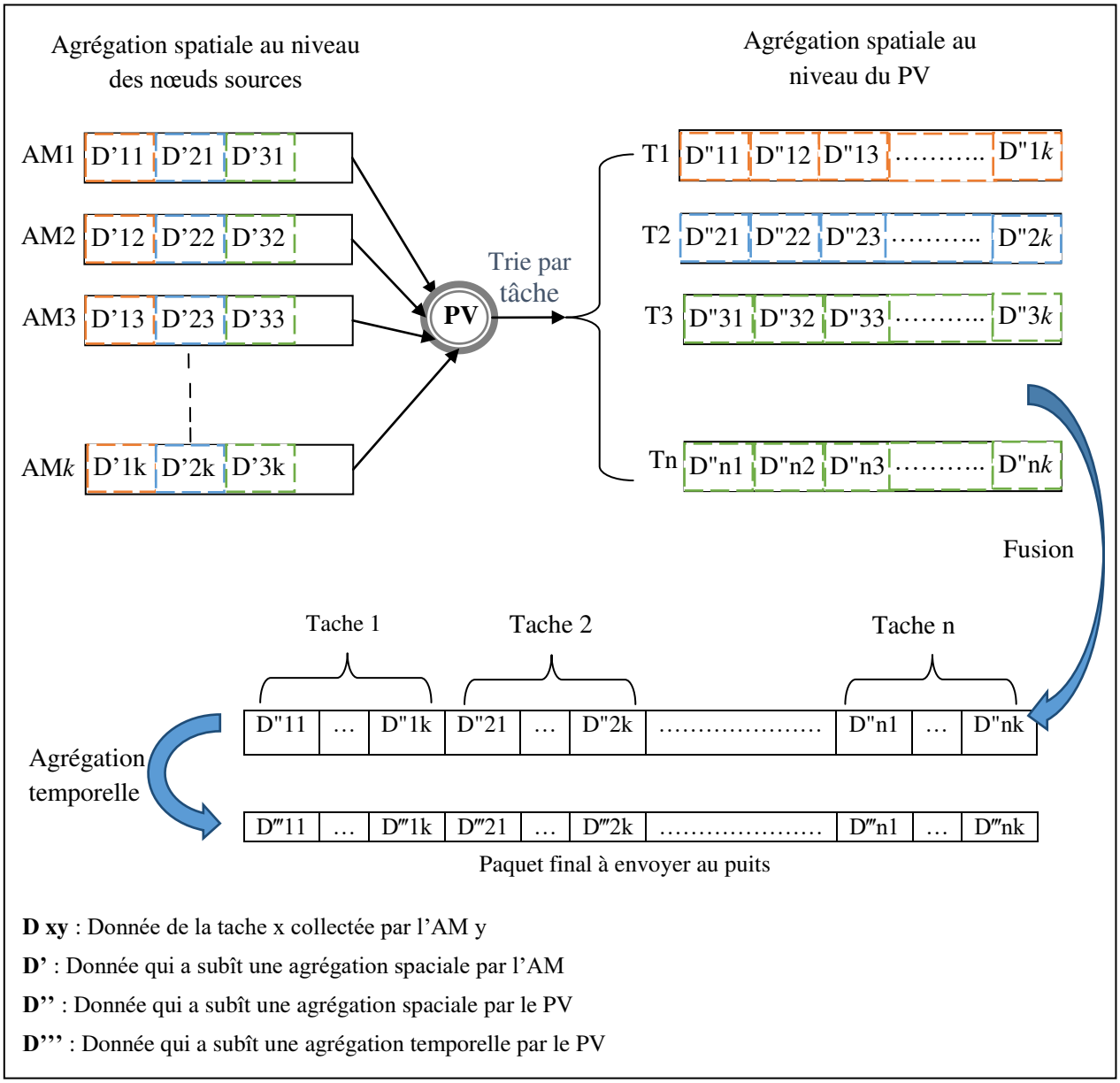


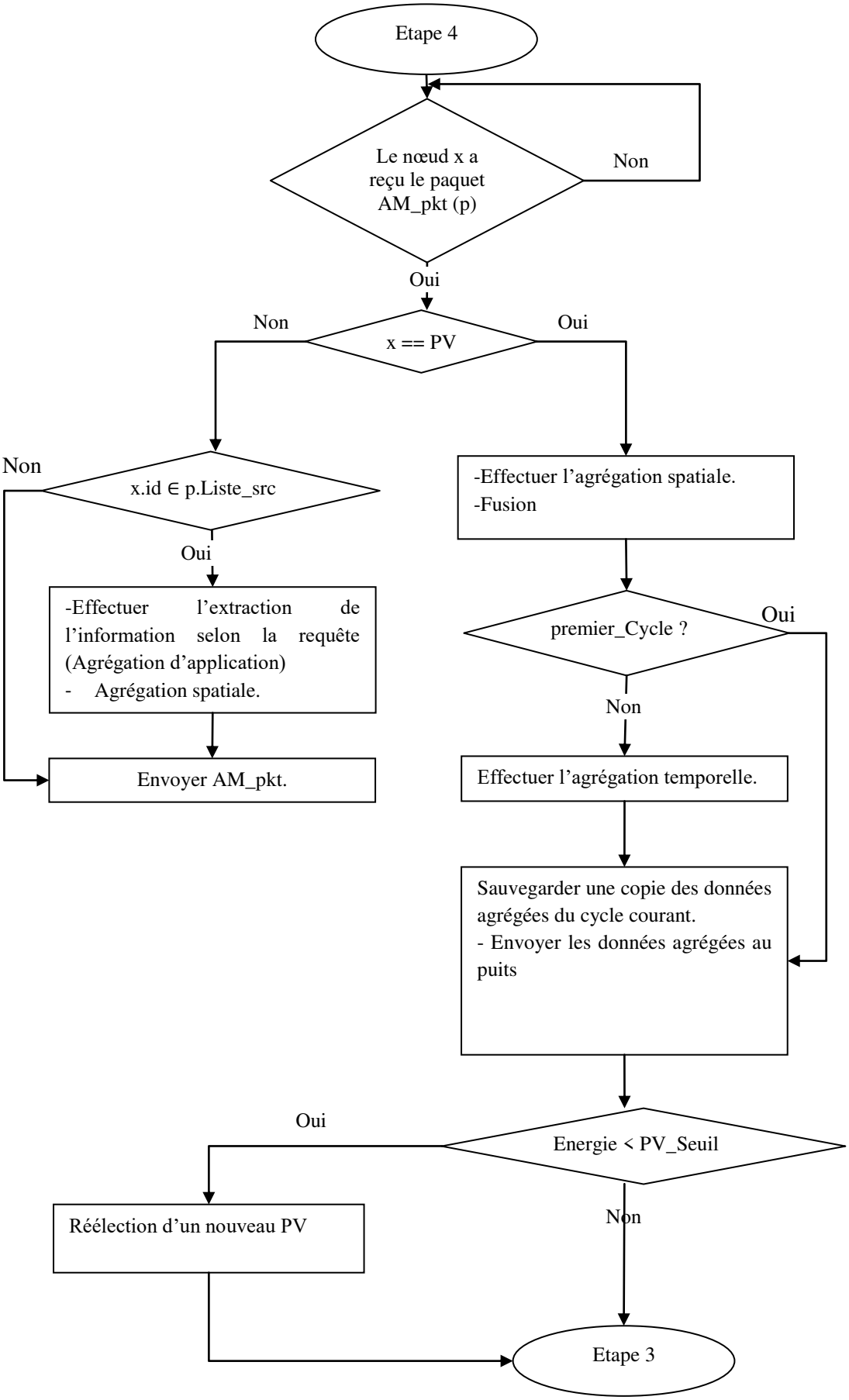
Figure IV.5 : Agrégation et fusion des données collectées.

De plus, dans IEMADI, l'échec d'un nœud source entraine forcément l'échec de l'AM de ce secteur et donc la perte de toutes les données de ce secteur. Par contre dans CHEESE, l'échec d'un nœud source n'entraine pas forcément l'échec de l'AM du secteur en question. En effet, Dans le cas où l'AM parcourt les anneaux pairs (respectivement impairs), et que le nœud défaillant appartient à un anneau impair (respectivement pair), seulement la donnée de ce nœud en panne sera perdue et non toutes les données de l'AM. Ceci permet une meilleure tolérance aux pannes dans CHEESE par rapport aux solutions qui font transiter l'AM par tous les nœuds sources.

IV.4.4.4. Agrégation temporelle :

A la fin du premier cycle, le PV sauvegarde une copie des données agrégées et fusionnées dans sa mémoire de stockage locale. Puis, à la fin de chaque cycle, le PV compare les données actuelles avec celles du cycle précédent (déjà sauvegardées durant le cycle précédent). Il élimine donc les redondances générées à travers le temps (par rapport à la copie sauvegardée du cycle précédent - D'''_{xy} dans la figure IV.5). Ensuite, le PV sauvegarde encore une copie des données agrégées du cycle courant pour l'utiliser dans l'agrégation du cycle suivant et ainsi de suite.

Le diagramme suivant résume le fonctionnement de la quatrième étape :


Etape 4
Le nœud x a reçu le paquet AM_pkt (p)
Non
Oui
x == PV
Non
Oui
Non
x.id ∈ p.Liste_src
Oui
-Effectuer l'agrégation spatiale.
-Fusion
-Effectuer l'extraction de l'information selon la requête (Agrégation d'application)
- Agrégation spatiale.
premier_Cycle ?
Oui
Non
Envoyer AM_pkt.
Effectuer l'agrégation temporelle.
Sauvegarder une copie des données agrégées du cycle courant.
- Envoyer les données agrégées au puits
Oui
Energie < PV_Seuil
Non
Réélection d'un nouveau PV
Etape 3

IV.5. Conclusion

Dans ce chapitre, nous avons proposé une amélioration des protocoles de collecte des données à multiples nommée CHEESE (Cooperative and hybrid energy efficient scheme for wireless sensor networks).

CHEESE utilise un puits virtuel dans la zone cible, afin de diminuer l'overhead de communication entre la zone cible et le puits. Ceci permet au puits d'envoyer un seul AM vers la zone cible, ensuite le PV va se charger de déployer de multiples AMs pour effectuer l'opération de collecte des données.

CHEESE opère sur le principe de multiples AMs qui parcourent les nœuds sources groupés en plusieurs secteurs centrés et suivant un chemin en forme de spirale (de l'anneau extérieur vers le centre) comme dans IEMADI. Seulement, CHEESE réduit ce chemin à presque sa moitié, en parcourant seulement un anneau sur deux au lieu de parcourir tous les anneaux. Ainsi, il y a réduction du temps de réponse et de l'énergie consommée avec une meilleure fiabilité.

Aussi, CHEESE étend les 3 niveaux d'agrégation des données dans MADD (agrégation d'application, spatiale et fusion) en rajoutant un quatrième niveau dédié à l'agrégation temporelle dans le but de diminuer davantage la taille des paquets transportés vers le puits. Ceci peut avoir un grand impact positif dans les réseaux à grande échelle où la zone cible est éloignée du nœud puits.

Afin de prouver l'efficacité de notre proposition, nous avons comparé les performances de CHEESE avec des travaux similaires. Ces évaluations sont présentées dans le chapitre suivant.

Chapitre V

Evaluation des performances

V.1. Introduction

Dans le chapitre précédent, nous avons détaillé notre contribution CHEESE qui utilise de multiples agents mobiles, afin d'économiser l'énergie et de réduire la latence. Pour évaluer les performances et tester l'efficacité d'un tel protocole par rapport à d'autres travaux similaires, il est indispensable d'effectuer des simulations.

La simulation nous permet de tester de nouvelles technologies et de nouveaux protocoles, de modéliser le monde réel, d'anticiper les problèmes qui pourraient se poser dans le futur et d'observer le comportement du système et de suivre son évolution dans le temps.

Pour ce faire, nous avons choisi le simulateur Glomosim [61] (Global Mobile Information System Simulator). Ce choix a été fait grâce à la spécialisation du simulateur aux environnements sans fil, et aussi à la diversité des fonctionnalités qu'il offre pour la gestion du système modélisé et des résultats de simulation.

V.2. Le simulateur GloMoSim

Couche Application (CBR, http, Telnet, FTP)
Couche Transport (TCP, UDP)
Couche Réseau (IP avec AODV, Flooding, Bellman-Ford, OSPF, DSR, WRP)
Couche Mac (CSMA, MACA, MACAW, FAMA, 802.11)
Couche Radio (Free Space, Rayleigh, Ricean, SIRCIM)
Couche Mobilité (Randomdrunken et Randomwaypoint)

Figure V.1 : Architecture en couches de Glomosim.

Global Mobile System Information Simulator (GloMoSim) est un environnement de simulation évolutive pour les grands réseaux de communication sans fil. Il utilise la simulation parallèle à événements discrets fournie par le langage PARSEC (Parallel Simulation Environment for Complex System) développé par PCL (Parallel Computing Laboratory) de l'université UCLA (University of California at Los Angeles). PARSEC permet à GloMoSim d'avoir une grande vitesse d'exécution et un bon passage à l'échelle. Glomosim simule le réseau avec un millier de nœuds reliés par une communication hétérogènes de capacité qui comprend le multicast, les communications asymétriques en utilisant des émissions directes par satellite, et les protocoles Internet traditionnels.

V.2.1. Architecture de GloMoSim

La plupart des systèmes réseaux sont construits en utilisant une approche basée sur les couches qui est semblable à l'architecture à sept couches de la norme OSI.

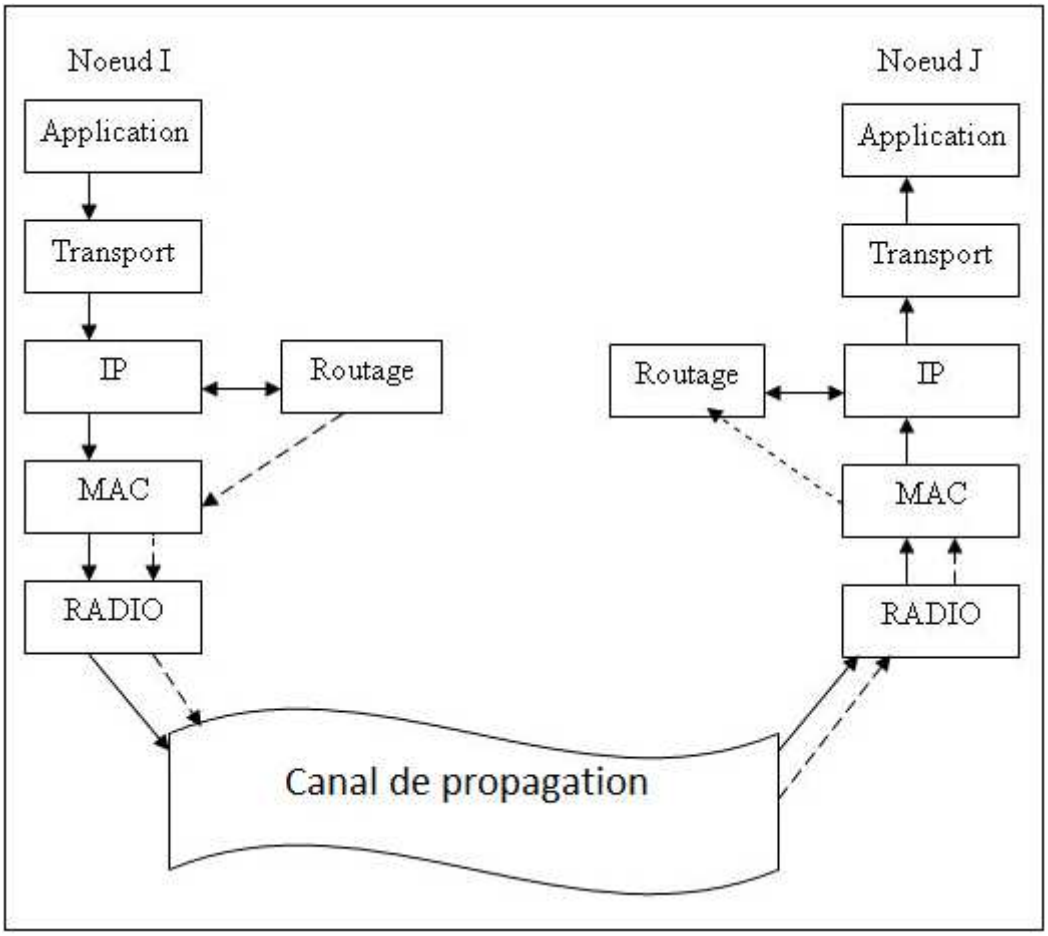


Figure V.2 : Transfert des paquets dans Glomosim.

Des APIs (Application Programming Interface) standards sont utilisées entre les différentes couches du simulateur. Ceci autorisera l'intégration rapide et facile des modèles et protocoles développés aux différentes couches par différents développeurs.

La figure V.1 montre les différentes couches du simulateur ainsi que les protocoles disponibles dans chaque couche. Le transfert de messages dans GloMoSim d'un nœud I vers un nœud J est réalisé selon le cheminement présenté dans la figure V.2.

V.2.2. Paramètres de configuration du simulateur

Avant de commencer la simulation, il faut d'abord préciser les paramètres de l'environnement simulé. En effet, des paramètres de configuration sont injectés au modèle de simulation à partir d'un fichier d'entrée nommé ***config.in*.** Ce fichier contient un ensemble de variables de paramètres de simulation avec les valeurs correspondantes. Parmi ces variables, nous trouvons :

- SIMULATION-TIME : Le temps de simulation. Il est exprimé en jours, heures, minutes, secondes, microsecondes et nano-secondes. Si on ne précise pas l'unité, la seconde est utilisée par défaut.

- TERRAIN-DIMENSIONS (X , Y) : Dimensions du terrain de simulation en mètres. X et Y représentent respectivement la largeur et la longueur du terrain**.**
- NUMBER-OF-NODES : nombre de nœuds dans le réseau à simuler.
- NODE-PLACEMENT et NODE-PLACEMENT-FILE : décrivent les emplacements des nœuds dans le terrain : (aléatoire, uniforme, rangés, en groupe, etc.).
- MOBILITY : spécifie si les nœuds seront mobiles ou pas.
- RADIO-BANDWIDTH : Largeur de la bande passante utilisée par les nœuds pour transmettre des messages.
- POWER-RANGE : représente la portée de communication de chaque nœud. Un nœud peut échanger des messages directs avec les nœuds se trouvant dans sa portée de communication (échange de messages à un saut).
- MAC-PROTOCOL : Protocole de la couche Mac (802.11, CSMA, FAMA (*Floor Acquisition Multiple Acces*), MACA (*Multiple Access Collision Avoidance*)).
- ROUTING-PROTOCOL : Protocole utilisé pour le routage des paquets. Nous y trouvons : BELLMANFORD, AODV, DSR, LAR1, WRP, ZRP, FISHEYE ou STATIC.
- APPLICATION : FTP, CBR, HTTP ou TELNET.
- TRANSPORT-PROTOCOL: UDP ou TCP.
- PROPAGATION-PATHLOSS : modèle de propagation: FREE-SPACE, TWO-RAY, PATHLOSS-MATRIX.

Il existe aussi d'autres paramètres comme la température, le bruit et les paramètres de statistiques.

V.3. Paramètres d'étude

Comme mentionné précédemment, le simulateur Glomosim permet de simuler des réseaux sans fil selon plusieurs paramètres. Ainsi, les caractéristiques par rapport auxquelles nous avons évalué notre système sont les suivantes :

V.3.1. La densité du réseau

La densité du réseau est indiquée par le nombre de nœuds dans un mètre carré. Ce paramètre est en relation directe avec la consommation de la bande passante et l'augmentation du trafic. En effet, une grande densité dans le réseau conduit à un grand risque de collisions et peut être une dégradation du système. C'est donc important d'étudier le comportement de notre solution par rapport à ce paramètre. Le nombre de nœuds est étudié dans l'intervalle [800, 1800].

V.3.2. La distance séparant le centre de la zone cible du puits

Nous nous sommes basés sur l'hypothèse que la distance entre le PV et le puits est supérieur strictement au rayon de la zone cible. Il est donc important d'évaluer les performances de notre protocole par rapport à la distance séparant le centre de la zone cible et le puits.

V.3.3. Taille de la zone cible

Ce paramètre est important dans les réseaux de capteurs, car il représente le nombre de nœuds sources qui participent à la tache de collecte des données. Nous allons étudier notre solution en faisant varier la taille de la zone cible entre 800m et 1200m.

V.3.4. Taille des données collectées

Ce paramètre est important pour montrer l'impact de la taille des données captées qui est généralement différente d'une tâche à une autre. Nous allons étudier notre solution en faisant varier la taille des données captées entre 0,5 Ko à 2 Ko.

V.3.5. Le nombre de cycles

Ce paramètre indique le taux de génération des données par les nœuds sources (demandé par l'application). C'est donc le nombre de fois que chaque agent mobile doit faire le tour de la zone cible pour collecter les données. Ce paramètre influe considérablement sur les performances du système conçu. Nous varions sa valeur entre 20 et 100 cycles.

V.3.6. Le nombre d'agents mobiles parallèles

Le nombre d'agents mobiles opérationnels en parallèle est un facteur clé dans les protocoles à multiples agents mobiles. Il est déduit suivant l'angle α. En effet, lorsque α augmente, le nombre d'agents mobiles diminue et vice versa.

V.4. Métriques à mesurer

En se basant sur les paramètres mentionnés précédemment, nous allons mesurer les métriques importantes pour évaluer les performances de notre approche :

V.4.1. L'énergie

Le paramètre d'énergie est très important dans les réseaux de capteurs sans fil, car ils sont munis d'une faible batterie généralement non rechargeable.

Au moment de la réception ou l'émission des paquets, chaque nœud dans le réseau consomme de l'énergie, ainsi que dans le cas où il exécute des opérations de traitement de données (agrégation de données).

V.4.2. Délai de délivrance des données

Ce paramètre représente le temps nécessaire pour acheminer les données collectées par les nœuds sources vers le puits. Ce paramètre pourrait être très important pour certaine applications critiques.

V.4.3. L'energie X Délai

Lorsque les performances en énergie et délai sont confondues, nous utilisons l'énergie X délai pour pouvoir juger de l'efficacité des protocoles les uns par rapports aux autres.

V.4.4.Taux de succès de la tâche

Ce paramètre indique le nombre de données délivrées au puits par rapport au nombre de données générées par les capteurs (et qui doivent donc être toutes délivrées au puits dans le cas idéal).

V.5. Résultats de simulation

Nous avons comparé notre proposition CHEESE avec trois autres travaux de différentes catégories:

- Protocole Client/serveur : DD [11]
- Protocole à base d'un seul AM : MADD [36]
- Protocole à base de multiples AMs : IEMADI [54]
- Notre contribution hybride : CHEESE [55]

V.5.1.Paramètres de simulation par défaut

Dimension du terrain	(2700m, 1700m)
Nombre de noeuds	1000
Centre de la zone cible	(1700, 850)
Diamètre de la zone cible	800m
Nombre d'AMs	8 (α= 45°)
Durée de simulation	15 minutes
Portée radio des capteurs	200 m
Position des noeuds	Uniforme
Mobilité des noeuds	statique
Débit de transmission	1Mbit/s
Taille de chaque donnée captée	500 bytes

Table V.1. Paramètres de simulation.

V.5.2.Energie consommée

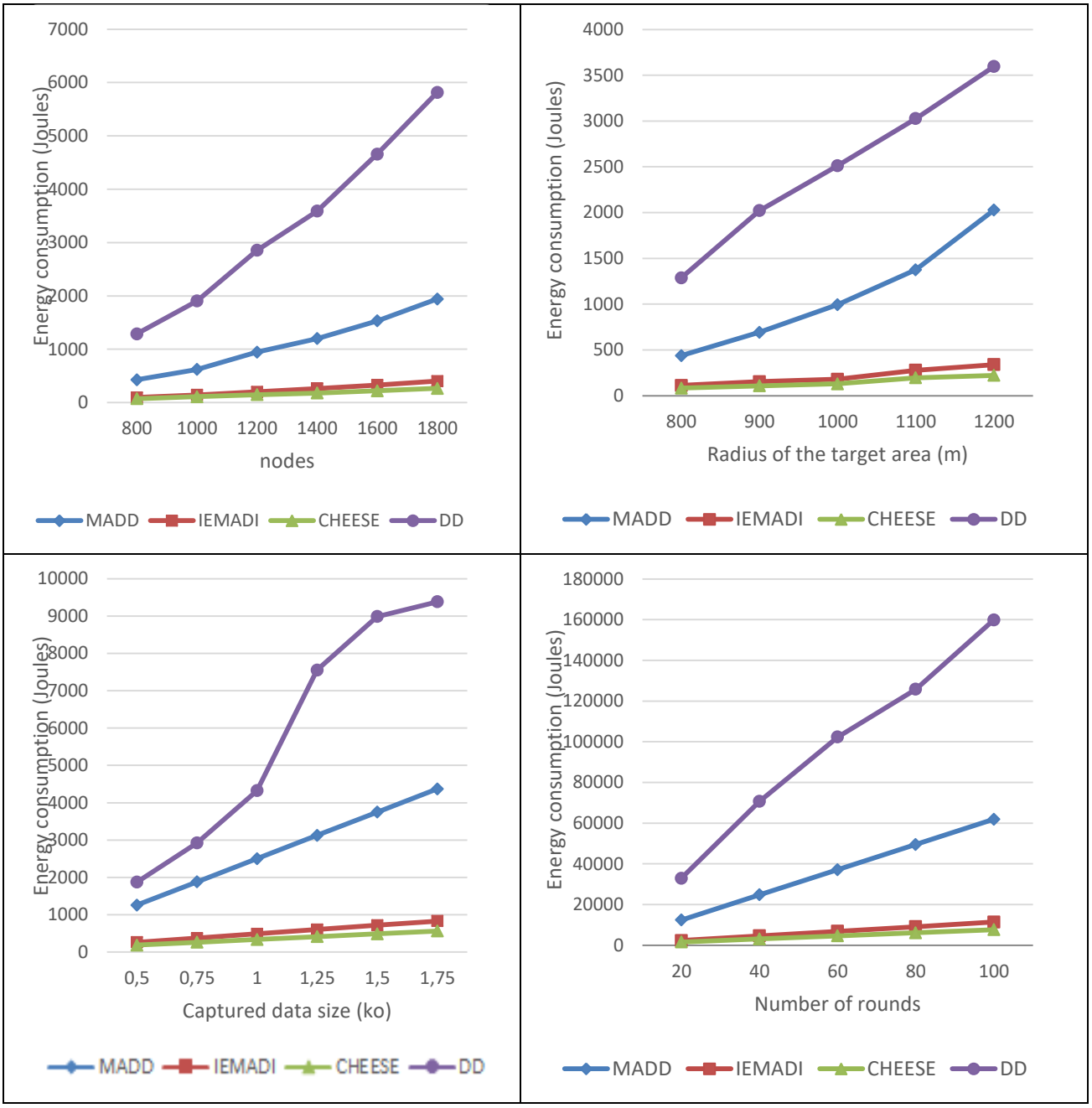


Figure V.3 : Consommation d'énergie

Dans la figure ci-dessus la consommation d'énergie augmente lorsque le nombre de nœuds augmente dans le réseau, car plus le nombre des nœuds augmente et plus le trafic réseau augmente.

Nous remarquons que l'énergie consommée dans DD est supérieure aux autres protocoles. Ceci est dû au fait que chaque données est envoyée individuellement au puits dans DD engendrant donc beaucoup de trafic et de collisions contrairement aux 3 autres solutions AM qui envoient moins de paquets de données (fusionnées et agrégées) en diminuant ainsi le trafic réseau.

Cependant, le protocole MADD consomme plus d'énergie que CHEES et IEMADI. Ceci est dû au fait que MADD emploi un seul AM pour collecter toutes les données de la zone cible et fait donc transporter ce paquet qui s'alourdie au niveau de chaque nœud source dans toute la zone cible, alors que CHEES et IEMADI dispatchent plusieurs AMs simultanément qui se partagent la tache de collecte et donc les données parcourent seulement une partie de la zone cible (un secteur) au lieu de parcourir toute la zone cible.

La consommation d'énergie de CHEES est inférieure à celle d'IEMADI puisque le chemin de l'AM est réduit à la moitié dans CHEESE.

V.5.3.Délai

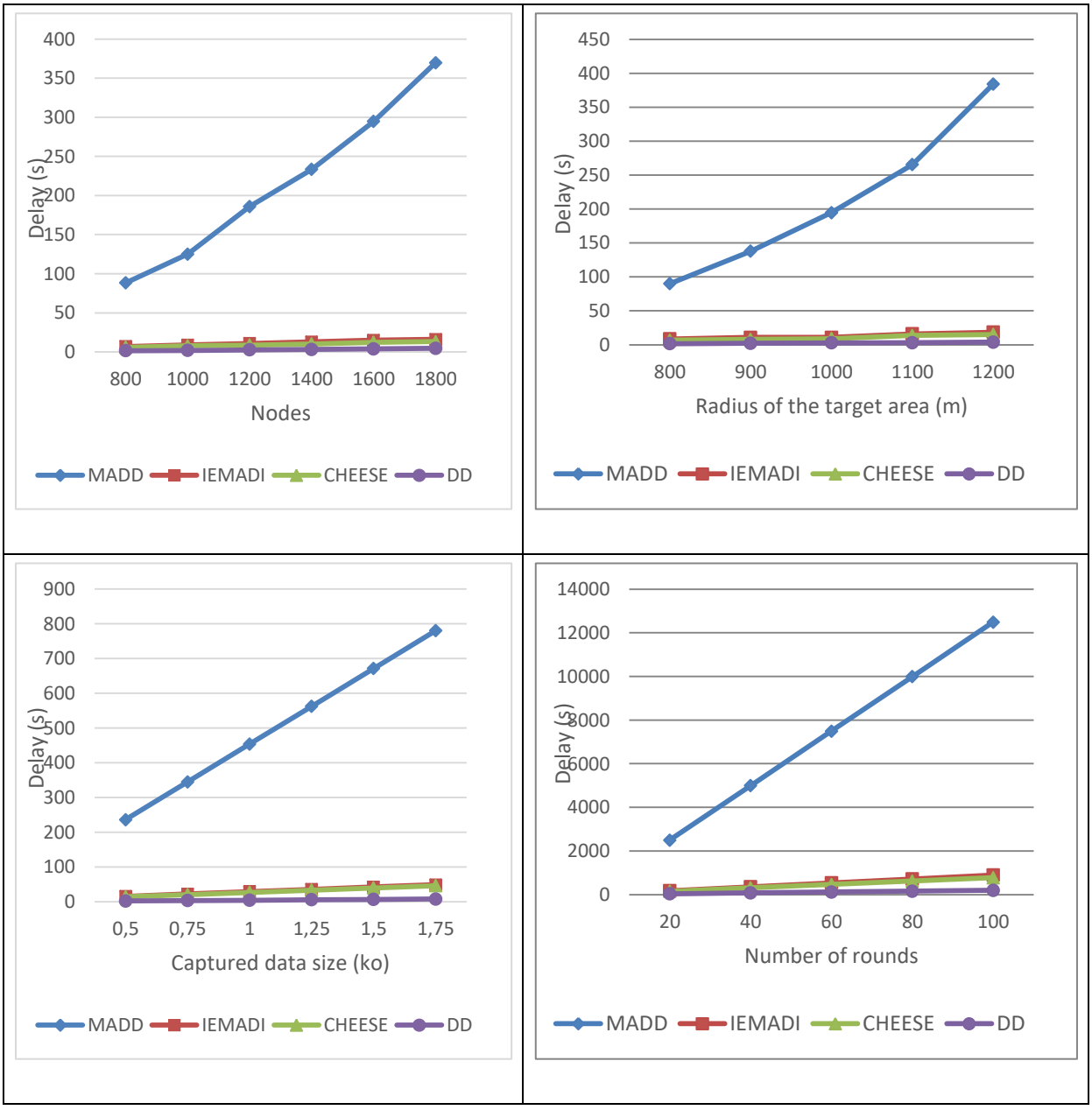


Figure V.4. Délai

La figure V.4 montre la comparaison entre CHEESE, IEMADI, MADD et DD, en termes de temps de réponse. Nous notons que le temps de réponse augmente avec l'augmentation de la taille du réseau, due à la croissance de l'itinéraire et de la taille de l'AM. À fur et à mesure que la taille du réseau augmente, cela prend plus de temps pour transmettre un agent puisque ce dernier accumule plus de données. CHEESE et IEMADI prennent approximativement 95% moins de temps que MADD pour accomplir un cycle. C'est dû à l'utilisation des multiples agents mobiles exécutés en parallèle. Le temps de réponse pour DD est le plus court. C'est dû au transfert direct de données dans le paradigme Client/Server sans la perte de temps pour la construction de structure et l'agrégation de données, comme font les paradigmes agents mobiles.

V.5.4. Energie X Délai

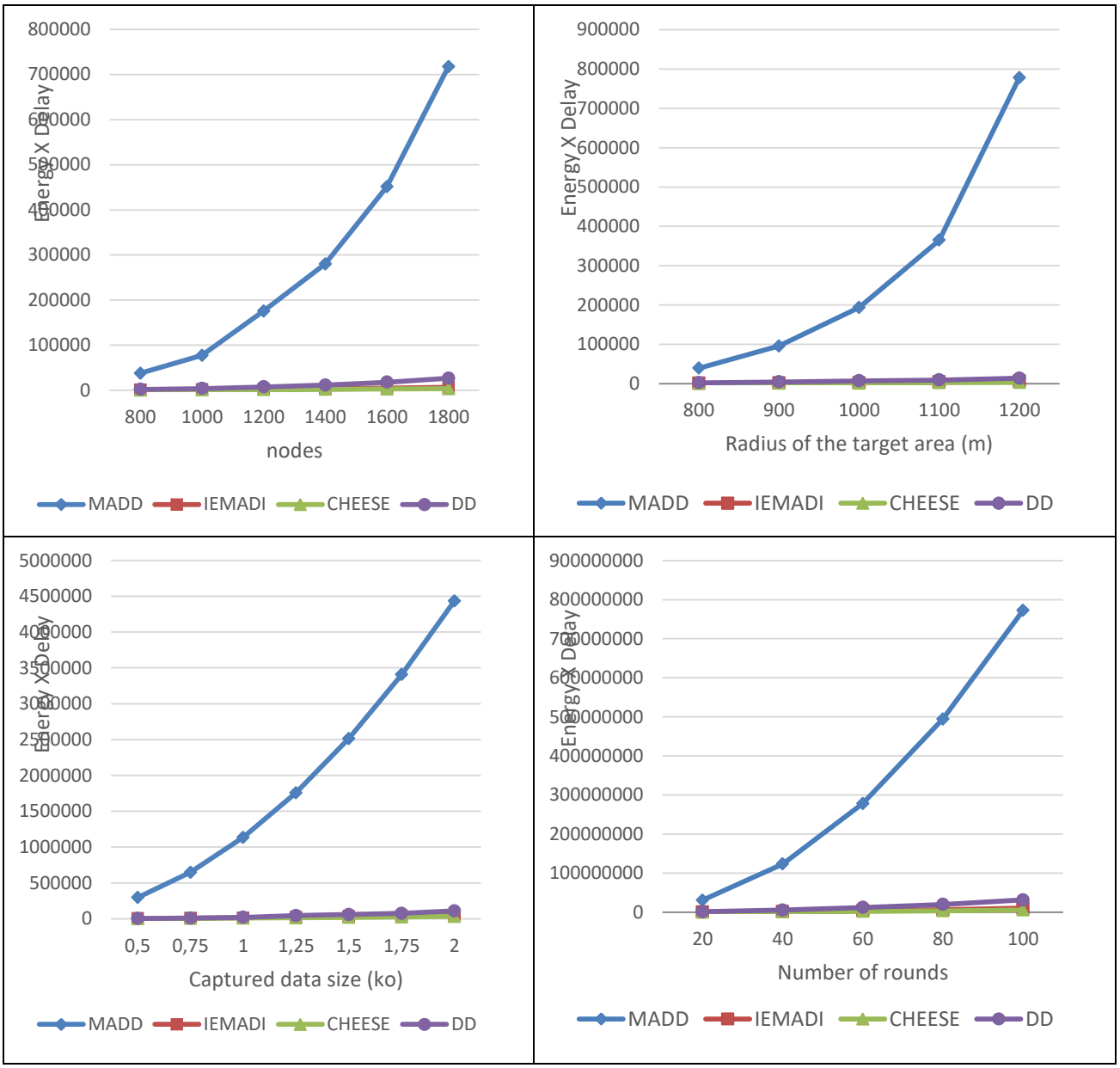


Figure V.5. Energie X Délai.

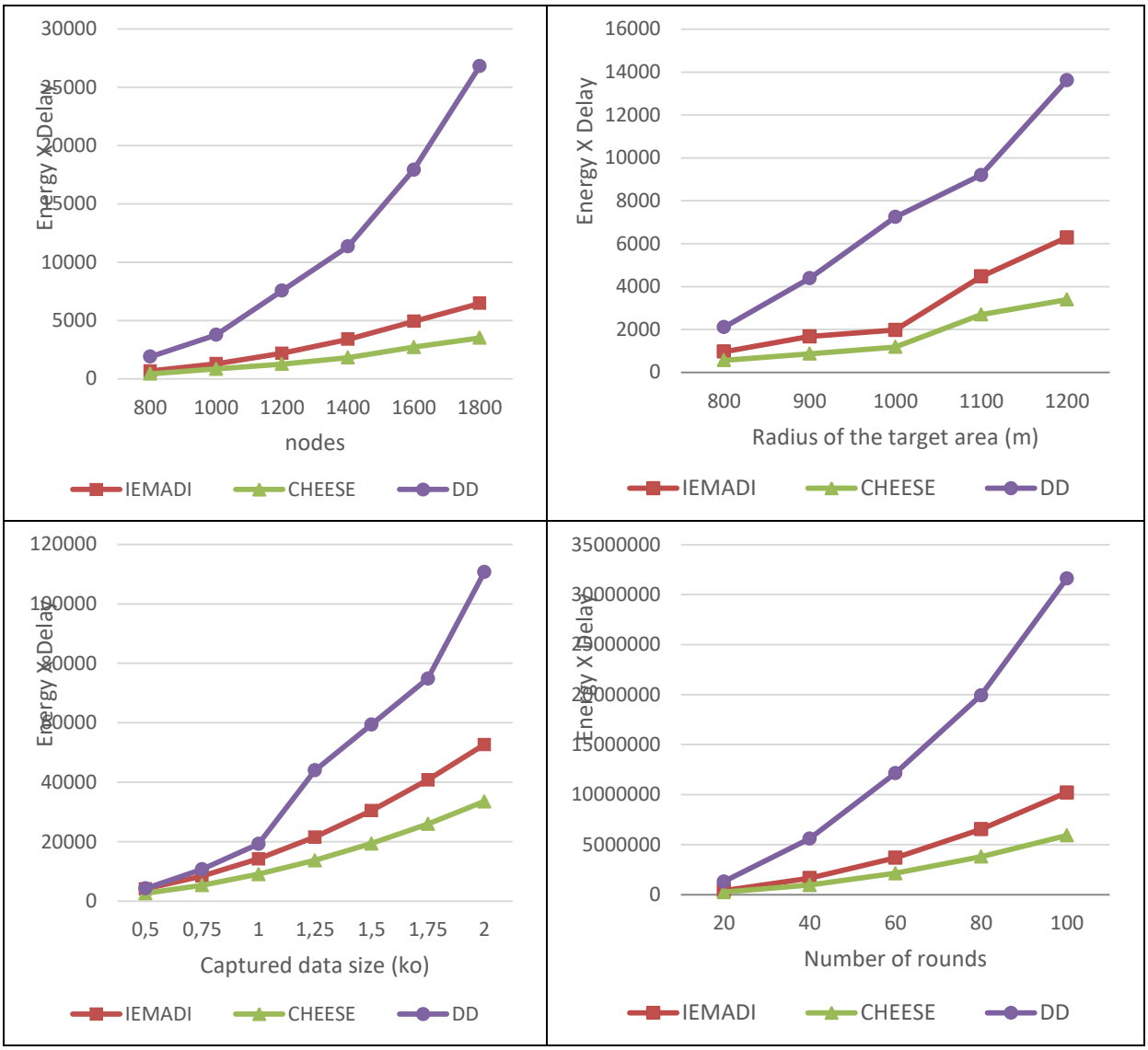


Figure V.6. Energie X Délai (échelle agrandie).

Les graphiques ci-dessus sont obtenus en multipliant les valeurs de chaque deux graphiques des deux figures V.3 et V.4. Ce paramètre peut être significatif quand la qualité de service demandée est balancée entre l'énergie et la latence.

Les résultats de délai de CHEESE obtenus sur la figure V.4 sont meilleurs qu'IEMADI et MADD, mais le protocole Client/serveur DD a un meilleur délai. Cependant, nous notons sur la figure V.5, que CHEESE a les meilleures performances en terme d'Energie X Délai.

V.5.5. Taux de succès

Dans les graphes de la figure V.7, nous avons varié le nombre de cycles de 1 jusqu'à 180 cycles, pour étudier le taux de succès (nombre de données collectées/nombre de données générées) de chaque algorithme. Le taux de succès diminue à chaque fois que le nombre de cycles augmente, ce qui est très attendu puisqu'à chaque fois qu'on rajoute des cycles, la durée de la tache augmente, les nœuds consomment plus d'énergie, et leur batterie lâche à fur et à mesure engendrant des pertes des nœuds et des données à collecter.

Nous remarquons que la solution C/S reste toujours meilleure en taux de succès puisque les données sont envoyées individuellement au puits via multiples chemins, contrairement aux 3 autres solutions AMs qui perdent toutes les données transportées par l'AM lorsque celui-ci échoue. En effet, dans MADD, cette perte est très grande à partir du 10ème cycle engendrant la perte de toutes les données collectées du cycle courant (un seul AM pour chaque cycle). Par contre dans IEMADI et CHEES, seulement les données du secteur de l'AM en panne qui sont perdues, les autres paquets (AMs) des autres secteurs vont être reçus normalement.

Cependant le graphe de CHEES a un taux de 95% avant 80 cycles qui est légèrement plus petit comparé à IEMADI avec 99% de taux de succès. Cette légère perte est causée par la phase C/S de CHEES qui parfois ne transfère pas les données à temps vers l'anneau précédent avant le passage de l'AM par cet anneau. Malgré cela, CHEES marque un meilleur taux de succès qu'IEMADI entre 80 et 180 cycles ; ceci s'explique par le fait que l'échec d'un nœud dans un secteur d'IEMADI entraine forcément l'échec de l'AM de ce secteur. Par contre, dans CHEES l'échec d'un nœud dans un secteur n'entraine pas forcément l'échec de l'AM de ce secteur. Car, ce nœud peut bien faire partie de la phase C/S, et donc ceci entraine la perte d'une seule donnée (celle du nœud en panne) au lieu de perdre tout le paquet AM.

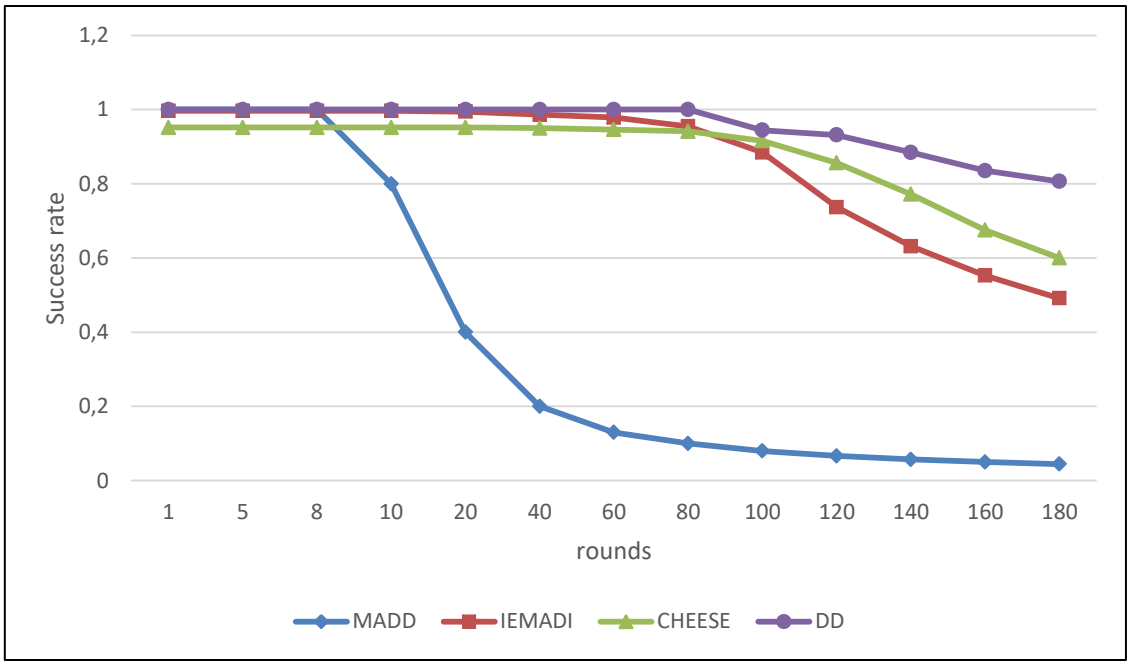


Figure V.7.Taux de succès

V.5.6. Impact du nombre d'AMs

Dans les graphes de la figure V.8, nous avons varié le nombre d'AMs pour les deux algorithmes CHEESE et IEMADI. Nous remarquons que la latence diminue à chaque fois que le nombre d'AMs augmente. Ceci est tout à fait attendu, puisque les AMs agissent en parallèle

et chaque AM aura donc à collecter moins de données dans un secteur plus petit (α plus petit). Mais le délai de CHEES est toujours meilleur que celui de IEMADI, ceci est dû au fait que les AMs dans CHEES parcourent seulement la moitié des anneaux parcourus par les AMs de IEMADI.

Concernant la consommation d'énergie, nous voyons bien que l'énergie consommée diminue à chaque fois que le nombre d'AMs augmente (α diminue). Ensuite, ça atteint un seuil minimal lorsque α ≈15°. Mais à partir de 15°, l'énergie augmente lorsqu'on augmente encore plus le nombre d'AMs (au-delà de 24 AMs). En effet, lorsque la valeur de α est trop petite, le secteur devient très étroit et chaque AM se déplace presque en ligne du PV vers l'anneau externe et retourne via presque le même chemin. En conséquence, chaque nœud doit transférer le paquet AM deux fois, engendrant ainsi une augmentation de la consommation d'énergie.

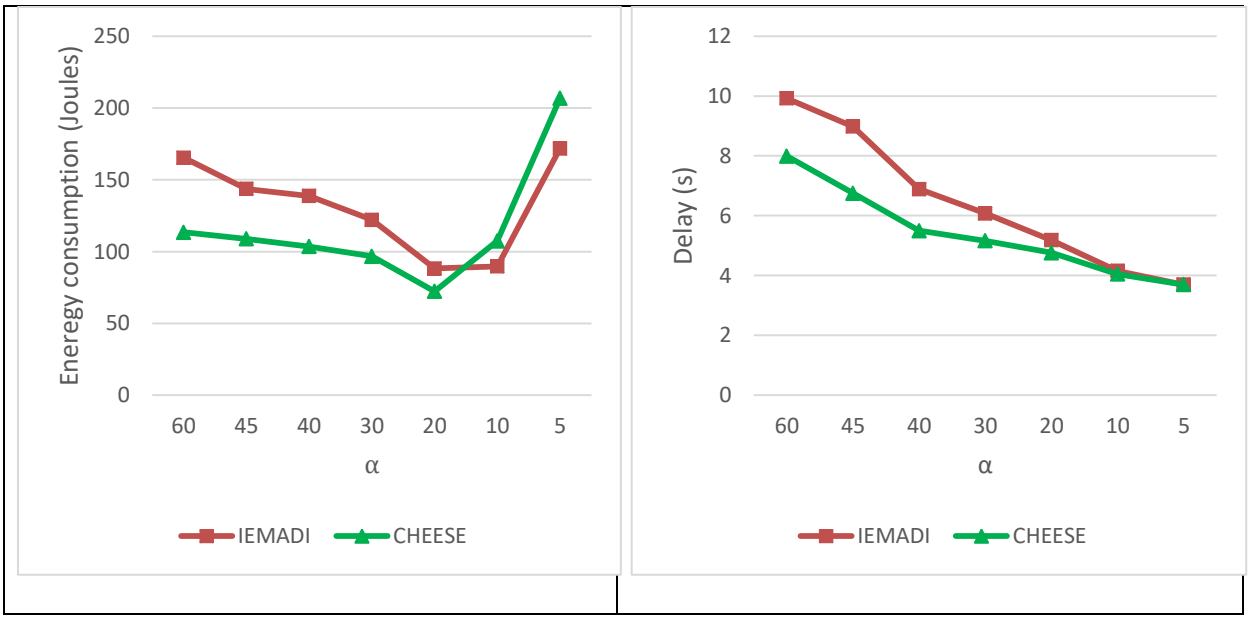


Figure V.8. Impact du nombre d'AMs

Cependant, CHEESE consomme plus d'énergie que IEMADI lorsque α<15° ; ceci se justifie par la phase C/S qui devient inutile dans CHEES, et qui consomme encore plus d'énergie inutilement lorsque α est trop petit (le secteur devient très étroit et les AMs passent donc par tous les nœuds sources).

V.6. Conclusion

Suite à notre conception présentée dans le chapitre précédent et la concrétisation de notre contribution, nous avons mis en œuvre l'ensemble des concepts qui caractérisent l'architecture proposée à l'aide du simulateur Glomosim.

Par conséquent, les résultats recueillis ont effectivement confirmé l'efficacité de notre contribution CHEESE dans la majorité des scénarios, comparée aux solutions Client/serveur, SIP et MIP. En effet, les résultats montrent que le protocole employant un seul AM est le plus gourmand en énergie et en temps de réponse avec une grande perte de données. Ce qui confirme que ce type de solutions est inconvenable pour les réseaux à grande échelle. La solution client/serveur consomme beaucoup d'énergie puisque les données sont transmises sans agrégation vers le puits. Cependant, son temps de réponse est meilleur, puisqu'aucune structure n'est construite et les données sont directement transmises vers le puits.

D'autre part, les solutions MIP sont meilleures pour l'économie de l'énergie des capteurs, puisque les AMs fusionnent les données avant de les délivrer. Cependant, elles marquent un retard de délivrance des données par rapport à l'approche C/S. Ceci se justifie par le ramassage en série des données par les AMs. Malgré cela, les solutions MIP restent meilleures que l'approche C/S lorsque le délai et l'énergie ont le même degré d'importance pour l'application.

Il est à noter que le nombre d'AMs dans les approches MIP est un facteur clé qui influe directement sur les performances du système. En effet, un grand nombre d'AMs engendre beaucoup de communications inutiles, entrainant une perte considérable en énergie qui est une ressource précieuse dans les RCSFs.

Conclusion générale

Conclusion générale

Dans ce travail, nous avons étudié le problème de dissémination des données dans les RCSFs. Nous avons vu que les travaux menés dans ce domaine ont évolués comme suit :

1. Protocoles Client/serveur : où le puits envoie des requêtes aux nœuds capteurs, ensuite chaque nœud capteur va traiter la requête du puits pour lui renvoyer individuellement les données désirées qui seront ensuite traitées et agrégées au niveau du puits. Cependant, il existe quelques inconvénients avec ce modèle qui devraient être pris en considération notamment pour les RCSFs comme la grande consommation énergétique.
2. Protocoles SIP : où le puits expédie un seul agent mobile vers les nœuds capteurs. Cet agent va transporter le code de traitement des données. De cette façon, les données seront traitées et agrégées localement au niveau des nœuds capteurs, ensuite l'agent va récolter séquentiellement ces données déjà traitées pour les faire parvenir au puits. Cependant, l'utilisation d'un seul agent mobile dans les RCSFs ne s'applique qu'aux réseaux de petite taille. L'augmentation de la taille du réseau conduit à l'augmentation du nombre de nœuds sources et donc l'alourdissement de l'AM qui consommera une grande énergie et qui tardera a finir la collecte des données.
3. Protocoles MIP : où la tache de récolte de données est partagée entre plusieurs AMs. La performance de ce modèle peut dépasser celle du modèle à base d'un seul agent mobile si les nœuds sont bien regroupés et les itinéraires sont bien conçus. Cependant, un système basé sur multiple AMs réintroduit plus de trafic de transmission dans le réseau. Par conséquence, le système à plusieurs AMs devrait mettre en place le meilleur équilibre sur le compromis entre la performance et l'efficacité.

Dans le cadre de cette étude, nous avons proposé un protocole coopératif et hybride nommé CHEESE (Cooperative and hybrid energy efficient scheme for wireless sensor networks), où nous avons travaillé sur les améliorations suivantes :

- Utilisation d'un puits virtuel ; dans le but d'envoyer un seul AM vers la zone cible qui sera hébergé par le PV, ensuite ce PV se charge de créer plusieurs AMs qui récolteront les données dans la zone cible. Vers la fin des collectes des AMs, le PV agrège et fusionne toutes les données pour les envoyer en un seul paquet vers le puits. De cette façon, nous évitons le trafic inutile des multiples AMs entre le puits et la zone cible.
- Etablissement d'un chemin dynamique allégé ; où l'AM ne passe pas par tous ses nœuds source comme font les autres solutions. Il passe seulement par un sous ensemble (presque la moitié) des nœuds sources, les autres nœuds sources non visités par l'AM envoient leur données aux nœuds sources voisins appartenant au chemin de l'AM. De cette façon, nous accélérons la vitesse de parcours de l'AM pour réduire le temps de réponse et nous assurons une meilleure tolérance aux pannes.
- Partage de la charge entre tous les nœuds de la zone cible ; en changeant l'itinéraire de l'AM à chaque cycle. En effet, L'AM dans CHEESE alterne entre 2 itinéraires dynamiques à chaque cycle de collecte périodique des données. Ceci a pour but

d'équilibrer la charge dans la zone cible pour éviter la chute énergétique rapide des nœuds appartenant à l'itinéraire de l'AM.

- L'agrégation maximale dans CHEESE en éliminant les redondances temporelles à chaque cycle de collecte des données au niveau du puits virtuel, en plus des agrégations et fusions habituelles utilisées par les autres solutions.

Les résultats de simulations ont prouvé l'efficacité de l'algorithme proposé CHEESE en ce qui concerne la durée de la tâche et la quantité d'énergie consommée ainsi que le taux de succès de la tâche dans plusieurs scénarios. Cependant, ce travail ouvre de nombreux défis et perspectives à l'avenir à savoir :

- La détermination du nombre optimal d'AMs : Le nombre d'agents mobiles est un facteur important dans la planification d'itinéraire. En effet, les résultats de simulations ont montré qu'un nombre élevé d'agents mobiles produit plus de flux de transmission. Par conséquent, plus d'énergie va être consommée. Cependant, l'utilisation d'un nombre réduit d'agents mobiles pose le problème de retard dans le temps de réponse. Donc il faut toujours songer à chercher le nombre optimal d'agents mobiles.
- La décentralisation des tâches des AMs : Dans CHEESE, toutes les données recueillies sont envoyées au puits virtuel situé au centre de la zone cible, et vu que les agents mobiles portent de plus grands paquets à fur et à mesure qu'ils avancent vers le PV, alors les nœuds autour du PV risquent une déplétion rapide de leur énergie. Il faut donc penser à décentraliser la destination des AMs.
- Assurer la sécurité des AMs **:** le principal inconvénient des AMs est le risque en sécurité. Par exemple, un AM malicieux peut endommager un hôte particulier, ou alors un hôte malicieux peut modifier le fonctionnement de l'agent…etc. Il faut donc toujours penser à mettre en place un système de sécurité des AMs.

Références bibliographiques

Références bibliographiques

[1] J. Cartigny et al. « Localized LMST and RNG based minimum energy broadcast protocols in ad hoc networks ». Ad Hoc Networks, 3(1):1–16, 2005.

[2] S. Moussaoui. « Contribution à l'amélioration de l'accessibilité dans les réseaux mobiles ad hoc », Thèse de doctorat en Informatique, USTHB 2007.

[3] J. Lu et F. Valois. « On the Data Dissemination in WSNs ». In the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007) 0-7695-2889-9/07, 2007.

[4] A. Tavakoli et al. « An empirical study of low-power wireless ». ACM Transactions on Sensor Networks, 6(16):1– 49, 2010.

[5] C. Cubat et D. Cros. « Agents Mobiles Coopérants pour les Environnements Dynamiques ». Thèse de doctorat, Institut National Polytechnique de Toulouse ,2 décembre 2005.

[6] M. Chen, S. Gonzalez et V.C. Leung. « Applications and design issues for mobile agents in wireless sensor networks ». IEEE Wireless Commun, vol. 14, pp. 20–26, 2007.

[7] X. Wang et al. «Multiple mobile agents' itinerary planning in wireless sensor networks: Survey and evaluation ». IET Communications. 5:1769–1776. doi: 10.1049/iet-com.2010.0638. 2011.

[8] L. Li et J. Y.Halpern. « Minimum-energy mobile wireless networks revisited ». IEEE International Conference on Communications ICC '01, Helsinki, Finland, 2001.

[9] S. Sentilles. « Architecture logicielle pour capteurs sans fil en réseau ». rapport de stage master recherche : Technologies de l'Internet, Université de Pau et des Pays de l'Adour, Juin 2006.

[10] L. Khelladi et N. Badache. « Les réseaux de capteurs : état de l'art ». Rapport de recherche N° LSI-TR0304, USTHB, Alger, Février 2004.

[11] C. Intanagonwiwat, R. Govindan, et D. Estrin. « Directed diffusion: a scalable and robust communication paradigm for sensor networks ». In Proceedings of the 6th Annual ACM/IEEE (MOBICOM '00), pp.56-67, USA, Août 2000.

[12] S. Harchi. « Un protocole de session dans les réseaux de capteurs sans fils. Réseaux et télécommunications ». Thèse de doctorat. Université de Lorraine, 2013.

[13] I. Chlamtac, I. Carreras et H. Woesner. « From internets to bionets: biological kinetic service oriented networks. The case study of Bionetic Sensor Networks ». Advances in Pervasive Computing and Networking. pp 75-95. Springer, Boston, MA, 2005.

[14] M. Val Machado et al. « Data Dissemination in Autonomic Wireless Sensor Networks». IEEE Journal on selected areas in communications, vol. 23, NO. 12. 2005.

[15] D. Niculescu. « Communication paradigms for sensor networks ». Communications Magazine, IEEE 43, no. 3, pp.116-122, 2005.

[16] C. Castelluccia et A. Francillion. « Protéger les réseaux de capteurs sans fil ». SSTIC2008, Renne, France, Juin 2008.

[17] N. Trigoni, A. Guitton et A. Skordylis. « Routing and processing multiple aggregate queries in sensor networks ». SenSys pp.391-392. 2006.

[18] S. Madden et al. « TAG: A Tiny Aggregation service for ad-hoc sensor networks ». In Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI 2002), Boston, Massachusetts, Decembre 2002.

[19] J. Chou, D. Petrovic et K. Ramchandran. « A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks ». IEEE INFOCOM 2003.

[20] J. Bacon et al. « TIME: An open platform for capturing, processing and delivering transport-related data ». In Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC), pp.687-691, 2008.

[21] A. Skordylis, A. Guitton et N. Trigoni. « Correlation-Based Data Dissemination in Traffic Monitoring Sensor Networks ». ACM CoNEXT'06, 2nd Conference on Future Networking Technologies, Lisboa, Portugal. Décembre 2006.

[22] R. S. Gray et al. « Mobile agents : Motivations and State of the Art‖ », AAAI/MIT-Press, 2001.

[23] B. Marzougui. « Contribution à la modélisation et à la vérification des systèmes multi agents ». Thèse de doctorat. Conservatoire national des arts et métiers (Paris). 2014.

[24] E. Yoneki et J. Bacon. « A survey of Wireless Sensor Network technologies: research trends and middleware's role ». Technical Report UCAM-CL-TR-646, University of Cambridge, UK, Sept. 2005.

[25] S. O. K. Hamza, «Découverte de services web via le Cloud computing à base d'agents mobiles,» chez thèse de doctorat, Université de Biskra, 2015.

[26] F. Aiello, G. Fortino et A. Guerrieri. « Using Mobile Agents as Enabling Technology for Wireless Sensor Networks ». The Second International Conference on Sensor Technologies and Applications, 2008.

[27] H. Qi et F.Wang. « Optimal itinerary analysis for mobile agents in Ad Hoc wireless sensor networks ». Int. Conf. Communications (ICC 2001), Helsinki, Finland, 2001.

[28] Y. Xu et H. Qi. « Mobile Agent Migration Modeling and Design for Target Tracking in Wireless Sensor Networks Ad Hoc Networks ». vol. 6, no. 1, pp. 1–16. Jan. 2007.

[29] R. S. Gray et al. Peterson et D. Rus. « D'Agents: Applications and Performance of a Mobile-Agent System ». Journal of Software Practice and Experience 32(6). May 2002.

[30] Q. Hairong, Xu. Yingyue et W. Xiaoling. « Mobile-Agent Based Collaborative Signal and Information Processing in Sensor Networks ». In Proceeding of the IEEE, Vol. 91, NO. 8, pp.1172-1183, Août 2003.

[31] P. Levis et D. Culler. « Maté: A Tiny Virtual Machine for Sensor Networks ». Proc. of the 10th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS X). San Jose (CA), USA, Oct 2002.

[32] X. Wang et al. « Multiple mobile agents' itinerary planning in wireless sensor networks: Survey and evaluation ». IET Commun, 5:1769–1776. doi: 10.1049/iet-com.2010.0638. 2011.

[33] H. Qadori et al. « A Spawn mobile agent itinerary planning approach for energy-Efficient data gathering in wireless sensor networks ». Sensors. 17:1280. doi:10.3390/s17061280. 2017.

[34] H. Qi et F. Wang. « Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks ». Proceedings of the IEEE International Conference on Communications (ICC 2001) 147–153, Helsinki, Finland (2001).

[35] M. Chen et al. « Mobile Agent Based Wireless Sensor Networks ». Journal of computers, vol. 1, NO. 1, Avril 2006.

[36] M. Chen et al. « Mobile agent-based directed diffusion in wireless sensor networks ». EURASIP Journal on Advances in Signal Processing, Article ID 36871, 13 pages, 2007.

[37] E. Shakshuki, H. Malik et M.K. Denko. « Software agent-based directed diffusion in wireless sensor network ». Telecommunication Systems 38(3-4): pp.161-174. 2008.

[38] J. Liu, et X. Jin. « Agent-based, energy efficient routing in sensor networks». In Proceedings of the third international joint conference on autonomous agents and multiagent systems (AAMAS' 04) .Vol. 1, pp.472-479. 2004.

[39] M. Chen et al. « Energy-efficient itinerary planning for mobile agents in wireless sensor networks ». In: IEEE international conference on communications (ICC'09), Dresden, pp.1–5. New York: IEEE. June 2009.

[40] A. Mpitziopoulos et al. « Deriving efficient mobile agent routes in wireless sensor networks with NOID algorithm ». Proceedings of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC 2007; Athens, Greece. September 2007.

[41] D. Gavalas et al. « New techniques for incremental data fusion in distributed sensor networks ». Proceedings of the 11th Panhellenic Conference on Informatics (PCI 2007) pp. 599–608; Patras, Greece. 18–20 May 2007.

[42] D. Gavalas et al. « Energy-efficient multiple itinerary planning for mobile agents-based data aggregation in WSNs ». Telecommun. Syst.; 63:531–545. doi: 10.1007/s11235-016-0140-z. 2016.

[43] A. Mpitziopoulos et al. « CBID: a scalable method for distributed data aggregation in WSNs‖, Hindawi Int. J. Distrib. Sens. Netw, Article ID 206517, DOI: 10.1155/2010/206517, 2010.

[44] C. Konstantopoulos et al. « Effective determination of mobile agent itineraries for data

aggregation on sensor networks », IEEE Trans. Knowl. Data Eng, pp. 1679–1693, 2010.

[45] G P. Gupta, M. Misra, et K. Garg. « An Energy Efficient Distributed Approach-Based Agent Migration Scheme for Data Aggregation in Wireless Sensor Networks ». J Inf Process Syst, Vol.11, No.1, pp.148~164, March 2015.

[46] M. Chen et al. « Balanced Itinerary Planning for Multiple Mobile Agents in Wireless Sensor Networks ». International Conference on Ad Hoc Networks ADHOCNETS 2010: Ad Hoc Networks pp 416-428. 2010.

[47] I. Aloui et al. « A new Itinerary planning approach among multiple mobile agents in wireless sensor networks (WSN) to reduce energy consumption ». International Journal of Communication Networks and Information Security (IJCNIS), Vol 7, No 2.2015.

[48] W. Cai et al. « GA-MIP: genetic algorithm based multiple mobile agents itinerary planning in wireless sensor network ». Proc. Fifth Int. Wireless Internet Conf. (WICON), Singapore, 2010.

[49] M. Chen et al. « Multi-agent itinerary planning for wireless sensor networks ». In: Bartolini N, Nikoletseas S, Sinha P, et al. (eds) Quality of service in heterogeneous networks », pp.584–597. Berlin, Heidelberg: Springer. 2009.

[50] S. Chaudhary, U. Kumar et M. Gambhir. «Review and comparison of Mobile Agent Itinerary Planning Algorithms in WSN». International Journal of Computer Sciences and Engineering (IJCSE) Vol. 7(6), E-ISSN: 2347-2693. 2019.

[51] M. Chen, S. Gonzalez, V. Leung. « Directional source grouping for multi-agent itinerary planning in wireless sensor networks‖ ». Proc. Int. Conf. ICT Convergence (ICTC), Jeju Isaland, Korea, 2010.

[52] G P. Gupta, M. Misra, et K. Garg. « Multiple Mobile Agents based Data Dissemination Protocol for Wireless Sensor Networks ». CCSIT 2012, Part I, LNICST 84, pp. 334–345, 2012.

[53] G P. Gupta, M. Misra et K. Garg. « An Energy Balanced Mobile Agents based Data Dissemination Protocol for Wireless Sensor Networks ». Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief. Pages 89-95. 2011

[54] G P. Gupta, M. Misra, et K. Garg. « Distributed information extraction in wireless sensor networks using multiple software agents with dynamic itineraries ». TIIS, 8(1):123–144, 2014.

[55] **D. Iabbassen et S. Moussaoui. « Cooperative and hybrid energy efficient schemE for wireless sensor networks ». International Journal of Computers and Applications, DOI: 10.1080/1206212X.2019.1609648. 2019.**

[56] A. Rais, K. Bouragba et M. Ouzzif. « Efficient hierarchical clustering routing in wireless sensor networks ». International Journal of Future Generation Communication and Networking, Vol. 9, No. 9, pp. 109–118, 2016.

[57] E. Ermel. « Localisation et Routage géographique dans les réseaux sans fil hétérogènes ».

Thèse de Doctorat de l'université Paris VI Pierre et Marie CURIE, Spécialité systèmes informatiques. Juin 2004.

[58] M. Nati. « Geographic routing: Mission possible ». Thèse de doctorat. Université de Rome. Italie, Mars 2008.

[59] S. Medjiah, T. Ahmed, F. Krief, P. Gélard. « AGEM : Un Protocole de Routage Géographique Angulaire Adaptatif ». Colloque francophone sur l'ingénierie des protocoles, Strasbourg, France, CFIP'2009

[60] Finn GG. « Routing and addressing problems in large metropolitan scale ». Internetwork. ISI Research Report ISU/RR-87-180, Mars 1987.

[61] X. Zeng, R. Bagrodia et M. Gerla. « GloMoSim: A library for parallel simulation of large-scale wireless networks ». Proceedings of the 12th Workshop on Parallel and Distributed Simulations–PADS '98, May 26–29, in Banff, Alberta, Canada, 1998.

[62] A. Giuseppe et al. « Energy conservation in wireless sensor networks: A survey ». Ad Hoc Networks, pp.537_568, 2009.

[63] **D. Iabbassen et S. Moussaoui. « Mobile agent and client/server data dissemination in wireless sensor networks ». Journal of Networking Technologies in Web Intelligence, 2015.**

[64] **D. Iabbassen et S. Moussaoui. « Data dissemination protocols in wireless sensor networks client/server versus mobile agent paradigms ». Fifth International Conference on Innovative Computing Technology (INTECH), pp.45-50. Pontevedra, Spain. 2015.**.

[65] S. El Falou. « Programmation répartie, optimisation par agent mobile ». Thèse de doctorat. Université de Caen. 2006.

[66] M. Bouallegue. « Protocoles de communication et optimisation de l'énergie dans les réseaux de capteurs sans fil ». Thèse de doctorat. Université du Maine. 2016.

[67] K. Sohraby, D. Minoli, et T. Znati. « Wireless Sensor Networks: Technology, Protocols, and Applications ». John Wiley & Sons, Inc.edition, 2007.

[68] H Q Qadori et al. « FuMAM: fuzzy-based mobile agent migration approach for data gathering in wireless sensor networks ». IEEE Access; 6: 15643–15652. 2018.

[69] I. F. Akyildiz, et al. « Wireless Sensor Networks: A Survey ». Computer Networks (Elsevier) Journal, pp.393-422, March 2002.

[70] H. Karl et A. Willig. « Protocols And Architectures For Wireless Sensor Networks ». Wiley-Interscience New York, NY, USA, 2007.

[71] C. Pomalaza-Rez. « Wireless Ad Hoc & Sensor Networks ». University of Oulu, Finland, 2004.

[72] E. Mercadal et al. « Improving the dynamism of mobile agent applications in wireless sensor networks through separate itineraries ». Computer Communications 36, 1011–

1023. 2013.

[73] H Q. Qadori et al. « CMIP: Clone Mobile-Agent Itinerary Planning Approach for Enhancing Event-to-Sink Throughput in Wireless Sensor Networks ». IEEE Access vol 6 (2018), pages: 71464-71473. Doi: 10.1109/ACCESS.2018.2882018. 2018.

[74] H Q. Qadori et al. « Multi-mobile agent itinerary planning algorithms for data gathering in wireless sensor networks: A review paper ». International Journal of Distributed Sensor Networks, Vol. 13(2). 2017.

[75] F. Derakhshan et S. Yousefi. « A review on the applications of multiagent systems in wireless sensor networks ». International Journal of Distributed Sensor Networks, Vol. 15(5). 2019.

[76] M. El Fissaoui, A. Beni-Hssane, et M. Saadi, « Multi-mobile agent itinerary planning-based energy and fault aware data aggregation in wire- less sensor networks ». EURASIP J. Wireless Commun. Netw, vol. 2018, no. 1, p. 92, 2018.

[77] A. Rais, K. Bouragba, et M. Ouzzif. « Determination of itinerary planning for multiple agents in wireless sensor networks ». Int. J. Commun. Netw. Inf. Secur., vol. 10, no. 1, pp. 99109, 2018.

[78] Xl. Zheng et Wan M. « A survey on data dissemination in wireless sensor networks ». Journal of computer science and technology, 29(3) : 470-486, 2014.