Titre: Title:	Mécanisme de collecte de données utilisant la carte des énergies et la qualité de services dans les réseaux de capteurs sans fil
Auteur: Author:	Abdelmorhit El Rhazi
Date:	2008
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	El Rhazi, A. (2008). Mécanisme de collecte de données utilisant la carte des énergies et la qualité de services dans les réseaux de capteurs sans fil [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8196/

UNIVERSITÉ DE MONTRÉAL

MÉCANISME DE COLLECTE DE DONNÉES UTILISANT LA CARTE DES ÉNERGIES ET LA QUALITÉ DE SERVICES DANS LES RÉSEAUX DE CAPTEURS SANS FIL

ABDELMORHIT EL RHAZI
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIAE DOCTOR
(GÉNIE INFORMATIQUE)
Octobre 2008

Your file Votre référence
ISBN: 978-0-494-48887-4
Our file Notre référence
ISBN: 978-0-494-48887-4

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

MÉCANISME DE COLLECTE DE DONNÉES UTILISANT LA CARTE DES ÉNERGIES ET LA QUALITÉ DE SERVICES DANS LES RÉSEAUX DE CAPTEURS SANS FIL

présentée par : EL RHAZI Abdelmorhit
en vue de l'obtention du diplôme de : Philosophiae Doctor
a été dûment acceptée par le jury d'examen constitué de :

M. GALINIER Philippe, Doct., président
M. PIERRE Samuel, Ph.D., membre et directeur de recherche
M. QUINTERO Alejandro, Doct., membre
M. KARMOUCH Ahmed, Ph.D., membre

À ma femme Hanane, mes deux enfants Hiba et Youssef
À toute ma famille

REMERCIEMENTS

Je tiens en premier lieu à remercier Dr. Samuel Pierre dont le soutien, les conseils et l'exemple m'ont guidé tout au long de ma recherche.

Je tiens également à remercier ma femme Hanane qui m'a supporté durant les longues années de préparation au détriment de ma disponibilité et mes devoirs d'époux.

Je remercie aussi mes parents, frères et sœurs pour leur soutien constant et indéfectible.

Je remercie le personnel du Laboratoire de Recherche en Réseautique et Informatique Mobile (LARIM) pour leur collaboration et surtout pour l'ambiance de travail chaleureuse.

Enfin, je remercie tous ceux que je n'ai pas eu la possibilité de nommer et qui m'ont soutenu, aidé ou encouragé d'une manière ou d'une autre durant ce long processus.

RÉSUMÉ

Les réseaux de capteurs sont considérés comme une génération de réseaux différente des autres types de réseaux, vu les contraintes et les caractéristiques de leurs nœuds. La principale caractéristique et la plus contraignante est le fait que les nœuds ont généralement des capacités d'énergie très limitées, facilement épuisables et difficilement remplaçables. Par conséquent, les sujets traitant de la consommation d'énergie ont beaucoup d'importance. En effet, la consommation d'énergie doit être optimisée pour pouvoir augmenter la durée de vie des réseaux et leurs couvertures. C'est un problème complexe vu qu'il fait intervenir beaucoup de paramètres combinant les états courants et futurs des capteurs ainsi que les exigences des applications.

Le sujet que nous avons choisi pour notre thèse se situe parmi les défis les plus prioritaires entravant le déploiement des réseaux de capteurs à grande échelle. Ceci est dû à plusieurs raisons. Premièrement, vu que les systèmes de capteurs sont déployés pour prendre des mesures et les envoyer aux applications, le mécanisme de la collecte des données constitue la partie la plus importante de ces systèmes. Deuxièmement, les capacités limitées des capteurs en énergie rendent l'optimisation de la consommation d'énergie cruciale. Enfin, la complexité des modèles mathématiques et le grand nombre de paramètres de ces problèmes rendent leur résolution non triviale.

Dans un premier temps, nous avons étudié les modèles existants de la consommation d'énergie dans les réseaux de capteurs. Ces modèles se sont avérés inadéquats à la conception d'un mécanisme de la collecte de données. En effet, la majorité de ces modèles ne considèrent pas le changement des états d'un capteur qui est toujours actif en train de capter ou de transmettre de l'information. L'analyse de ces modèles nous a permis de trouver une formule mathématique générale de la consommation d'énergie. Cette formule a été utilisée afin de définir la carte des énergies qui combine la prédiction de l'énergie consommée dans un intervalle de temps futur avec l'énergie résiduelle.

Nous avons aussi étudié et analysé les mécanismes de la collecte des données existants. Il s'est avéré que la majorité de ces mécanismes ne permettent pas de réduire la consommation d'énergie d'une manière efficace due au fait qu'elles ne considèrent qu'un seul aspect des réseaux de capteurs. À partir de ces faiblesses, nous avons conçu une

approche novatrice qui combine les exigences des applications avec la carte des énergies des capteurs. Cette approche constitue une première contribution très importante dans la littérature vu qu'elle permet de réduire la consommation d'énergie d'une manière significative et d'alléger le traitement des applications en ne leur fournissant que les données pertinentes. À notre connaissance, c'est la première fois qu'une telle approche voit le jour dans la littérature.

Cette nouvelle approche utilise la formation des grappes avec des contraintes liées aux exigences des applications et à la carte des énergies. Afin de résoudre le problème de la formation des grappes, nous avons conçu deux nouveaux algorithmes : un algorithme réparti et un autre centralisé. L'algorithme réparti utilise les capacités de traitement de tous les nœuds du réseau afin de former les grappes de proche en proche à partir du nœud collecteur. L'algorithme centralisé a été conçu pour pallier aux faiblesses de l'algorithme réparti qui ne permet pas d'optimiser la consommation d'énergie vu que les nœuds n'ont qu'une vision restreinte limitée aux nœuds voisins. Ainsi, nous avons étudié les approches centralisées existantes de la formation des grappes. L'analyse des lacunes de ces algorithmes nous a permis de concevoir un algorithme novateur de la formation des grappes. La particularité de cet algorithme réside dans le fait que ni les têtes de grappes ni le nombre de grappes ne sont prédéterminées. Ceci permet d'exploiter tous les scénarios possibles contrairement aux approches existantes. Le problème a été modélisé par un problème de partitionnement d'un hypergraphe qui est NP-Complet. Sa résolution utilise une adaptation intelligente de la *méta-heuristique la Recherche Taboue.*

Afin de mesurer ses performances, l'approche centralisée a été simulée à l'aide de l'outil OMNET++. Les résultats, comparés à ceux de l'approche existante TAG, montrent que notre approche réduit d'une manière significative la consommation d'énergie. L'approche centralisée a été implémentée à l'aide des outils de programmation C++, avec des bibliothèques spécialisées dans les traitements des graphes. Les résultas, comparés à ceux d'une deuxième résolution basée sur CPLEX et d'une troisième résolution basée sur la méta-heuristique Recuit Simulé, montrent que l'adaptation de la recherche taboue donne des grappes avec une meilleure qualité et que notre approche centralisée se comporte d'une manière efficace avec l'extensibilité des réseaux de capteurs sans fil. La comparaison des deux approches centralisée et répartie a révélé que l'approche centralisée est plus efficace

dans le cas où le temps d'exécution des requêtes est long. Dans le cas contraire, l'approche répartie est plus efficace.

Les principales contributions de cette thèse s'articulent autour de deux grands axes qui sont : la conception d'un mécanisme de la collecte des données basé sur la qualité de service et la carte des énergies dans les réseaux de capteurs et la proposition de nouvelles approches centralisées de la formation des grappes. En effet, nous avons conçu une nouvelle approche de la collecte des données qui combine les exigences des applications avec la carte des énergies afin de minimiser la consommation d'énergie par les nœuds et ne fournir aux applications que les données pertinentes. En ce qui a trait aux nouvelles approches centralisées de la formation des grappes, après avoir analysé les approches existantes les plus connues, il s'est avéré qu'aucune d'elles ne permet de former les grappes sans une phase préalable de la fixation des têtes de grappes. À notre connaissance, c'est la première fois dans la littérature que la formation des grappes et la détermination de leurs têtes sont jumelées dans une seule phase d'optimisation. Ce qui permet de former les grappes d'une manière efficace en prenant en considération les combinaisons les plus intéressantes sans être contraint par l'emplacement préalable des têtes de grappes.

ABSTRACT

Increasingly, several applications require the acquisition of data from the physical world in a reliable and automatic manner. This necessity implies the emergence of new kinds of networks which are typically composed of low-capacity devices. Such devices, called sensors, make it possible to capture and measure specific elements from the physical world (e.g. temperature, pressure, humidity). Generally, device capacities are highly limited. In fact, small batteries are used to provide the power required for them to function. Consequently, power consumption must be optimized in order to prolong the lifetime of these devices. This power optimization must be taken into account for all network layers, including the physical and applicative layers.

Initially, we have studied and analyzed the existing power consumption models for the sensor networks. This analysis revealed that the existing models are inadequate to design an efficient data collection mechanism. Indeed, the majority of these models do not take into account the node modes and they consider that the nodes are always in an active mode. This fact leaded us to devise a general mathematical model for the power consumption. This model allows defining the energy maps that combine the residual and the predicted energy of each node in the sensor network.

We also have studied and analyzed the existing data collection mechanisms. We have concluded from this analysis that these mechanisms do not reduce the power consumption while collecting sensor data due mainly to the fact that they consider only one sensor network aspect at the same time. As consequence, we have designed a novel data collection mechanism that combines the application requirements with the energy map of the sensor network to optimize the power consumption and to provide applications with the information they require without loading them with unnecessary data. To our best knowledge, this is the first time that such kind of approach sees the light.

This novel approach utilizes the node clusters building with certain specific constraints related to the application requirements and the network energy map. In order to build the clusters, we have designed two algorithms: a distributed and a centralized algorithm. The distributed approach uses each node treatment capacity to build the clusters. Each node decides by executing the algorithm whether to belong to an existing cluster or create a new cluster. The centralized approach was designed to overcome the drawbacks of the distributed

approach. In deed, the power consumption is not optimized since the information of each node is limited to its neighbours. Consequently, we studied the existing centralized cluster building approaches. From the drawbacks of these approaches, we designed a novel algorithm for cluster building in sensor networks. The particularity of this algorithm is that the number of clusters and cluster heads are unknown beforehand which helps the algorithm exploring multitude of solutions. We modeled the cluster building problem as a hyper graph partitioning. Since this problem is known as NP-hard, we opted for an adaptation of the Meta heuristic Taboo Search. The adaptation consists of defining three types of moves that allow reassigning nodes to clusters, selecting cluster heads and removing existing clusters. Such moves use the largest size clique in a feasibility graph which facilitates the analysis of several solutions and makes it possible to compare them using a gain function.

In order to evaluate the performances of the distributed approach, simulations were conducted to analyze node behavior using OMNET++ simulator. Compared to the TAG algorithm, the results of our approach show that this novel algorithm can reduce energy consumption in variety of networks using various network sizes and topologies. In order to evaluate the performance of the centralized approach, the algorithms were implemented using C++ programming language and graph libraries. The performance is compared to that obtained by a second resolution CPLEX-based method, a third approach based on Simulated Annealing heuristic and an existing algorithm (TAG). Finally, results show that a Taboo search-based resolution method provides quality solutions in terms of cluster cost and execution time. Furthermore, it behaves well with network extensibility. Nevertheless, compared to a distributed approach, this centralized approach suffers from a major drawback linked to the additional costs generated by communicating the network node information and the time required to solve an optimization problem.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES, ABRÉVIATIONS ET ACRONYMES

ACK	Acknowledgement
ALE	Application Level Events
API	Application Programming Interface,
ATMS	Authenticated Tracking and Monitoring System
BEE	Battery Energy Efficient
BGL	Boost Graph Library
CDMA	Code Division Multiple Access
CSMA	Carrier Sense Multiple Access
CTS	Clear To Send
EPC	Electronic Product Code
FCC	Federal Communications Commission
GPS	Global Positioning System
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrie, Science et Médical
NM	Nœud Médiateur
LCA	Linked Cluster Architecture
PDA	Personal Digital Assistant
QdS	Qualité de Service
RCSF	Réseau de capteurs sans fil
RFID	Radio Frequency Identification
RS	Recuit Simulé
RT	Recherche Taboue
RTS	Request To Send
SEM	Seuil des Erreurs de Mesures
SQL	Structured Query Language
TAG	Tiny Agregation
TDMA	Time Division Multiple Access
UWB	Ultra Wideband
WPAN	Wireless Personal Area Network

CHAPITRE I
INTRODUCTION

L'émergence d'une nouvelle génération d'applications qui nécessitent l'acquisition de données du monde physique d'une manière automatique et fiable a fait apparaître un nouveau type de réseaux qui utilisent des dispositifs dont la principale caractéristique est leurs faibles capacités. Ces dispositifs, appelés des *capteurs*, permettent généralement de prendre des mesures d'un phénomène (e.g. température, pression), de les stocker dans des mémoires internes pour une utilisation ultérieure, d'effectuer des traitements et de les communiquer à ses voisins jusqu'à ce que les données arrivent aux applications.

Les capacités de ces dispositifs sont très limitées. Ils utilisent en général des petites batteries qui leur fournissent l'énergie électrique permettant leur fonctionnement. Par conséquent, la consommation d'énergie doit être gérée d'une manière optimale afin d'assurer une durée de vie des systèmes de capteurs la plus longue possible. Cette optimisation d'énergie devra être prise en considération par toutes les couches des réseaux, depuis la couche physique jusqu'à la couche applicative. Un capteur, par exemple, qui n'intervient pas dans l'opération de la collecte des mesures devra se mettre dans un état de veille où il consommera un minimum d'énergie.

Les caractéristiques particulières de ces dispositifs rendent la majorité des définitions et des concepts des systèmes traditionnels inadéquate pour ces nouvelles générations de réseaux. Par exemple, la définition de la Qualité de Service devra prendre en considération ce chevauchement de rôles entre les couches d'un réseau de capteurs. En effet, vu que la couche applicative est aussi intéressée par la consommation d'énergie, elle devra être en mesure de spécifier ses requis en terme de qualité de données collectées. Un autre concept qui devra être revu est celui relié au rôle du réseau qui se chargera non seulement d'échanger les données entre les nœuds, mais aussi d'effectuer des traitements de données, e.g. exécuter des agrégations et des filtrages.

Ce chapitre d'introduction présente les définitions et concepts de base permettant d'exposer les éléments de la problématique. Ensuite, les différents objectifs de recherche

seront énoncés suivis de l'esquisse méthodologique. Par la suite, les principales contributions de la thèse seront présentées. Enfin, le plan d'ensemble de la thèse est décrit.

1.1. Définitions et concepts de base

Un *capteur* est défini comme étant un dispositif qui détecte soit une valeur absolue d'une quantité physique, soit un changement de la valeur de cette quantité et qui convertit la mesure en un signal d'entrée pour un instrument qui indique la mesure ou l'enregistre.

Les capteurs sont, pour la plupart des cas, électriques ou électroniques, mais d'autres types existent aussi, comme les capteurs biologiques. Les capteurs sont généralement classés selon deux critères :

- la grandeur mesurée : on parle dans ce cas des capteurs de pression, de températures, de forces, etc. ;
- le caractère de l'information fournie : on parle dans ce cas des capteurs logiques, analogiques et numériques.

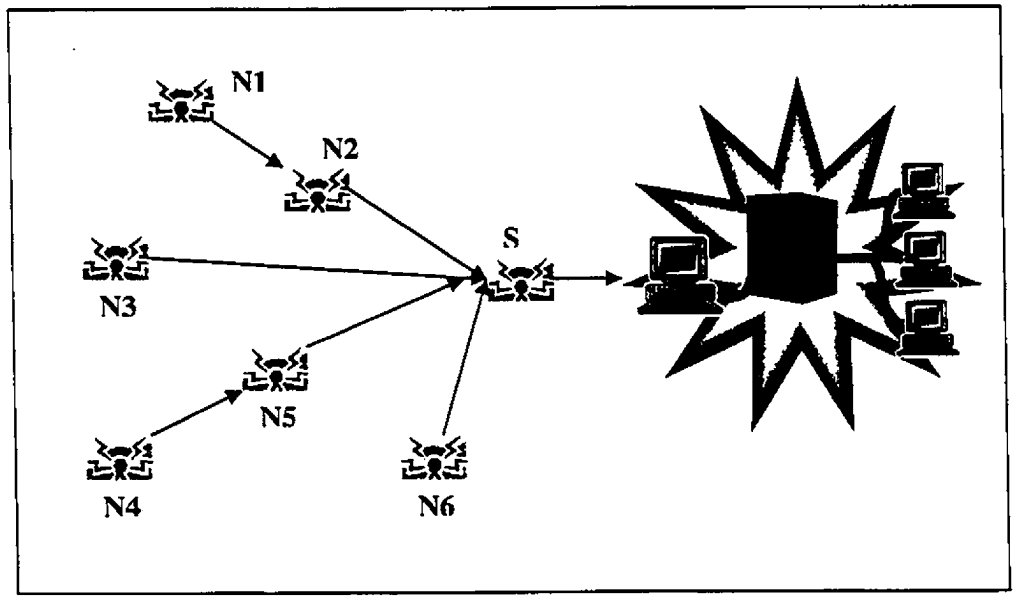

Figure 1.1 Exemple de réseau de capteurs

Un réseau de capteurs sans fil (**RCSF**) est un réseau constitué d'un grand nombre de petits *nœuds-capteurs* indépendants. Les nœuds-capteurs, dont la taille est typiquement égale à celle d'une boîte d'un film 35 mm, sont des unités qui consistent en une batterie, une antenne, un capteur et un minimum de capacité de calcul et de mémoire. Dans la suite de ce

document, les nœuds-capteurs seront désignés par « nœud » ou tout simplement par « capteur ». La Figure 1.1 illustre un exemple de réseau de capteurs. Les nœuds N_i captent des mesures et les acheminent au nœud S, appelé nœud *collecteur*. Le nœud collecteur a pour rôle de centraliser les données et de les communiquer à travers un autre type de réseau classique ou directement aux applications.

Les applications utilisant les réseaux de capteurs [CAL04], et qui sont caractérisées par une faible exigence en terme de débit (souvent mesuré en quelques bits par jour), couvrent plusieurs domaines incluant le contrôle et la surveillance industriels, l'automatisation des demeures, la sécurité militaire, le traçage des biens et la gestion de la chaîne d'approvisionnement, l'agriculture intelligente et enfin la surveillance de la santé des personnes.

Vu leur nature, les réseaux de capteurs présentent des particularités et des caractéristiques qui sont différentes de celles des autres types de réseaux sans fil (e.g. ad hoc) et filaires. Ces caractéristiques peuvent être subdivisées en plusieurs catégories, celles qui sont liées aux types d'applications utilisant ces réseaux, celles qui sont liées aux nœuds/capteurs, et celles qui sont liées au médium de communication et à la topologie du réseau. L'étude de ces caractéristiques est très importante pour concevoir d'une manière adéquate ces réseaux. Les principales caractéristiques de ces réseaux peuvent être résumées de la manière suivante : ressources limitées, sécurité, imprévisibilité, grande densité, temps réel, coopération, centré-données, orienté application et conçu pour de longues périodes.

Ressources limitées

Les capteurs ont généralement des capacités très limitées. D'une part, l'énergie fournie par des batteries généralement faibles risque de s'épuiser facilement si elle n'est pas utilisée d'une manière optimale. Il est nécessaire, par exemple, de gérer le compromis entre la puissance du signal et le nombre de nœuds avec lesquels un autre nœud peut communiquer, vu que les études récentes ont montré que la communication radio est l'opération la plus consommatrice d'énergie dans un réseau de capteurs.

D'autre part, la vitesse d'exécution du processeur du capteur et la taille de sa mémoire sont toujours très limitées et ne peuvent pas supporter des opérations complexes. Le dilemme est que les concepteurs de capteurs doivent garder le coût des dispositifs très faible vu qu'un réseau RCSF peut contenir un nombre très grand de nœuds.

Sécurité

Vu que les réseaux de capteurs seront déployés pour des applications qui sont critiques (e.g. applications militaires, incendies), l'aspect de la sécurité a une grande importance. Malheureusement, cet aspect est le plus difficile à satisfaire. En particulier, il est facile d'espionner les messages de réseaux et d'effectuer des attaques de déni de service. Le grand dilemme est que les algorithmes et les protocoles de la sécurité demandent un temps de calcul important et un trafic additionnel, alors que les ressources des capteurs sont limitées.

Imprévisibilité

Les environnements dans lesquels les réseaux RCSF seront amenés à opérer sont des environnements dont les états ne peuvent pas être prévisibles. Ceci est dû à plusieurs raisons:

- les réseaux de capteurs sont déployés dans des environnements physiques qui sont incontrôlables (e.g. le cas des tremblements de terre) ;
- la communication sans fil engendre des erreurs et des pertes de messages dues aux interférences des fréquences radio ;
- le calibrage des capteurs n'est pas toujours adéquat ;
- la topologie peut être dynamique et peut changer d'une manière continue.

Grande densité

Les réseaux de capteurs sont généralement déployés avec une grande densité de nœuds. Par conséquent, le nombre des erreurs et des pertes de messages sera amené à augmenter ainsi que les incertitudes des mesures prises par les capteurs. De ce fait et vu que les nœuds ne sont pas toujours répartis sur les lieux géographiques, les protocoles de routage des réseaux mobiles existants ne peuvent pas donner une performance satisfaisante s'ils sont appliqués sans aucune amélioration.

Temps réel

Les réseaux de capteurs opèrent dans un monde réel. Les contraintes temporelles sont donc très importantes. Ces contraintes peuvent être explicites (e.g. une personne qui rentre à la maison doit être reconnue d'une manière instantanée). Du fait que ces réseaux peuvent

avoir une grande taille, il est difficile de satisfaire les contraintes temporelles des réseaux de capteurs.

Coopération

Vu qu'un réseau de capteurs est considéré comme un cas particulier d'un réseau ad hoc et donc sans infrastructure, la coopération entre les différents nœuds est nécessaire pour le routage et l'acheminement des paquets. Pour conserver son énergie, un nœud peut choisir de ne pas acheminer les paquets de ses voisins ou se mettre dans un état de veille. Le réseau doit implémenter un mécanisme qui stimule la participation des nœuds dans la fonction de routage, et un mécanisme qui gère l'état des capteurs (e.g. en veille) pour pouvoir minimiser la consommation d'énergie des nœuds et maximiser la durée de vie du réseau en entier.

Centré-Données

Les réseaux traditionnels sont centrés-adresses. Les données sont communiquées entre des nœuds qui ont des adresses qui les identifient. Par contre, les réseaux de capteurs sont largement centrés-données. Les données qui émanent de plusieurs sources et qui sont liées à un seul phénomène doivent être agrégées et envoyées à la station de base, ce qui nécessite la conception de nouveaux protocoles qui sont centrés-données.

Orienté application

Contrairement aux autres réseaux qui sont conçus pour faire fonctionner une multitude de types d'applications, les réseaux de capteurs sont déployés pour un type d'application bien défini. Les données peuvent être collectées, agrégées selon les spécifications d'un type d'application et envoyées à un autre nœud, ou bien elles peuvent être traitées localement dans les nœuds du réseau. Ainsi, il est crucial d'effectuer l'agrégation lorsque les données d'un seul nœud ne sont pas très importantes.

Conçu pour de longues périodes

L'une des caractéristiques des systèmes de capteurs [LIU03] est qu'ils sont toujours conçus pour fonctionner durant de longues périodes dans des situations où l'intervention humaine directe est minimale (pendant des mois ou des années). Ceci implique des systèmes qui garantissent la fiabilité et la facilité de mise à jour durant la durée de vie des systèmes des capteurs.

1.2. Éléments de la problématique

Les réseaux de capteurs sont considérés comme une génération de réseaux différente des autres types de réseaux, vu les contraintes et les caractéristiques de leurs nœuds. La principale caractéristique et la plus contraignante est le fait que les nœuds ont généralement des capacités d'énergie très limitées, facilement épuisables et difficilement remplaçables. Par conséquent, les mécanismes et les manières de déploiement des nœuds devront considérer la contrainte de la consommation d'énergie. De ce fait, il est crucial de trouver un modèle de la consommation d'énergie des nœuds d'un réseau de capteurs. Ce modèle devrait être le plus proche possible de la réalité et devrait prendre en considération les différents états des nœuds.

Les travaux de recherche actuels [BAS04] [CAI05] [LEE04] se sont concentrés sur des modèles réduits de la consommation d'énergie pour étudier un phénomène en particulier. Basu *et al.* [BAS04] modélisent le coût de la consommation d'énergie des nœuds afin d'étudier l'impact de la sur-écoute des messages par les nœuds avoisinants le nœud qui envoie son message de capture. Leur modèle considère que les nœuds ont un seul statut, i.e. qu'ils sont toujours actifs et que les nœuds communiquent d'une manière directe sans qu'ils soient organisés dans des grappes ou des hiérarchies. Une autre faiblesse de ce modèle est qu'il considère que les nœuds ont tous les mêmes capacités.

Le modèle d'énergie de Caicedo et al. [CAI05] a pour objectif d'étudier l'impact du nombre des nœuds sur le coût de la consommation de l'énergie et de comparer la communication point à point avec la communication multicast. Leur modèle connaît les mêmes faiblesses que celles du modèle de Basu et al. mentionné précédemment.

Lee et al. [LEE04] modélisent la consommation d'énergie afin d'étudier l'impact du déploiement hétérogène sur la durée de vie de la couverture de la capture dans un réseau de capteurs. Dans leur modèle, ils considèrent que les nœuds restent toujours actifs, en train de capturer et de communiquer leurs messages. Ils prennent en compte les opérations d'agrégation qu'un nœud intermédiaire peut exécuter lors de la réception des messages d'autres nœuds, mais ils supposent que tous les nœuds ont la capacité d'exécuter l'agrégation.

En étudiant ces différents travaux de recherche (trois sont cités, mais la liste n'est pas exhaustive), il s'avère que les modèles de la consommation d'énergie ne reflètent pas la

réalité des réseaux de capteurs et ne considèrent pas un composant qui a une très grande importance, celui de l'état du nœud. Par exemple, les nœuds capteurs sont conçus physiquement pour qu'ils puissent se mettre en état de veille afin qu'ils économisent leur énergie lorsque le système n'a pas besoin de certains nœuds. D'où l'intérêt de trouver un modèle étendu de la consommation d'énergie qui intègre ce composant.

Ce modèle servira à concevoir un mécanisme de collecte de mesures qui optimise l'utilisation de la consommation d'énergie. L'importance de ce mécanisme réside dans le fait que les réseaux de capteurs sont considérés comme des bases de données réparties où chaque nœud contient une ou plusieurs données. Les applications seront toujours intéressées par la collecte de mesures captées par les nœuds du réseau. Ainsi, la manière selon laquelle ces données seront collectées a un impact direct sur la durée de vie du réseau, vu que la transmission des messages est l'opération qui consomme la plus grande partie de l'énergie.

L'une des techniques les plus utilisées pour la collecte de données qui optimisent la consommation d'énergie est l'agrégation. Madden et al. [MAD02] ont conçu une agrégation en réseau qui permet de calculer les données au fur et à mesure qu'elles arrivent aux nœuds. Ils construisent un arbre qui permet de hiérarchiser le réseau afin d'exécuter l'agrégation par les nœuds intermédiaires. L'une des faiblesses de leur méthode est qu'elle ne contient pas un mécanisme de filtrage. De plus, tous les nœuds parents calculent la fonction d'agrégation. Enfin, le mécanisme n'explicite pas la méthode adoptée pour construire l'arbre.

Yoon et al. proposent [YOO04] un algorithme de traitement des requêtes en réseau qui permet de réduire la consommation d'énergie en diminuant le nombre de messages échangés. Des grappes sont formées en fonction des mesures actuelles des nœuds et d'un cœfficient qui constitue l'incertitude des mesures afin de hiérarchiser la collecte des données. L'une des faiblesses de leur méthode est que les grappes sont constituées pour chaque requête. De plus, l'agrégation fait abstraction de la sémantique des nœuds qui ont envoyé leurs mesures. Enfin, le protocole ajoute des messages qui pourront être volumineux à cause de l'envoi des mesures des têtes de grappes.

D'où l'importance de concevoir un nouveau mécanisme qui essaye de pallier les faiblesses de ces techniques existantes et qui utilise le modèle de la consommation d'énergie le plus proche possible de la réalité.

1.3. Objectifs de recherche

L'objectif principal de cette thèse est de concevoir un mécanisme de collecte de données qui utilise un modèle de consommation d'énergie d'un réseau de capteurs proche de la réalité en prenant en considération les différents états d'un nœud (capture, communication et en veille). Plus spécifiquement, ce travail vise à :

- analyser les modèles de consommation d'énergie existants et les mécanismes de collecte de données en vue de les caractériser pour en déceler les points forts et les points faibles ;
- concevoir un mécanisme de collecte de données qui pallie les faiblesses des mécanismes existants et qui utilise la notion de grappes afin de réduire la consommation d'énergie tout en prenant en considération les requis des applications en terme de qualité de service ;
- formuler le problème de la formation de grappes et proposer des approches de la résolution de ce problème ;
- évaluer la performance des algorithmes et des différents mécanismes en utilisant une approche analytique supportée par des outils de simulation (OMNET, OPNET).

1.4. Esquisse méthodologique

Pour atteindre nos objectifs, nous devrons franchir les étapes suivantes :

- Dans un premier temps, nous analyserons les modèles de consommation d'énergie dans les réseaux de capteurs en mettant l'accent sur les méthodes de calcul de l'énergie consommée durant la capture des mesures, la transmission des messages et la réception des messages. L'intégration d'un modèle d'état des nœuds permettra de trouver une formule mathématique plus étendue de la consommation d'énergie, qui nous permettra d'estimer l'énergie consommée pendant une période de temps.
- Dans un deuxième temps, nous étudierons les principaux mécanismes de collecte des données. Les faiblesses de ces approches devront nous permettre de concevoir un nouveau mécanisme qui utilise les formules mathématiques

trouvées dans la première phase. Le but de ce mécanisme est de fournir aux applications les mesures requises tout en minimisant la consommation d'énergie et en maximisant la couverture du réseau de capteurs. Un deuxième objectif sera d'éviter l'encombrement des données reçues par les applications et ainsi alléger leurs traitements.

- Dans un troisième temps, nous proposerons une formulation mathématique de la formation de grappes. Deux approches de résolutions seront conçues. Une première approche répartie faisant intervenir tous les nœuds du réseau et une deuxième approche centralisée dont la résolution est basée sur des heuristiques.
- Finalement, nous utiliserons une approche analytique pour vérifier les formules mathématiques de la consommation d'énergie. Par la suite, nous évaluerons les performances du mécanisme de la collecte de données proposé en les comparant aux performances d'une ou de plusieurs méthodes de collecte existantes. Ces analyses de performance se feront en utilisant des simulateurs reconnus par la communauté scientifique dans ce type de réseau et système. Une comparaison des deux approches répartie et centralisée sera effectuée afin d'étudier la qualité des solutions trouvées par les méthodes heuristiques.

1.5. Principales contributions et originalité

Les principales contributions de cette thèse s'articulent autour de deux grands axes qui sont : la conception d'un mécanisme de la collecte des données basé sur la qualité de service et la carte des énergies dans les réseaux de capteurs et la proposition de nouvelles approches centralisées de la formation des grappes. Ces deux principales contributions aident à la résolution de deux problématiques qui constituent les défis les plus importants pour les systèmes de réseaux de capteurs et qui empêchent leur déploiement à grande échelle. En effet, la manière avec laquelle les données sont collectées et la manière avec laquelle les grappes sont formées ont un impact direct sur la durée de vie des capteurs. Ces deux problèmes ont été traités par plusieurs chercheurs mais aucun d'eux n'a pu proposer des approches efficaces en terme d'optimisation de la consommation d'énergie dû d'une part, à

la complexité de ces problèmes qui font intervenir plusieurs facteurs, et d'autre part, à la négligence de l'importance des applications dans les mécanismes de la collecte des données et la détermination des têtes de grappes avant la formation de ces dernières.

De manière plus spécifique, nous avons conçu une nouvelle approche de la collecte des données qui combine les exigences des applications avec la carte des énergies afin de minimiser la consommation d'énergie par les nœuds et ne fournir aux applications que les données pertinentes. En effet, la combinaison de ces deux composants constitue notre première contribution. Pour cela, un certain nombre d'apports secondaires a été réalisé. En effet, nous avons donné l'expression générale de la prédiction de la consommation d'énergie. En plus, nous avons apporté une nouvelle définition de la qualité de service dans les réseaux de capteurs et nous avons proposé une séparation entre les mécanismes de filtrage et ceux de l'agrégation. Enfin, nous avons explicité une nouvelle formulation de la formation de grappes basée sur ces critères (i.e. la qualité de service et la carte des énergies). À notre connaissance, c'est la première fois que les requis des applications ont été pris en considération dans la formulation de la formation des grappes, contrairement aux approches existantes qui ne considèrent que les facteurs liés aux capteurs.

En ce qui a trait aux nouvelles approches centralisées de la formation des grappes, après avoir analysé les approches existantes les plus connues, il s'est avéré qu'aucune d'elles ne permet de former les grappes sans une phase préalable de la fixation des têtes de grappes. À notre connaissance, c'est la première fois dans la littérature que la formation des grappes et la détermination de leurs têtes sont jumelées dans une seule phase d'optimisation. Ce qui permet de former les grappes d'une manière efficace en prenant en considération les combinaisons les plus intéressantes sans être contraint par l'emplacement préalable des têtes de grappes. La méthode de la résolution de ce problème, connu comme étant un problème NP-complet, est novatrice. En effet, l'adaptation de la méta-heuristique recherche taboue est basée sur la construction d'un nouveau graphe et la recherche des grappes réalisables comme étant des cliques. Une deuxième résolution basée sur une méthode exacte est proposée afin d'analyser la qualité des solutions trouvées par la première approche de résolution. C'est une nouvelle résolution qui consiste à réduire l'espace des solutions, trouver un recouvrement au lieu d'un partitionnement et exécuter une phase post optimisation pour raffiner la solution trouvée par la méthode exacte.

1.6. Plan de la thèse

Cette thèse est répartie sur six chapitres. Le chapitre suivant présente les différents aspects des couches réseaux et une analyse des différents modèles de la consommation de l'énergie rencontrés dans la littérature. Le chapitre 3 décrit les approches existantes de la collecte des données et notre nouvelle approche répartie. Le chapitre 4 présente une deuxième approche centralisée de la formation des grappes tout en détaillant deux manières de résoudre le problème. Le chapitre 5 présente l'analyse de performance des mécanismes proposés. Enfin, le chapitre 6 résume les principales contributions des travaux accomplis et présente les limitations et les extensions potentielles à la thèse.

CHAPITRE II
MODÈLE DE LA CONSOMMATION D'ÉNERGIE

Les sujets liés aux réseaux de capteurs commencent à avoir de l'importance vu l'intérêt qu'ils suscitent dans les applications de monde réel. Plusieurs applications sont conçues sur la base des services fournis par ce type émergent de réseaux. Cependant, les réseaux de capteurs présentent des défis dans toutes les couches réseaux. De nombreux travaux de recherche ont vu le jour traitant une partie des ces défis. Un intérêt particulier a été donné aux modèles de la consommation d'énergie. En effet, une grande majorité de ces études ont besoin de ces modèles pour comprendre la manière avec laquelle les nœuds d'un réseau de capteurs consomment de l'énergie pour pouvoir optimiser son utilisation.

Ainsi, l'analyse des modèles de la consommation d'énergie permet de comprendre les facteurs qui influencent le plus cette consommation et de déceler les lacunes de ces approches. Dans ce chapitre, nous allons tout d'abord introduire quelques applications utilisant les réseaux de capteurs pour pouvoir faire ressortir l'importance de ce type de réseau. Puis, nous décrirons les fonctions de chaque couche réseau et ses défis. Enfin, nous présenterons les modèles de la consommation d'énergie.

2.1. Les applications des réseaux de capteurs

Les applications utilisant les réseaux [CAL04] de capteurs et qui sont caractérisées par une faible exigence en terme de débit (souvent mesurée en quelques bits par jours), couvrent plusieurs domaines incluant le contrôle et la surveillance industrielle, l'automatisation des demeures, la sécurité militaire, le traçage des biens et la gestion de la chaîne d'approvisionnement, l'agriculture intelligente, et enfin la surveillance de la santé des personnes.

2.1.1. Le contrôle et la surveillance industriels

Un exemple d'application est l'utilisation des réseaux de capteurs pour la sûreté industrielle [CAL04]. Un système de capteurs sans fil peut utiliser les capteurs pour détecter la présence des produits nocifs ou dangereux afin de prévenir de sérieux dégâts que des

agents biologiques ou chimiques peuvent causer. Du fait que les réseaux de capteurs sans fil utilisent des algorithmes répartis et un routage sur différents chemins, ils s'avèrent très efficaces dans le cas des explosions ou des accidents industriels en fournissant aux responsables des informations critiques dans des environnements difficiles.

2.1.2. L'automatisation des maisons et les produits électroniques

Une des applications est la télécommande universelle, qui peut être un dispositif (PDA) qui contrôle non seulement la télévision, le lecteur DVD, la chaîne audio et les autres équipements électroniques mais aussi les lumières, les rideaux et les serrures qui sont équipés des capteurs reliés avec un réseau de capteurs sans fil [CAL04]. Le contrôle de tous ces équipements peut se faire manuellement par une personne à partir de sa chaise, ou bien automatiquement, par exemple le rideau qui se ferme quand la télévision s'allume ou le volume de la chaîne audio qui s'abaisse lors d'un appel téléphonique.

2.1.3. Sécurité et applications militaires

Comme la plupart des technologies, les premières applications proposées pour les réseaux de capteurs sans fil étaient des applications militaires [CAL04]. L'un des avantages les plus importants de l'utilisation des réseaux de capteurs sans fil est qu'ils permettent de remplacer les gardes et les sentinelles dans les périmètres défensifs, prévenant ainsi les soldats contre toutes attaques imprévues. Ils peuvent aussi servir comme mines anti personnelles, prévenant ainsi les dégâts que ces mines font après les guerres. Ils peuvent aussi être utiles pour localiser les troupes ennemies et les troupes amies.

2.1.4. Le traçage des biens et gestion de la chaîne d'approvisionnement

Ce champ d'applications est prévu être le plus envahi par ces types de réseaux [CAL04]. L'utilisation des réseaux de capteurs sans fil pour tracer des produits nucléaires a été déjà démontrée dans le projet ATMS (Authenticated Tracking and Monitoring System). Ce projet utilise des capteurs sans fil (incluant l'état des portes, la température et les radiations) dans les conteneurs de transport. La notification des événements des capteurs est transmise à travers une liaison sans fil à une unité mobile connectée à un système de GPS. Ce dernier satellite fournit le lieu du conteneur et l'état de son contenu.

2.1.5. L'agriculture intelligente

La mesure de l'humidité de sol n'est pas la seule application des réseaux de capteurs sans fil dans le champ des applications de l'agriculture, parce que ces réseaux peuvent contenir une variété de capteurs biologiques et chimiques [CAL04]. Les données fournies par ces réseaux peuvent informer le fermier de l'état de l'humidité de son sol, la température, les besoins en pesticides, herbicides, les fertilisants, et bien d'autres quantités importantes. Ce type d'applications est crucial pour les vignes vu qu'un petit changement de l'environnement peut avoir un impact très important.

2.1.6. La surveillance de la santé des personnes

Deux classes générales sont définies pour les applications des réseaux de capteurs sans fil dans le domaine de la surveillance de la santé des personnes [CAL04]. La première classe est celle de la surveillance des performances des athlètes, par exemple, tracer les battements de cœurs et le taux de la respiration et les envoyer à un ordinateur personnel afin de les analyser. La deuxième classe d'applications est celle de la surveillance des malades dans leurs demeures, par exemple, mesurer le poids d'un patient et l'envoyer à son ordinateur personnel pour le stocker. D'autres exemples peuvent être imaginés incluant la surveillance du taux de sucre dans le sang pour les diabètes et la surveillance à distance des malades chroniques.

2.2. Les couches réseaux des réseaux de capteurs sans fil et leurs défis

Les défis des couches réseaux des réseaux de capteurs sans fil présentent des particularités liées à la nature de ces réseaux. Dans les sections suivantes, nous allons analyser les différentes couches réseaux et leurs défis respectifs.

2.2.1. La couche physique

La première caractéristique à prendre en considération durant la conception de la couche physique est le coût. Le coût de la couche physique affecte d'une manière significative le coût des nœuds en raison du coût du matériel qu'il fait intervenir. La couche physique doit être compatible avec les réglementations de la plupart des pays. Comme la majorité des bandes ISM, la bande 2.4 Ghz est occupée par d'autres services tels que

Bluetooth et WPAN. Une alternative qui devient possible est l'utilisation de la bande ultra large (UWB) définie par FCC (Federal Communications Commission). A présent, cette couche physique ne peut être utilisée que dans les É-U.

La deuxième importante caractéristique à prendre en considération durant la conception de la couche physique est la consommation d'énergie. Deux aspects du problème de l'énergie doivent être considérés: le comportement de la source d'énergie et la consommation de l'énergie par les nœuds. En effet, vu que la consommation de l'énergie par les nœuds est supposée être faible, de l'ordre de 50 uW ou moins, les sources d'énergie non conventionnelles peuvent s'avérer des alternatives plausibles (e.g. Energie solaire, vibration mécanique), mais ces sources ont leurs limitations.

Parmi les couches physiques les plus connues, on peut citer :

- *Bluetooth* : Opère dans la bande ISM avec une fréquence de 2.4 Ghz. Cette couche n'est pas appropriée pour les réseaux de capteurs vu que la manière dont elle fonctionne pour trouver les nœuds de réseaux consomme une grande quantité d'énergie ;
- *IEEE 802.11b* : Opère avec une fréquence de 2.4 Ghz. C'est une technologie qui peut s'avérer un candidat approprié pour ce type de réseau vu que les exigences matérielles sont simples ;
- *PicoRadio* : Rabaey et al., les concepteurs de cette couche, trouvent que les systèmes à bande ultra large sont attractifs pour les réseaux de capteurs sans fil [RAB00] car la facilité d'intégration (e.g. coût faible) est prioritaire à l'efficacité d'utilisation de la largeur de bande dans le cas des réseaux de capteurs.

Cette couche présente beaucoup de défis pour les réseaux de capteurs sans fil. Les problèmes ouverts touchent:

- La modulation qui doit être simple et à faible consommation d'énergie ;
- La stratégie de contourner l'effet de la propagation de signal ;
- La conception des composants électroniques. Les capteurs doivent avoir une petite taille et un coût très réduit vu que les réseaux de capteurs sont supposés fonctionner avec un très grand nombre de noeuds.

2.2.2. La couche liaison de données

Les principales techniques d'accès au medium existantes sont :

ALOHA : Il est décrit comme étant le premier système pour les communications sans fil. Il emploie l'accès aléatoire. Il utilise la technologie étoile, qui consiste en un ordinateur central et plusieurs sites distants et emploie deux canaux physiques : l'un pour les communications entrantes et l'autre pour les communications sortantes. La collision des paquets est gérée en retransmettant les paquets après un temps aléatoire d'attente.

Les réseaux de capteurs sans fil ont un faible débit et un faible rapport de puissance signal/bruit pour conserver la durée de vie des batteries. Par conséquent, ALOHA peut s'avérer pertinente pour ce type de réseau, mais le fait qu'il exige l'utilisation d'un nœud maître qui doit recevoir les paquets d'une manière continue rend cette technologie incompatible avec le fait que ce type de réseau optimise l'utilisation de l'énergie pour tous les nœuds.

CSMA (Carrier Sense Multiple Access) : Sa forme la plus simple est CSMA non persistante qui fonctionne de la manière suivante : si le canal est libre, les paquets sont transmis. S'il est occupé, le nœud reporte la transmission avec un temps déterminé d'une manière aléatoire. CSMA soufre du problème du '*terminal caché*' et du problème du '*terminal exposé*'. Ces problèmes réduisent la capacité du canal. La difficulté majeure d'appliquer CSMA au réseau de capteurs sans fil est le temps requis pour écouter le canal avant la transmission.

Polling : Cette technologie fonctionne de la manière suivante : un nœud ne peut transmettre dans le canal que s'il reçoit une permission d'un nœud maître dans le réseau. Le nœud maître envoie régulièrement et individuellement une invitation aux autres nœuds (esclaves) leurs demandant s'ils ont des paquets à transmettre. La plus grande utilisation du Polling dans les réseaux WPAN est celle du Bluetooth. Les avantages de *Polling* par rapport à CSMA peuvent être résumés en trois points importants:

1. Le temps d'accès au canal peut être déterministe et ne souffre pas d'un temps aléatoire comme c'est le cas pour CSMA ;
2. La stratégie d'accès peut s'ajuster facilement (ne fait intervenir que le nœud maître) et ainsi fournir une certaine qualité de service ;

3. Le problème du 'terminal caché' est évité.

Quant à ses désavantages par rapport à son utilisation dans les réseaux de capteurs sans fil, ils peuvent se résumer en trois points importants :

1. Le nœud maître sera plus chargé que les autres nœuds et ainsi son énergie s'épuisera facilement ;
2. Le temps de faire le polling augmente avec le nombre de nœuds dans le réseau ;
3. Tous les nœuds esclaves doivent être dans la zone du nœud maître pour qu'ils aient accès au canal.

Les techniques d'accès dans les réseaux de capteurs sans fil

PicoRadio : C'est un protocole d'accès au médium qui assigne 30 codes CDMA de manière à ce que les nœuds voisins de chaque nœud aient un code distinct. Tous les nœuds sont asynchrones vu l'absence d'un temps global qui permettra de faire fonctionner les nœuds dans un mode où ils sont inactifs. Par conséquent, tous les nœuds gardent leurs receveurs toujours actifs. Pour réduire la consommation d'énergie, Zhong propose une onde radio de réveil 'wake-up radio' dont la puissance est très faible, de l'ordre de 1 microwatt [ZHO01]. Son seul objectif est d'écouter le canal en tout temps et d'activer le receveur principal dans le cas où le nœud reçoit un message.

Le nœud médiateur (NM) :

Callaway propose un protocole d'accès au médium basé sur le principe de médiation entre les noeuds [CAL04]. Un nœud de réseau, appelé nœud médiateur (NM), agit comme un médiateur entre les différents nœuds du réseau, et il est en mesure d'enregistrer les messages de contrôle et de les rejouer. La Figure 2.1 illustre ce protocole.

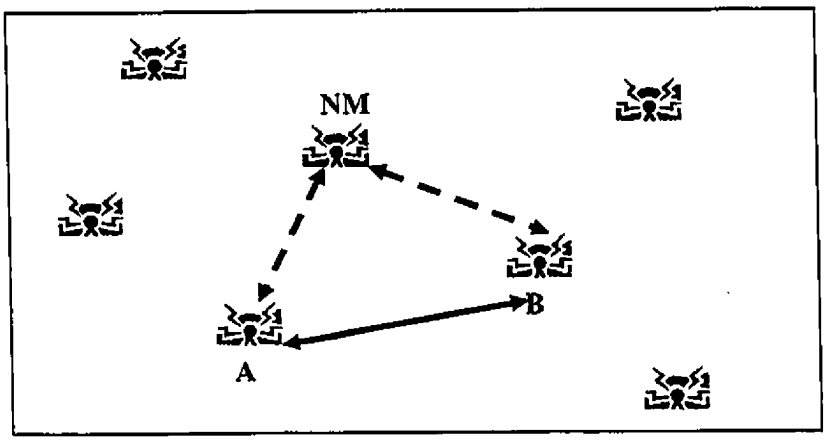

Figure 2.1 Protocole du Nœud Médiateur

Dans cette figure, le nœud A veut envoyer un message au nœud B. Les étapes de transfert sont:

1- Le nœud A commence à envoyer une série de messages RTS et attend un message de réponse RTS. Les deux nœuds A et B sont asynchrones. Le nœud NM enregistre le message RTS de A vu que le nœud B est supposé être dans un état où il ne peut ni recevoir ni transmettre des messages pour conserver son énergie ;

2- Quand le nœud B se réveille, il envoie une requête au nœud NM pour le consulter à propos des messages qu'il a reçus pendant sa période de veille. Le nœud NM peut calculer et enregistrer le temps de décalage entre A et B ;

3- Quand le nœud B reçoit une réponse de sa requête, il ajuste son temps et se synchronise avec le nœud A. Les deux nœuds peuvent maintenant commencer à échanger des messages.

Les échanges entre les nœuds A, B et NM sont présentés à la Figure 2.2.

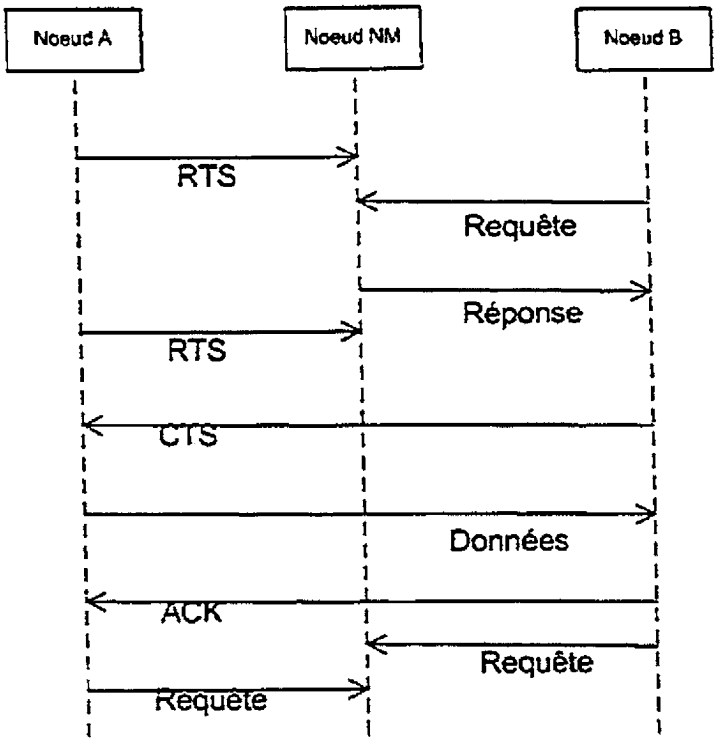

Figure 2.2 Échange de messages dans le protocole Nœud Médiateur

Les acronymes utilisés à la Figure 2.2 sont : (RTS: Request To Send – CTS: Clear To Send- ACK: Acknowledgement).

Le protocole Nœud Médiateur distribué

Dans ce protocole, la fonction du nœud médiateur est répartie sur les différents nœuds du réseau. Chaque nœud opère comme un nœud NM pendant une période de temps choisie par un processus aléatoire indépendant (i.e. un nœud commence à opérer comme NM à un temps aléatoire indépendamment des autres nœuds). Dans ce protocole, il n'est pas nécessaire d'avoir des nœuds dédiés pour la fonction de la médiation pour se prévenir contre le problème de partitionnement du réseau. En plus, la consommation de l'énergie est réduite vu qu'un nœud n'opère dans le mode NM que pendant une période limitée.

2.2.3. La couche réseau

Dans les réseaux de capteurs, la couche réseau doit traiter deux aspects qui sont reliés entre eux : la topologie du réseau et l'algorithme de routage. En ce qui concerne la topologie,

dans un réseau complètement plat sans structure logique, tous les nœuds doivent coopérer pour contrôler le réseau. Ce contrôle implique la génération des messages additionnels dont la transmission consomme l'énergie des nœuds. Dans le cas où le réseau est organisé en un ensemble de grappes de nœuds, chaque nœud est relié à au moins une grappe. Chaque grappe a une tête de grappe qui agit comme un contrôleur des nœuds qui appartiennent à cette grappe. Dans un LCA (Linked Cluster Architecture) qui emploie la TDMA comme protocole d'accès au médium, les nœuds suspendent la transmission des données pour exécuter un algorithme de formation de grappes réparti. Vu que chaque nœud n'enregistre et ne maintient que les informations qui sont reliées à son environnement immédiat, la LCA s'adapte bien avec la taille du réseau. L'inconvénient majeur de ce protocole et qui peut empêcher son utilisation aux réseaux de capteurs sans fil est le fait qu'il utilise une horloge globale avec laquelle tous les nœuds devront se synchroniser [CAL04].

Le deuxième aspect traité dans la couche réseau est lié aux protocoles de routage. Vu que les nœuds dans les réseaux de capteurs sans fil sont considérés comme quasi-stationnaires, leurs protocoles de routage peuvent éviter la surcharge liée à la gestion de la mobilité et réduire la consommation d'énergie par conséquent. Un exemple de protocole de routage dans ces types de réseaux est le protocole appelé *GRAd* (*Gradient Routage Protocole*) [POO01], qui fonctionne de la manière suivante : Les nœuds maintiennent une table contenant les coûts d'envoi des messages aux différentes destinations. Le routage des messages est basé sur le calcul du chemin des coûts les plus faibles.

L'un des principes les plus importants utilisé pour définir le routage est que pour maximiser la durée de vie du réseau, le trafic doit être routé en balançant la consommation d'énergie sur tous les nœuds du réseau suivant leurs réserves d'énergie au lieu de minimiser la consommation d'énergie absolue de tout le réseau. Ce principe est utilisé dans le protocole BEE (Battery Energy Efficient) [CHI00]. BEE assigne à chaque route un coût en fonction de la consommation d'énergie des nœuds et du comportement des batteries (Recouverture de la charge par exemple).

Les opérations et le routage dans les réseaux de capteurs sans fil représentent des défis aux concepteurs, vu que le facteur d'utilisation des nœuds doit être toujours maintenu à des valeurs faibles. Cette exigence limite la surcharge permise pour la synchronisation, la négociation, la coordination et les activités des autres couches de réseau.

Des protocoles de routage multi-hop doivent être conçus pour pouvoir acheminer les messages entre les nœuds de réseau et la station de base. Ces protocoles doivent utiliser efficacement l'énergie des noeuds et prendre en considération l'agrégation des données. Ils doivent aussi prendre en considération la topologie changeante et l'extensibilité du réseau. Due au problème d'énergie, la maximisation de la durée de vie du réseau doit aussi être prise en compte.

2.2.4. La couche applicative

Selon Y. Yau et al. [YAU04], la conception des applications pour les réseaux de capteurs sans fil doit prendre en considération certains principes de base :

1- Un intergiciel doit fournir des mécanismes centrés sur les données pour l'interrogation et le traitement des données dans le réseau. Vu sa simplicité et son efficacité, l'architecture basée sur des grappes est largement utilisée. Intuitivement, cette architecture est adéquate pour implémenter le paradigme centré sur les données ;

2- Le savoir d'application peut être utilisé pour faire adapter la conception et l'implémentation des logiciels aux caractéristiques des capteurs ;

3- Des algorithmes localisés doivent être utilisés pour achever collectivement l'objectif global tout en offrant un système extensible et robuste. Un intergiciel peut opérer comme un logiciel réparti composé de plusieurs grappes.

4- L'application doit être conçue en prenant en considération les capacités réduites des nœuds de réseau. Elle doit optimiser le nombre de messages communiqués et minimiser le temps de calcul.

Des services et des couches intermédiaires doivent être développés en prenant en compte la nature de ces réseaux. Les requêtes doivent inclure la localisation des noeuds à la place de leurs adresses et identités. Des protocoles et des services de synchronisation des temps doivent être conçus pour pouvoir simplifier les protocoles de communications entre les noeuds.

2.3. Modèles de la consommation d'énergie

Le modèle de la consommation d'énergie détermine la durée de vie du dispositif. Il est influencé par les types d'applications qui utilisent le réseau, le modèle de l'extraction des données par le réseau ainsi que le modèle de la communication.

2.3.1. Modèle de base de la consommation d'énergie

Lee *et al.* [LEE04] expriment la consommation d'énergie dans un cycle de la manière suivante :

$$E_{cycle} = E_D + E_S + E_T + E_R \qquad (2.1)$$

E_D : l'énergie requise pour le traitement des opérations de l'application ;

E_S : l'énergie requise pour capturer les données ;

E_T : l'énergie requise pour transmettre les données ;

E_R : l'énergie requise pour recevoir les données.

L'énergie consommée dans chaque tâche dépendra du réseau et du modèle des événements. La relation (2.1) s'exprime selon le modèle de la communication considéré [LEE04] ('single hop' ou 'multi hop'). Dans le cas de la communication 'single hop', chaque nœud ajuste son intervalle d'émission radio de telle sorte que ses paquets puissent arriver directement au nœud *collecteur* le plus proche de sa localisation. La relation (2.1) devient pour un nœud situé à une distance d du nœud *collecteur* :

$$E_{cycle}(d) = E_1 + E_2 d^2 \qquad (2.2)$$

La composante E_1 représente l'énergie requise pour le traitement des données (E_D), la capture des données (E_S) et une partie de la communication qui ne dépend pas de l'intervalle de l'émission radio. E_2 est le coefficient d'amplification qui dépend de la distance du nœud par rapport au nœud *collecteur*, et le degré de l'atténuation de la propagation de signal est supposé égal à 2.

Dans le cas de la communication multi hop, un nœud utilise un ensemble de nœuds intermédiaires afin d'acheminer ses données au nœud *collecteur*. Les énergies E_D, E_T et E_R seront déterminées principalement par le nombre de paquets transmis et le degré d'agrégation des données par les nœuds intermédiaires.

2.3.2. Impact de la sur-écoute des messages sur le modèle de la consommation d'énergie

Étant donné la nature du canal de transmission radio, les messages sont toujours écoutés par les nœuds avoisinant le nœud qui capture la donnée et l'envoie, même dans le cas où ces nœuds ne sont pas intéressés par ce message. Ce phénomène est connu comme la

sur-écoute des messages [SIN98]. Basu *et al.* [BAS04] ont étudié l'impact de ce phénomène sur la consommation d'énergie durant la capture et l'envoi des données.

Un réseau de capteurs statique peut être modélisé par un ensemble V de nœuds qui ont des emplacements géographiques prédéterminés [BAS04]. L'énergie requise pour transmettre un bit de données du nœud u vers le nœud v dans un canal sans fil dépend du degré de l'atténuation de la propagation de signal sur la distance d_{uv} qui les séparent [BAS04]. Elle est exprimée par la relation :

$$E_{uv}^{envoie} = E_{txelec} + \varepsilon_{amp} d_{uv}^{\alpha} \qquad \alpha \geq 2.0 \tag{2.3}$$

α est le degré d'atténuation de la propagation de signal et dépend du canal de transmission et des conditions de l'environnement. E_{txlec} est l'énergie dissipée sur le transmetteur électronique (par bit) et ε_{amp} est une constante qui caractérise l'amplification du transmetteur. L'énergie requise pour recevoir un bit de données dépend de l'électronique du receveur, et est donnée par :

$$E_{uv}^{reception} = E_{rxelec} \tag{2.4}$$

L'énergie consommée lors de l'envoi et de la réception entre deux nœuds u et v est :

$$\begin{aligned} E_{uv} &= E_{uv}^{envoie} + E_{uv}^{reception} \\ &= E_{txelec} + E_{txelec} + \varepsilon_{amp} d_{uv}^{\alpha} \end{aligned} \tag{2.5}$$

Quand un nœud u transmet des paquets de données à un autre nœud v en utilisant une puissance de transmission optimale, tous les nœuds dont la distance d est inférieure à la distance d_{uv} reçoivent ces paquets inutilement (phénomène de sur-écoute), et par conséquent, la consommation d'énergie augmente. Ceci est vrai spécialement dans le cas d'un réseau dense où les nœuds ont un grand nombre de nœuds voisins.

En prenant en considération ce facteur, la relation (2.5) devient :

$$\begin{aligned} E_{uv} &= E_{uv}^{envoie} + E_{uv}^{reception} + E_{uv}^{sur-écoute} \\ E_{uv} &= E_{txelec} + \varepsilon_{amp} d_{uv}^{\alpha} + N_{uv}^{se} E_{rxelec} \end{aligned} \tag{2.6}$$

N_{uv}^{se} est le nombre de nœuds avoisinant le nœud u et qui reçoivent les messages du nœud u à destination du nœud v. Généralement, la consommation d'énergie est symétrique, mais en considérant cette dernière équation, le coût de la consommation d'énergie devient asymétrique i.e. $E_{uv} \neq E_{vu}$.

À partir de la relation (2.6) qui calcule l'énergie consommée par la transmission d'un bit entre deux nœuds adjacents, l'énergie utilisée par l'envoi d'un bit par un nœud *l* au nœud *collecteur*, appelé *s*, dans un modèle de réseau multi-hop peut être déduite de la manière suivante :

$$E_{ls} = hop_{ls} E_{txelec} + \sum_{i=l}^{s} \left(\varepsilon_{amp} d_{i,i+1}^{\alpha} + N_{i,i+1}^{sc} E_{rxelec} \right) \qquad (2.7)$$

hop_{ls} est le nombre de nœuds qui séparent le nœud *l* du nœud *collecteur*. Cet ensemble de nœuds présente la route que le bit emprunte lors de sa transmission du nœud *l* vers le nœud *s*. $d_{i,i+1}$ est la distance entre deux nœuds successifs dans cette route. $N_{i,i+1}$ est le nombre de nœuds voisins qui pourront sur-écouter les messages envoyés du nœud *i* vers le nœud *i+1*.

2.3.3. Résultats expérimentaux de la consommation d'énergie dans les réseaux Ad-Hoc

Feeney *et al.* [FEE01] ont mené des expériences pour étudier la consommation d'énergie par les nœuds d'un réseau Ad hoc. Leurs résultats peuvent être considérés dans le cas des nœuds d'un réseau de capteurs vu que le médium de communication est le même, que leurs études portent sur le médium et ne font pas intervenir les caractéristiques physiques des nœuds du réseau. De plus, leurs résultats semblent être logiques et applicables aussi dans le cas des réseaux de capteurs.

D'après les études de Feeney *et al.* [FEE01], l'interface réseau opérant dans un mode ad hoc ne dort pas. Mais il y a toujours une consommation d'énergie qui permet d'écouter le canal sans fil. Cette consommation d'énergie ne diffère pas beaucoup de celle utilisée lors de la réception des messages. Dans des cas simples, l'énergie consommée par une interface réseau lorsqu'un nœud envoie, reçoit ou rejette un paquet peut être décrite par une équation linéaire :

$$Énergie = m \times taille + b \qquad (2.8)$$

Il est trivial qu'il y a une partie d'énergie fixe généralement associée au changement de l'état de nœud, et une partie variable qui dépend de la taille des paquets [FEE01]. Ce modèle d'équation linéaire est confirmé par des expériences qui permettent de déterminer les différents cœfficients pour les différents modes d'opérations des nœuds.

À travers ces expériences, Feeney *et al.* [FEE01] réalisent que l'énergie consommée par la réception des paquets est légèrement supérieure en la comparant avec celle consommée lorsque le nœud attend des paquets.

2.3.4. Modèle des états des nœuds et carte d'énergie

La conservation d'énergie est le volet prépondérant durant la conception des réseaux de capteurs. La meilleure manière de satisfaire cette contrainte est de rendre des dispositifs inactifs pendant les périodes où ils ne sont pas utilisés. Dans certains environnements d'exécution, un nœud peut opérer dans différents modes avec différents niveaux d'activation et ainsi différents niveaux de consommation d'énergie [MIN04]. Le nœud aura tendance à faire basculer son état à un mode où il consommera moins d'énergie aussitôt que cela est possible.

Mini *et al.* [MIN04] proposent un modèle qui décrit ces différents modes d'opération d'un nœud. Dans ce modèle, chaque nœud a quatre modes d'opération :

- État 1 : Capture inactive et radio inactive, appelé état *dort* ;
- État 2 : Capture active et radio inactive, appelé état *capte* ;
- État 3 : Capture active et radio recevant, appelé état *capte-reçoit* ;
- État 4 : Capture active et radio émettant, appelé état *capte-transmet.*

Le comportement d'un nœud est décrit à la Figure 2.3. Au début (déploiement ou simulation) du système, chaque nœud rentre dans l'état *dort* avec la probabilité (*proba_ dormir*) ou dans l'état *capte* avec une probabilité (*1- proba_dormir*). Quand un nœud rentre dans l'état *dort*, il va dormir pendant une période de *temps_dormir* secondes. Durant cette période, le nœud ne peut ni recevoir des messages ni capturer des événements. Après *temps_dormir* secondes, le nœud rentre dans l'état *capte-reçoit* pour pouvoir vérifier s'il existe un événement qui lui est destiné ou un autre nœud tentant de communiquer avec lui. Si le nœud trouve un événement, il rentre dans les états *dort*, *capte*, *capte-reçoit* ou *capte-transmet* avec des probabilités *proba_état1*, *proba_état2*, *proba_état3* et *proba_état4* respectivement. S'il n'y a pas d'événement, le nœud rentre dans l'état *dort* avec la probabilité *dormir_proba* ou dans l'état *capte* avec une probabilité (*1-dormir_proba*).

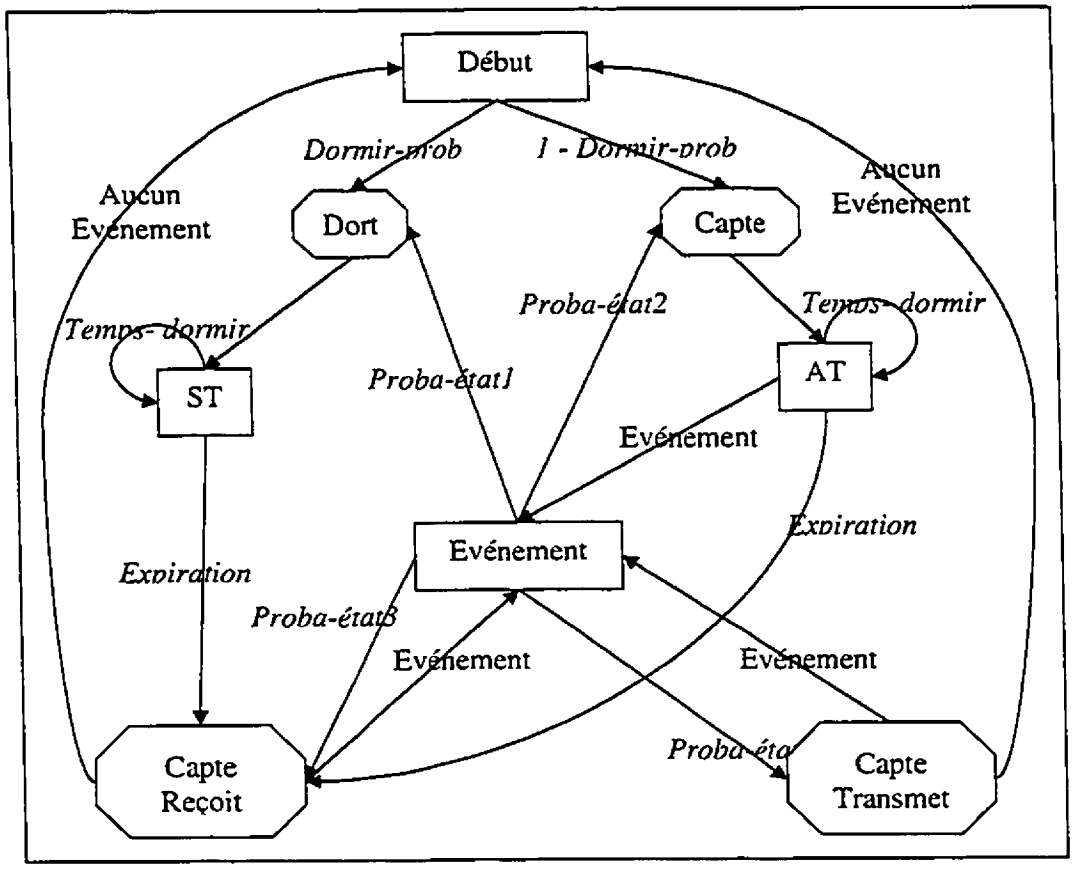

Figure 2.3. Modèle du comportement des nœuds

Dans l'état *capte*, le nœud reste *temps_dormir* secondes, mais à la différence de l'état *dort*, il peut capturer les événements durant cette période. Si un événement est déclenché pendant cette période, le nœud rentre dans les états *dort*, *capte*, *capte-reçoit* ou *capte-transmet* avec des probabilités *proba_état1*, *proba_état2*, *proba_état3* et *proba_état4* respectivement. Si le délai est expiré sans déclenchement d'événement, le nœud rentre dans l'état *capte-reçoit* afin de vérifier si un autre nœud tente de communiquer avec lui. De la même manière, il rentre dans l'état *dort* avec la probabilité *dormir_proba* ou dans l'état *capte* avec une probabilité (*1-dormir_proba*).

Les événements sont modélisés par un processus de Poisson avec un paramètre λ. Par conséquent, le nombre d'événements par seconde est décrit par la variable aléatoire

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!} \qquad (2.9)$$

Dans leur modèle probabiliste de la prédiction de la consommation d'énergie, Mini *et al.* modélisent un nœud par une chaîne de Markov. Dans cette modélisation, les modes d'opération du nœud sont représentés par les états de la chaîne de Markov et les variables aléatoires représentent les probabilités de rester dans un état pendant une certaine période de temps. Ainsi, on a, pour chaque nœud, une séquence de variables aléatoires X_0, X_1, X_2 ... qui représentent ces états dans les périodes de temps. Si $X_n = i$, alors le nœud est dans le mode d'opération i pendant la nième période de temps.

On définit aussi la probabilité P_{ij} qu'un nœud qui est dans le mode d'opération i devienne dans le mode j. $P_{ij} = P(X_{m+1}=j/X_m=i)$. On définit aussi la probabilité de transition à n-état, P_{ij}^n, qu'un nœud dans le mode i devienne dans le mode j après n transitions.

$P_{ij}^n = \sum_{k=1}^{M} P_{ik}^r P_{kj}^{n-r}$ pour toute valeur $0 < r < n$. M est le nombre de modes d'opération des nœuds. Supposons maintenant que E_S est l'énergie dissipée par un nœud dans l'état S pendant une seule période de temps, et si le nœud est dans le mode d'opération i, alors la prédiction de la consommation d'énergie E^T après une période de temps T est calculée par la formule :

$$E^T = \sum_{k=1}^{M} \left(\sum_{t=1}^{T} P_{ik}^t \right) \times E_S \qquad \textbf{(2.10)}$$

En utilisant cette formule, chaque nœud peut calculer son taux de consommation d'énergie pendant la période de temps suivante T, et l'envoyer avec l'énergie restante au nœud *collecteur* pour que ce dernier puisse construire ce qu'on appelle la *carte d'énergie.*

L'information sur l'énergie restante dans chaque nœud du réseau et son énergie prédite est appelée la *carte d'énergie* qui peut aider à la prolongation de la durée de vie du réseau. Cette information peut être utilisée par plusieurs algorithmes des différentes couches des réseaux de capteurs afin d'aider à l'optimisation de la consommation d'énergie ou simplement à la prise de décision quant au déploiement des nœuds et des applications. Dans notre approche de la conception d'un mécanisme de collecte des données, nous allons utiliser la carte d'énergie.

2.3.5. Amélioration du modèle d'états

Une amélioration du modèle de Mini *et al.* consiste à considérer les temps de transmission des messages et le temps de leur réception. Ces périodes sont très importantes vu que le capteur consomme beaucoup d'énergie pendant ces périodes. Le nœud qui rentre dans l'état *capte-reçoit* pour vérifier si des messages lui sont destinés reste dans cet état pendant *temps-reception*, le temps qu'il achève la réception du nombre de bits. Pendant cette période, un événement de capture peut se déclencher, le nœud attend la terminaison de la réception des messages avant de rentrer dans les autres états. De la même manière, un nœud qui rentre dans l'état *capte-transmet* reste dans cet état pendant temps-transmission, le temps qu'il achève la transmission des bits des messages qu'il doit envoyer. Pendant cette période, un événement de capture peut se déclencher, le nœud attend la terminaison de la transmission des messages avant de rentrer dans les autres états. La Figure 2.4 illustre cette amélioration.

Étant donné que le nœud peut être dans quatre états différents, la relation (2.10) peut être écrite de la manière suivante :

$$E^T = E_1(T) + E_2(T) + E_3(T) + E_4(T) \qquad \textbf{(2.11)}$$

$E_1(T)$ est l'énergie dissipée pendant la durée T quand le nœud est dans l'état 1 (*dort*). $E_2(T)$ est l'énergie dissipée pendant la durée T quand le nœud est dans l'état 2 (capte). $E_3(T)$ est l'énergie dissipée pendant la durée T quand le nœud est dans l'état 3 (*capte-reçoit*). $E_4(T)$ est l'énergie dissipée pendant la durée T quand le nœud est dans l'état 4 (*capte-transmet*).

En explicitant les formules de la consommation d'énergie à partir des relations (2.1) à (2.10), la relation (2.11) devient

$$E^T = \sum_{t=1}^{T} P_{i,1}^t \times E_1 + \sum_{t=1}^{T} P_{i,2}^t \times E_2 + \sum_{t=1}^{T} P_{i,3}^t \times n_t \times E_3 + \sum_{t=1}^{T} P_{i,4}^t \times m_t \times d^{\alpha} \times E_4 \qquad \textbf{(2.12)}$$

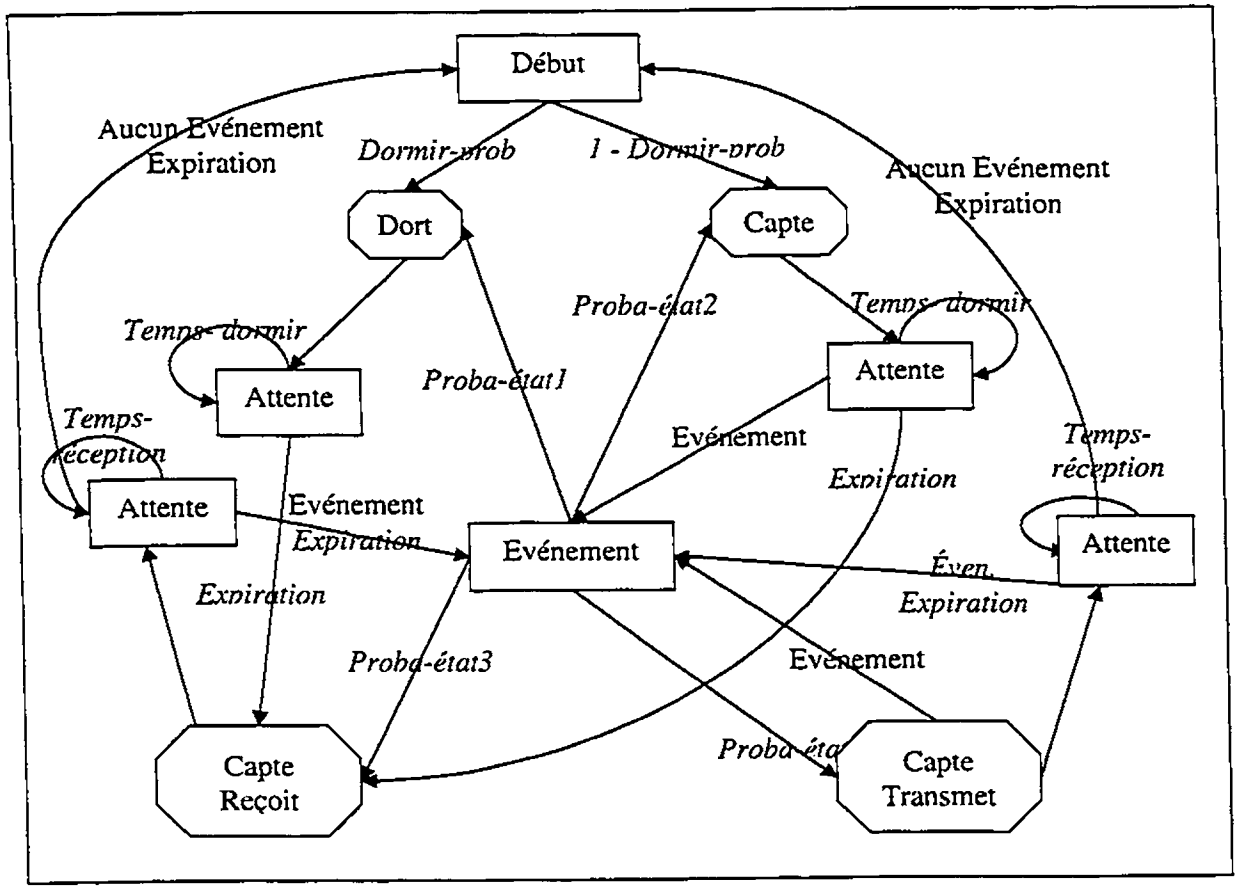

Figure 2.4. Modèle amélioré du comportement des nœuds

Le système est supposé être dans un état *i* quelconque au moment du calcul de l'énergie E^T. E_1 et E_2 sont respectivement les énergies consommées par le capteur dans le cas où il est dans les états *dort* et *capte* respectivement. Elles sont constantes puisque la durée T est subdivisée en des intervalles égaux. E_3 est l'énergie consommée par le capteur à la réception d'un seul bit, n_t est le nombre de bits reçus par le capteur pendant la période *t*. E_4 est l'énergie consommée par le capteur à la transmission d'un seul bit, m_t est le nombre de bits transmis par le capteur pendant la période *t*. *d* est la distance qui sépare le capteur de son voisin qui se chargera d'acheminer le message, supposée être constante pendant la période de calcul.

CHAPITRE III
MECANISME DE COLLECTE DE DONNÉES PROPOSÉ

L'ultime objectif de déploiement d'un réseau de capteurs sans fil demeure la collecte de données à partir des mesures que les nœuds prennent de leur environnement. Les méthodes de collecte de données devront prendre en considération le fait que les capteurs ont des capacités très réduites en terme d'énergie emmagasinée. Ils utilisent en général des mécanismes basés sur l'agrégation et le filtrage des messages. Nombreux sont les chercheurs qui ont conçu de tels mécanismes. Nous proposons une approche de collecte de données utilisant les mécanismes d'agrégation et de filtrage, ainsi que le principe de la formation des grappes basé sur la carte d'énergie.

Dans ce chapitre, nous allons tout d'abord présenter notre définition de la qualité de service (*QdS*) dans les réseaux de capteurs et notre définition de l'agrégation et filtrage. Ensuite, nous analyserons quelques approches de la collecte de données en mettant l'accent sur leurs lacunes. Enfin, nous décrirons notre approche de la collecte de données.

3.1. Traitement en réseau des requêtes : QdS, Agrégation et filtrage

La composante principale dans un système basé sur les réseaux de capteurs est le système opératoire. Le cœur d'un système opératoire transparent fonctionnant sous un réseau est un ensemble d'algorithmes qui décident quand et où faire immigrer les composants d'une application. Pour concevoir un système d'exploitation pour les réseaux ad hoc et les réseaux de capteurs qui réduit la consommation d'énergie par les applications, Barr *et al.* [BAR02] utilisent des algorithmes dont l'objectif est de réduire la longueur des chemins empruntés par les paquets communiqués par les composants d'une application, et ceci en faisant migrer automatiquement ces composants dans des nœuds qui sont topologiquement proches. D'où le principe de faire exécuter certaines fonctionnalités par les nœuds du réseau. Ce principe est connu sous le nom du *traitement en réseau.*

La raison principale qui justifie le traitement en réseau est que l'envoi de message par un nœud est toujours plus coûteux que le traitement des données par son processeur. En

effet, du point de vue de la consommation d'énergie [MAD02], transmettre un seul bit de données est équivalent à exécuter 800 instructions.

3.1.1. Qualité de service au niveau applicatif dans les réseaux de capteurs sans fil

Dans le domaine des réseaux, la qualité de service (QdS) est définie comme étant un ensemble de requis qu'un système de télécommunications peut offrir lors d'une session [ABO05]. Ces requis peuvent être spécifiés par une application de manière à satisfaire des exigences humaines ou d'autres exigences telle que la performance. Les différents indices de performance généralement rencontrés sont : le débit (e.g. 64 kbps), le délai de bout en bout (e.g. 300 ms), la gigue (e.g. 10 ms), la perte de paquets (e.g. 3%), le mode de transmission (unidirectionnelle ou bidirectionnelle), service garanti ou statistique, service interdomaine ou restreint à un domaine. Cette définition s'avère incomplète dans le cas des réseaux de capteurs.

Définition de la QdS

Dans le cas des réseaux de capteurs, les applications sont généralement intéressées aux données mesurées par les nœuds du réseau pendant une période de temps prédéterminée. Ainsi, la qualité des mesures et la durée de vie du réseau constituent les indices de performance les plus importants dans le cas des réseaux de capteurs. Par exemple, une application intéressée par la moyenne des températures dans des zones géographiques dans lesquelles le réseau est déployé, peut avoir des requis en terme de la fréquence des mesures (i.e. les mesures doivent être faites toutes les 15 minutes), de l'incertitude des mesures (i.e. une différence de 10% entre la valeur actuelle et la valeur précédente peut être négligée), et en terme de la durée de vie du réseau (i.e. les mesures doivent être prises pendant une année).

Certes, les indices de performance mentionnés dans la définition de la QdS dans le domaine des réseaux sont toujours utiles dans l'analyse des couches inférieures des réseaux de capteurs. Cependant, ils n'ont pas une grande importance dans le cas d'une catégorie très large d'applications de réseaux de capteurs. En effet, d'une part, ces applications sont caractérisées par une faible exigence en terme de débit (souvent mesuré en quelques bits par

jour) [CAL04]. D'autre part, ces types de réseaux sont généralement déployés en vue d'être dédiés pour faire fonctionner un seul type d'application. Ainsi, le réseau n'est pas amené à satisfaire des exigences de plusieurs types d'applications différentes.

À notre connaissance, l'inclusion de ces aspects dans la Qualité de Service des réseaux de capteurs n'a jamais été traitée par un travail de recherche. Par conséquent, notre première contribution dans la littérature est une redéfinition de la QdS dans ces nouvelles générations de réseaux pour lesquelles les données collectées constituent la raison d'être de leur déploiement et leur existence.

Spécification des exigences de la QdS

Madden *et al.* [MAD02] ont introduit la fréquence de la lecture des données dans les requêtes que les applications envoient au réseau de capteurs. Ils ont utilisé un langage similaire au langage SQL pour exprimer les requêtes. Une seule clause est ajoutée *EPOCH DURATION i*. Avec cette clause, l'utilisateur exprime le fait qu'il a besoin d'un ensemble de données envoyé par le réseau à chaque période de temps égale à i . La Figure 3.1 illustre un exemple de requête. L'utilisateur peut se désenregistrer et arrêter l'envoi de ces séries de données à n'importe quel moment. Par conséquent, la durée de la collecte de données n'est pas exprimée dans la requête initiale.

```
SELECT {aggn{expr},attrs} FROM sensors
      WHERE {selPreds}
      GROUP BY {attrs}
      HAVING {havingPreds}
      EPOCH DURATION i
```

Figure 3.1 Exemple de requête avec fréquence de collecte

Le détail de l'implémentation de cette manière d'exprimer les requêtes, ainsi que les significations des clauses peuvent être retrouvées dans [MAD02].

Pour exprimer l'incertitude tolérée dans la collecte des mesures, Beaver *et al.* [BEA03] introduisent une nouvelle clause dans les requêtes que les applications envoient au réseau afin d'obtenir des mesures. Ils ont adopté le même langage que les auteurs de [MAD02], basé sur le langage de requête SQL. La Figure 3.2 constitue un exemple de requête. La clause *VALUES WITHIN tct* permet au nœud de comparer sa nouvelle mesure avec la

dernière mesure captée et envoyée à son nœud parent. Si la différence entre les deux mesures est supérieure à *tct*, le nœud envoie la mesure, sinon il s'abstient. Leur objectif était de réduire le nombre de messages envoyés par les nœuds, et de réduire ainsi la consommation de l'énergie.

```
SELECT {aggregates,attrs} FROM sensors
      WHERE {conditions}
      GROUP BY {attrs}
      HAVING {conditions}
      EPOCH DURATION i
      VALUES WITHIN tct
```

Figure 3.2 Exemple de requête avec incertitude de mesure

3.1.2. Agrégation et filtrage

Un réseau de capteurs est généralement constitué d'un grand nombre de nœuds qui seront amenés à envoyer une masse très importante de données collectées aux applications. Le traitement de ce nombre fulgurant de données est complexe dans le cas où toutes les mesures arrivent aux applications qui se chargent de les analyser. Un système de capteurs devrait fournir aux applications un mécanisme qui leur permet d'exprimer leurs besoins en terme de données et de la fréquence des mesures, ce que nous avons appelé QdS. Il existe plusieurs manières de réduire le nombre de données reçues par une application. Les plus importantes sont l'*agrégation* des données et le *filtrage*. Le but de ces deux méthodes n'est pas seulement de maximiser la durée de vie du système en minimisant le nombre et la taille des messages échangés dans un réseau de capteurs, et ainsi réduire la consommation d'énergie, mais aussi de fournir aux applications les données nécessaires à leur fonctionnement sans les encombrer inutilement d'une masse de données.

Définition

Le mécanisme d'agrégation des données permet de rassembler plusieurs mesures en un seul enregistrement dont la taille est inférieure à la somme des tailles des mesures initiales. La sémantique du résultat ne devra ni se contrarier avec la sémantique des mesures initiales, ni faire perdre leur signification. Par exemple, une application veut calculer le maximum de la température mesurée par trois nœuds *N1*, *N2* et *N3* qui constituent un réseau de capteurs

structuré dans une hiérarchie en arbre, i.e. *N1* est le parent de *N2* et *N3*. Le nœud *N1* peut communiquer directement avec l'application. La Figure 3.3 illustre le calcul de l'agrégation par le nœud *N1*. Au lieu que l'application reçoive les trois mesures et par la suite calcule le maximum des trois valeurs, le nœud *N1* peut collecter les deux mesures de *N2* et *N3*, capter sa température, calculer le maximum et envoyer un seul résultat à l'application. Dans le premier scénario sans agrégation, cinq envois de messages sont exécutés. Dans le deuxième scénario avec agrégation, seulement trois envois de messages sont exécutés.

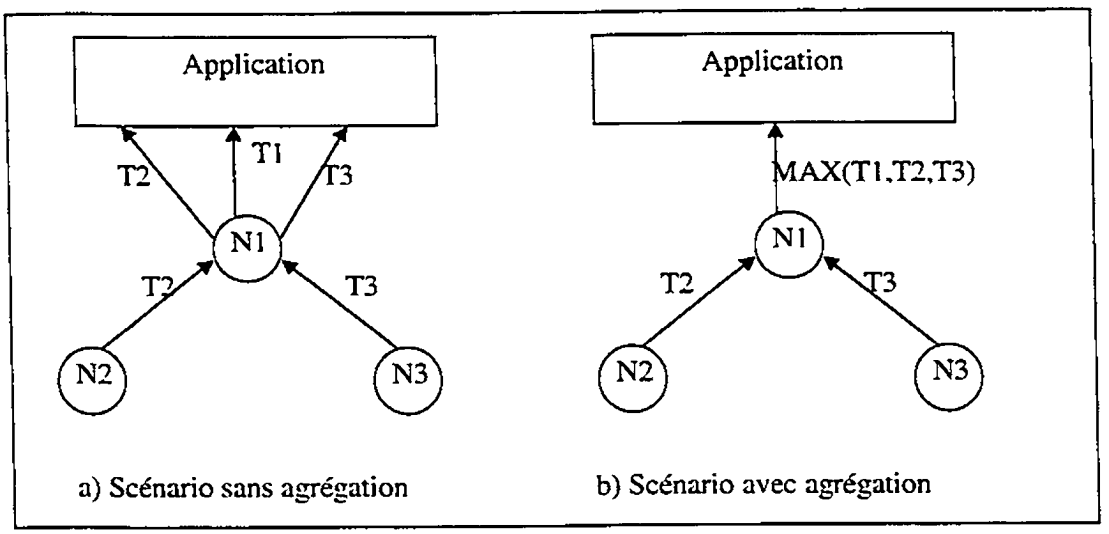

Figure 3.3 Principe de l'agrégation

Le mécanisme de filtrage des données permet d'ignorer des mesures qui sont redondantes ou qui ne sont pas pertinentes pour le fonctionnement d'une application. Un système de réseau de capteurs devra fournir aux applications des mécanismes qui leur permettent de spécifier les conditions qu'une mesure doit satisfaire pour qu'elle soit pertinente. Par exemple, une application peut être intéressée par la température qui est supérieure à un seuil et est prise dans une zone géographique spécifique. Le système devrait filtrer les messages dans le réseau et ne faire acheminer que les mesures qui satisfont les conditions de filtre.

La Figure 3.4 illustre le mécanisme du filtrage des mesures. Dans cet exemple, l'application a besoin des températures qui sont supérieures à 25 degrés. Dans le scénario *a* (sans filtrage), tous les nœuds envoient leurs mesures à l'application qui se chargera de sélectionner les mesures pertinentes. Dans ce scénario, cinq messages sont transmis et

l'application est encombrée par leur réception. Dans le scénario *b* (avec filtrage), les nœuds *N2* et *N3* s'abstiennent d'envoyer leurs mesures qui sont inférieures à 25 degrés, et un seul message est envoyé. Ainsi, le système gagnera l'envoi de quatre messages et l'application ne reçoit que la mesure pertinente.

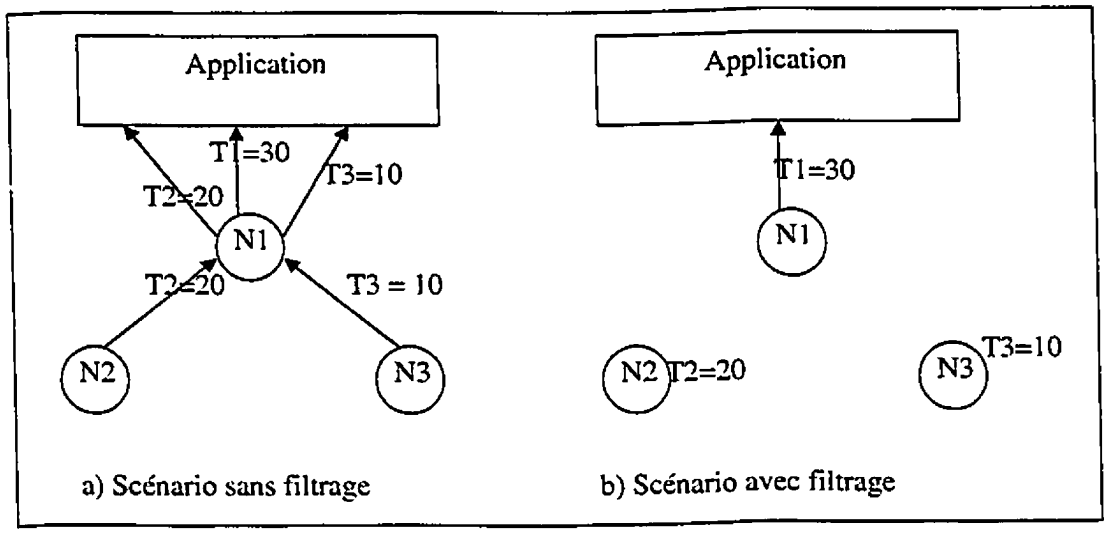

Figure 3.4 Principe du filtrage des mesures

Le mécanisme de filtrage des données a été utilisé dans le protocole ALE (Application Level Events) [ALE05] qui permet aux applications traitant les données provenant des dispositifs RFID (Radio Frequency IDentification) de communiquer avec les contrôleurs de ces dispositifs [WEI03]. La technologie RFID est une technologie émergente qui permet d'identifier des objets physiques par des composants électroniques qui peuvent être questionnés à travers des fréquences radio. Dans le protocole ALE, une application peut spécifier le type de dispositifs électroniques qui peuvent envoyer leurs données. Afin de rendre ceci possible, un système de filtrage basé sur la norme EPC (Electronic Product Code) [CHO01] est conçu. Le filtrage s'exécute au moment de l'envoi des données à l'intérieur de l'environnement d'exécution des contrôleurs de dispositifs électroniques.

A notre connaissance, il n'y a pas un travail de recherche qui sépare le mécanisme de filtrage de celui de l'agrégation. Ceci constitue une nouvelle contribution dans la littérature des réseaux de capteurs.

Classification des fonctions d'agrégation

Madden *et al.* [MAD02] introduisent une classification des fonctions d'agrégation basée sur leurs comportements vis-à-vis de la collecte des mesures des capteurs. Cette classification permet d'étudier la possibilité d'intégrer le calcul de ces fonctions dans le réseau, appelé *traitement en réseau* des fonctions. Les fonctions de l'agrégat étudiées sont celles les plus connues dans les requêtes SQL : COUNT, MIN, MAX, SUM, AVERAGE, MEDIAN, COUNT DISTINCT, HISTOGRAM. Ces fonctions sont classifiées selon quatre propriétés différentes : Agrégat *sensible à la duplication*, agrégat *exemplaire ou sommaire*, agrégat *monotone* et *état partiel* de l'agrégat. Le Tableau 3.1 contient cette classification.

Tableau 3.1 Classification des agrégats

	Sensible à la duplication	**Exemplaire (E), Sommaire (S)**	**Monotone**
MAX, MIN	Non	E	Oui
COUNT, SUM	Oui	S	Oui
AVERAGE	Oui	S	Non
MEDIAN	Oui	E	Non
COUNT DISTINCT	Non	S	Oui
HISTOGRAM	Oui	S	Non

La première propriété est sensible à la duplication. Un agrégat qui n'est pas sensible à la duplication n'est pas affecté par la duplication de l'envoi des mesures par un nœud. Cette propriété implique des restrictions dans les propriétés du réseau. La deuxième propriété est agrégat exemplaire. Un agrégat exemplaire retourne une ou plusieurs valeurs représentatives à partir de l'ensemble des valeurs. La somme (SUM) retourne une valeur à partir de toutes les valeurs. L'importance de cette propriété est que les agrégats exemplaires se comportent d'une manière imprédictible devant une perte de données dans le réseau. Troisièmement, les agrégats monotones ont la propriété suivante : si on a deux résultats intermédiaires *s1* et *s2*, alors le résultat de l'agrégation *s'* a la propriété suivante :

$$\forall s_1, s_2, e(s') \leq MAX(e(s_1), e(s_2))$$

ou

$$\forall s_1, s_2, e(s') \geq MIN(e(s_1), e(s_2))$$

La fonction e est la fonction qui permet d'évaluer l'agrégation. La quatrième propriété « état partiel » est reliée au nombre de variables requis pour présenter un état. Par exemple, la moyenne (AVERAGE) a besoin de deux variables pour présenter un état partiel. Quant à COUNT, il a besoin d'une seule variable.

3.1.3. Méthodes d'agrégation et de filtrage

Plusieurs travaux de recherche se sont intéressés à la conception d'un mécanisme de collecte de mesures qui optimise l'utilisation de la consommation d'énergie. Leurs approches se basent sur l'implémentation des mécanismes d'agrégation et de filtrage des données. L'importance de ces approches réside dans le fait que les réseaux de capteurs sont considérés comme des bases de données réparties où chaque nœud contient une ou plusieurs données. Les applications seront toujours intéressées par la collecte des mesures captées par les nœuds du réseau.

Agrégation légère TAG (Tiny Agregation)

Madden et al. [MAD02] ont conçu une agrégation en réseau qui permet de calculer les données au fur et à mesure qu'ils arrivent aux nœuds, en supprimant les données qui ne sont pas pertinentes et en combinant les données en un seul enregistrement compact quand ceci est possible. Ils utilisent un routage en arbre qui permet de satisfaire les exigences de l'agrégation. L'algorithme de routage, dans leur cas, fixe deux objectifs. Premièrement, il doit acheminer les requêtes à partir du nœud *collecteur* vers tous les autres nœuds du réseau, et deuxièmement, il doit trouver des routes à partir des nœuds où l'agrégation collecte les données vers le nœud racine. Ces chercheurs [MAD02] ont utilisé un langage pour exprimer les requêtes similaires au langage SQL.

La structure de l'agrégat, appelé agg_n, est implémentée à travers trois fonctions : une fonction de fusion f, un initialiseur i, et un évaluateur e. En général, f a la forme suivante :

$$<> = f(<x>,<y>$$

où <x> et <y> sont des résultats partiaux émanant d'un ou de plusieurs nœuds. Par exemple, si la fonction de fusion *f* est celle de la *moyenne*, la fonction a besoin de la somme *S* et du compte *N* (i.e. le nombre de nœuds qui ont envoyé leurs mesures). Étant donné deux résultats intermédiaires $<S1,N1>$ et $<S2,N2>$, $f(<S1,N1>,<S2,N2>) = <S1+S2,N1+N2>$, l'initialiseur *i* est utilisé pour initialiser le calcul avec une seule valeur. L'initialiseur de la *moyenne* sera $<S,1>$. L'évaluateur *e* permet de calculer la valeur actuelle de l'agrégat. L'évaluateur de la *moyenne* est $e(<S,N>) = S / N$. La Figure 3.5 illustre les trois composants d'un agrégat qui calcule la moyenne.

Fonction Agrégat : $f(<S1,N1>,<S2,N2>) = <S1+S2,N1+N2>$

Initialiseur : $i(S) = <S,1>$

Évaluateur : $e(<S,N>) = S / N$

Figure 3.5 Différents composants d'un agrégat qui calcule la moyenne

Le mécanisme de l'agrégation TAG s'exécute en deux phases : *la phase de distribution* dans laquelle les requêtes sont acheminées aux différents nœuds du réseau, et *la phase de collection*, dans laquelle les valeurs agrégées sont continuellement routées des nœuds fils aux nœuds parents. Pour pouvoir réduire le nombre de messages durant la phase de collection, les nœuds parents devront attendre les résultats partiaux de leurs nœuds fils avant de calculer les agrégats et envoyer les résultats finaux au nœud central ; et ceci devrait être avant que la période *i* soit expirée, i.e. la période spécifiée dans la clause EPOCH de la requête. Pour cette raison, le nœud parent subdivise la période *i* en un ensemble d'intervalles de temps pour que les nœuds fils puissent avoir suffisamment de temps pour calculer leurs résultats partiaux et les envoyer à leurs parents. Les auteurs n'ont pas explicité un algorithme qui permet de subdiviser cet intervalle de manière à ce que les performances du système ne soient pas affectées par une attente prolongée ou un intervalle de temps insuffisant. Le seul algorithme présenté consiste à connaître le nombre de niveaux dans l'arbre (i.e. la profondeur *d* de l'arbre) et à faire une division euclidienne de la période *i* sur le nombre de niveaux (i.e. l'intervalle = i / d).

La Figure 3.6 illustre les deux phases de l'agrégation TAG qui permettent de calculer la moyenne de la température dans un réseau de capteurs. La fréquence de la collecte des mesures est égale à 10 minutes. Le nombre de niveaux dans la hiérarchie en arbre est égal à deux. Par conséquent, les nœuds qui constituent les feuilles envoient leurs mesures après toutes les 5 minutes et les parents qui se situent dans le premier niveau envoient leur mesure après 10 minutes.

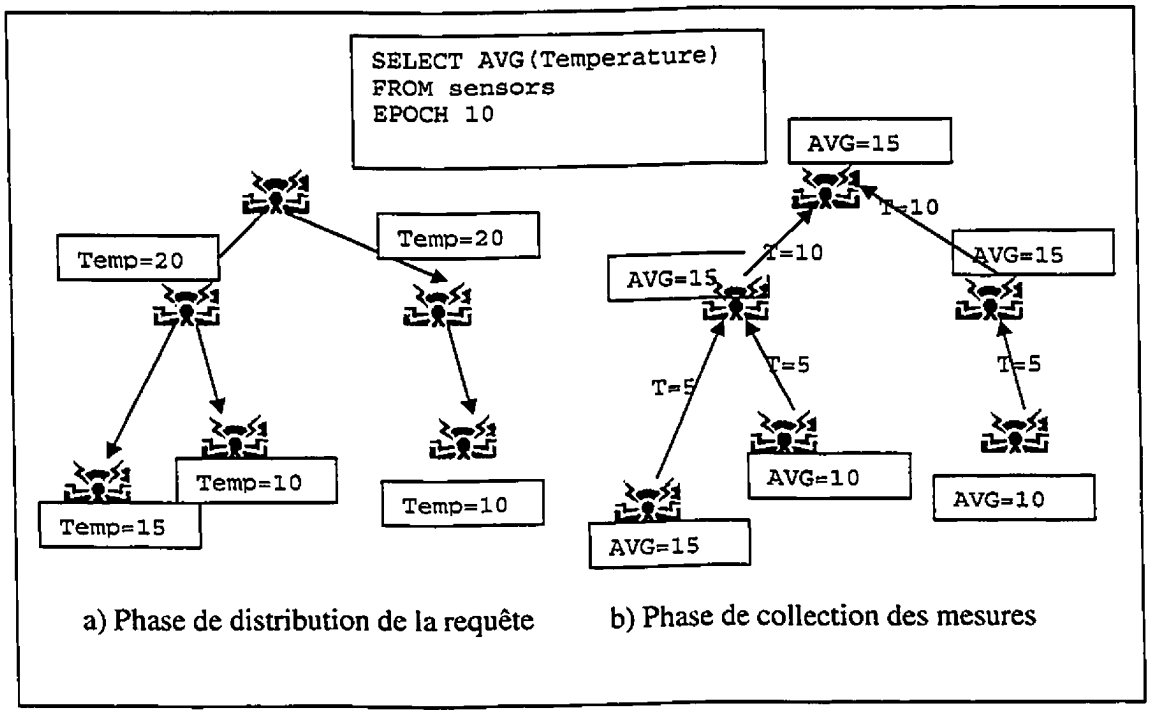

Figure 3.6 Phases de l'agrégation TAG

Les auteurs [MAD02] utilisent des mécanismes qui permettent d'optimiser l'agrégation et d'augmenter sa fiabilité devant la perte de message. Le premier mécanisme se base sur l'avantage d'avoir des canaux de communications partagés entre les différents nœuds du réseau. Un nœud qui n'a pas reçu la requête initiale peut participer à l'envoi des résultats partiels en analysant le trafic des nœuds dans son voisinage. Si le nœud réalise que les autres nœuds envoient des résultas, il peut en déduire qu'il doit envoyer les siens. Un nœud n'est pas obligé d'écouter le réseau continuellement mais il peut choisir des périodes pour le faire pour qu'il puisse conserver son énergie. L'analyse de trafic peut aussi aider à réduire le nombre de messages utilisés. En connaissant les résultats des nœuds voisins, un nœud peut

décider de ne pas envoyer le sien, e.g., dans le cas de calcul de la valeur maximale. L'application de cette dernière technique est directe dans le cas des fonctions monotones, MAX et MIN. Les auteurs ne présentent pas une analyse profonde de l'impact de cette optimisation sur la consommation de l'énergie due à la réception des messages.

Afin d'améliorer la qualité des résultats agrégés, les auteurs [MAD02] proposent un système d'antémémoire : les nœuds parents conservent les résultats partiels de leurs fils pour un ensemble de cycles d'agrégation, les parents utilisent ces résultats dans le cas où ils ne peuvent pas obtenir de résultats dus à une perte de message par exemple. Généralement, avoir une ancienne mesure vaut mieux que de ne pas avoir de résultat.

Le système d'antémémoire n'est pas toujours possible (i.e. certains capteurs n'ont pas cette capacité). De ce fait, les auteurs proposent une autre alternative : un nœud peut avoir plus qu'un nœud parent, ainsi il peut envoyer ses mesures partielles à ces deux parents. Si un message est perdu, le deuxième message peut être pris en considération. Ce mécanisme qui consiste à dupliquer les messages peut fonctionner uniquement pour les agrégats qui ne sont pas sensibles aux duplications, e.g. MAX, MIN, mais il n'est pas convenable dans le cas des agrégats sensibles à la duplication, e.g. COUNT, AVRAGE. Dans ces derniers cas, le nœud peut subdiviser ces mesures en deux parties et envoyer la première moitié au premier parent et la deuxième moitié au deuxième parent.

En plus des désavantages recensés lorsque l'agrégation TAG est décrite dans cette section, il faut mettre l'accent sur d'autres faiblesses qui sont aussi importantes :

1. Ce mécanisme ne contient pas un mécanisme de filtrage. Tous les nœuds du réseau envoient leurs mesures pendant des périodes qui peuvent être proches. Ceci aura un impact sur la durée de vie du réseau dû à une consommation d'énergie accrue causée par la communication de plusieurs messages ;
2. Tous les nœuds parents calculent la fonction d'agrégation et constituent une passerelle pour tous les nœuds fils. Par conséquent, les nœuds parents qui sont proches du nœud *collecteur* seront très sollicités et auront tendance à consommer plus d'énergie que les autres nœuds. Cette faiblesse est due à la hiérarchie en arbre que le mécanisme a adopté pour construire ces routes ;
3. Le mécanisme n'explicite pas la méthode adoptée pour construire l'arbre. La construction de cet arbre pourra avoir un impact sur l'efficacité de l'agrégation.

Dans le pire des cas, l'arbre peut être une seule chaîne avec un seul nœud racine et un seul nœud feuille.

Filtrage basé sur les grappes et l'introduction des incertitudes

Yoon *et al.* [YOO04] proposent un algorithme de traitement des requêtes en réseau qui permet de réduire la consommation d'énergie en diminuant le nombre de messages échangés. Similaire à l'agrégation légère TAG, l'algorithme conçu par Yoon *et al.* opère en deux phases : distribution de la requête et collecte des mesures. Dans la phase de distribution de la requête, les grappes sont formées en fonction des mesures actuelles des nœuds et d'un cœfficient, qui constitue l'incertitude des mesures τ définie par l'application. Lors de la phase de la prise des mesures, seules les têtes de grappes envoient leurs mesures. Les mesures des autres nœuds sont considérées comme redondantes par rapport à la valeur de τ. Un nœud décide de participer à une grappe en fonction de la mesure M_T de la tête de grappe et de sa mesure locale M_L. Si $M_T - M_T \times \tau \leq M_L \leq M_T + M_T \times \tau$, alors le nœud appartient à la même grappe.

La Figure 3.7 illustre cet algorithme. Dans cet exemple, l'application a besoin de calculer la moyenne de la température dans le réseau avec une incertitude de 10 % (τ = 10 %). Le nœud *N1* qui est le premier à recevoir la requête constitue la tête de la première grappe. Il assigne sa température à $M_T = 50$ et envoie la requête aux nœuds *N2* et *N3* (M_T qui fait partie de la requête envoyée aux *N2* et *N3*). Les températures de *N2* et *N3* ($M_{L2} = 47$ et $M_{L3} = 52$) sont entre 55 (50 + 10%) et 45 (50 – 10%) et donc ils appartiennent à la première grappe. À son tour, le nœud *N2* envoie la requête, sans modifier la valeur M_T, aux nœuds *N4* et *N5*. La température du nœud *N5* ($M_{L5} = 59$) n'est pas dans l'intervalle [45,55] et donc il constitue une deuxième grappe et assigne sa température à M_T, et ainsi de suite. Dans la phase de la distribution, cinq grappes sont constituées. Lors de la phase de la collecte des mesures, les nœuds *N1*, *N5*, *N7*, *N8* et *N10*, qui constituent les têtes de grappes, envoient leurs mesures. Les autres nœuds sont des passerelles pour acheminer les messages et peuvent aussi calculer la moyenne des températures, i.e. un traitement en réseau des requêtes.

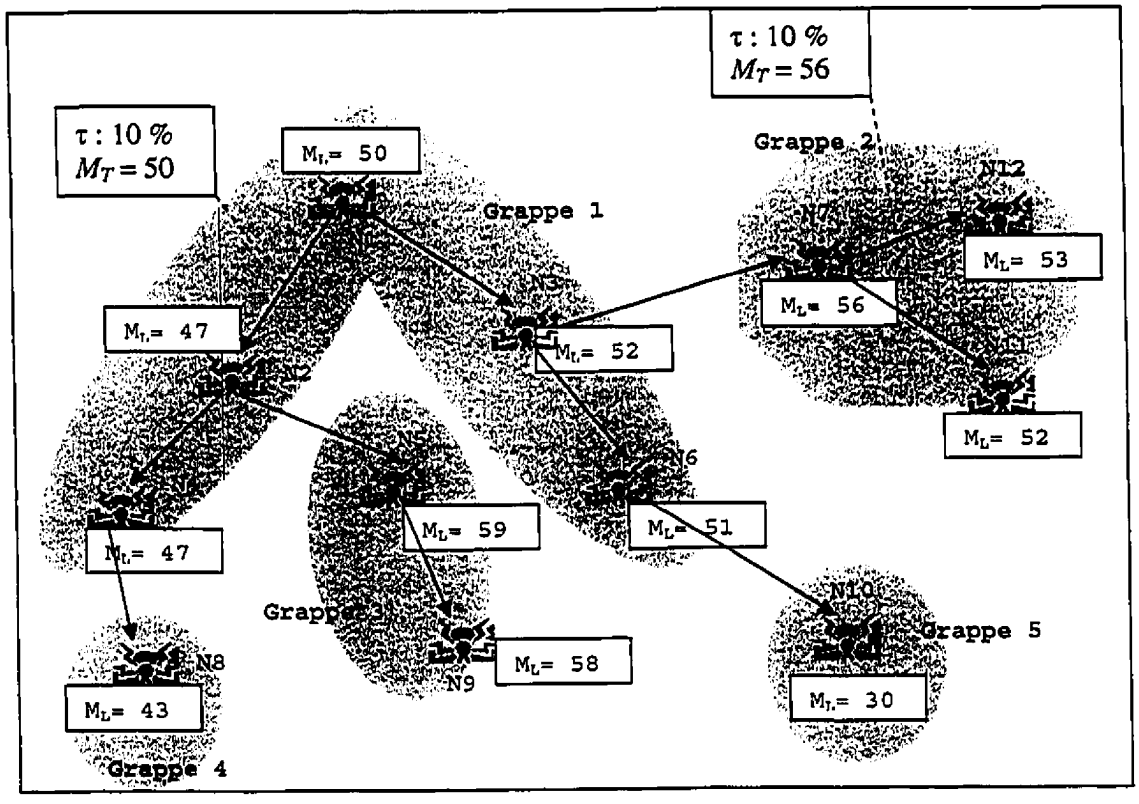

Figure 3.7 Phase de distribution de la requête du filtrage basé sur les grappes

Plusieurs faiblesses ont été relevées en analysant cette approche :

1. Les grappes sont constituées pour chaque requête. Imaginons qu'une application a *n* requêtes par *T* temps, alors les grappes seront constituées *n* fois pour la même période ;
2. L'agrégation fait abstraction de la sémantique des nœuds qui ont envoyé leurs mesures. Les localisations des grappes devraient être communiquées au nœud central pour qu'il puisse faire la correspondance entre les mesures des grappes et leur localisation. Imaginons que l'application veut savoir la température des zones, en utilisant cet algorithme, l'application ne pourra avoir que les températures de certains nœuds, têtes de grappes ;
3. Le protocole ajoute des messages qui pourront être volumineux dû à l'envoi des mesures des têtes de grappes durant la phase de la distribution de la requête, ce qui aura une influence sur la consommation d'énergie. Il faut étudier la taille des messages échangés par rapport à la taille des messages de mesures ;

4. Cette manière de collecter l'information ne peut pas être utilisée dans le cas où l'utilisateur envoie une seule requête pour demander continuellement des mesures. La formation des grappes ne sera plus valide vu que les mesures sont généralement en perpétuelles modifications.

Yoon *et al.* [YOO04] ont proposé un mécanisme de filtrage basé sur un seuil d'erreur que l'application définit afin de limiter le nombre de capteurs qui participent dans la phase de la collecte de mesure, mais leur manière d'exprimer les requêtes ne respecte pas un standard. Beaver *et al.* [BEA03] proposent d'adopter le langage SQL afin de faciliter aux applications d'exprimer ce requis. En fait, ils ont amélioré la proposition de Madden *et al.* [MAD02] en introduisent une nouvelle clause dans les requêtes SQL, qui permettent de questionner les nœuds des réseaux pour pouvoir limiter les messages envoyés par les nœuds, et ainsi réduire la consommation de l'énergie. Un exemple de requête est présenté à la Figure 3.6 précédemment analysée.

Cette nouvelle clause permet au nœud de comparer sa nouvelle mesure avec la dernière mesure envoyée à son nœud parent. Si la différence entre les deux mesures est supérieure à un seuil défini par l'application, le nœud envoie la mesure, sinon il s'abstient. Des messages '*heart beat*' sont envoyés entre les différents nœuds du réseau pour pouvoir distinguer entre un nœud qui tombe en panne et une mesure qui n'est pas envoyée à cause de cette contrainte introduite. Ceci constitue un défaut de cette méthode. Les auteurs étudient la réduction de la consommation d'énergie mais oublient d'introduire les effets de ces messages sur la consommation d'énergie.

3.2. Méthodes et algorithmes proposés

Dans ce travail, nous proposons une nouvelle approche d'agrégation qui utilise la carte d'énergie afin de minimiser la consommation d'énergie et de maximiser la couverture du réseau. Nous avons essayé de combler les lacunes des agrégations que nous avons analysées afin de concevoir un filtrage et une agrégation qui permettent de satisfaire les exigences des applications en terme de QdS préalablement définie.

3.2.1. Modèle du réseau

Le réseau considéré est un réseau fixe où les emplacements géographiques des nœuds sont connus et ne changent pas avec le temps. Il est constitué d'un ensemble V de N capteurs : $V = \{N_i / i = 1..N\}$. Un nœud S, appelé nœud *collecteur*, reçoit toutes les requêtes que l'application envoie au réseau et transmet les résultats des requêtes à l'application. Dans le modèle du réseau considéré, les nœuds ne peuvent pas envoyer directement leurs mesures au nœud S, i.e. le modèle multi-hop. Par contre, ils utilisent les nœuds voisins pour pouvoir communiquer. Le modèle de nœud est celui décrit à la section « Modèle des états des nœuds et carte d'énergie ». La Figure 3.8 illustre le modèle réseau pris en considération. Les nœuds configurent leurs ondes radios de manière à assurer la connectivité du réseau.

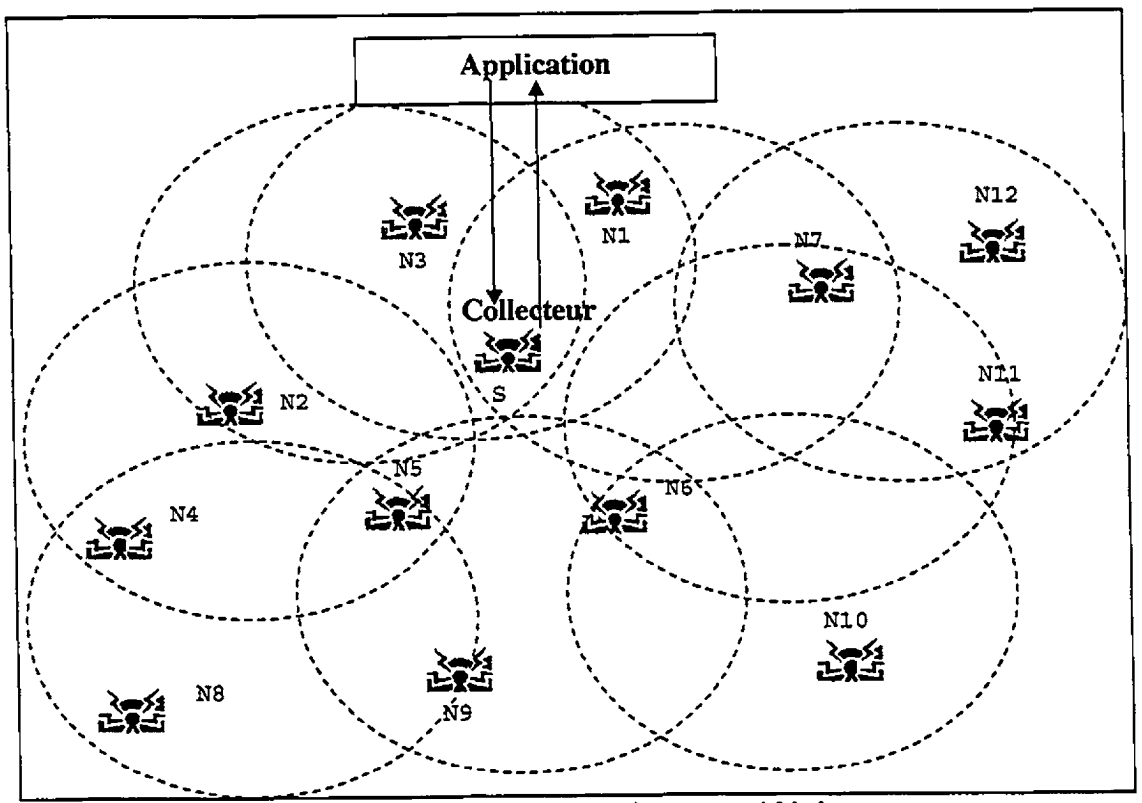

Figure 3.8 Modèle du réseau considéré

3.2.2. Phase de distribution

Le but de cette phase est de construire des grappes avec leurs têtes qui optimisent la consommation d'énergie en faisant intervenir la QdS que l'application fournira dans sa requête. L'application peut spécifier les requis suivants :

1. La fréquence des envois des résultats *fq* : à chaque expiration d'une période de temps égale à *fq*, le réseau doit fournir un résultat à l'application ;
2. Le seuil des erreurs de mesures *sem* : à l'intérieur d'une zone, dont la longueur est fixée par l'application (Quatrième requis), si la différence entre deux mesures est inférieure à la valeur *sem*, alors l'une des deux mesures est considérée redondante ;
3. La durée totale de la requête *T* : le réseau doit fournir les mesures pendant une durée totale égale à *T* ;
4. Le pas des mesures *pas* : la valeur de *pas* détermine la longueur des zones. À l'intérieur d'une zone, les mesures sont considérées redondantes. Dans le cas où l'application a besoin de plus de précision, elle peut diminuer la longueur des zones ou bien elle peut omettre de spécifier cette valeur.

L'application spécifie ces requis en utilisant le langage préconisé par Madden *et al.* [MAD02] et Beaver *et al.* [BEA03] et qui est basé sur le standard SQL. Deux nouvelles clauses sont ajoutées : la première permet d'exprimer le pas des mesures requis *ZONES WITHIN pas* et la deuxième permet de spécifier la durée totale de la requête *TOTAL DURATION T*. La Figure 3.9 présente un exemple général de requête.

```
SELECT {aggregates,attrs} FROM sensors
      WHERE {conditions}
      GROUP BY {attrs}
      HAVING {conditions}
      EPOCH DURATION fq
      VALUES WITHIN sem
      ZONES WITHIN pas
      TOTAL DURATION T
```

Figure 3.9 Exemple de requête avec deux nouvelles clauses

3.2.3. Formation des grappes

Le but de cette phase est de diviser les nœuds du réseau en un ensemble de grappes G_i qui ont pour objectif de satisfaire les requis de l'application tout en maximisant la durée de vie du réseau et ainsi maximiser sa couverture. Les grappes sont constituées en prenant en considération les critères suivants :

- maximiser la couverture du réseau en utilisant la carte d'énergie ;
- rassembler les nœuds susceptibles de prendre des mesures considérées redondantes ;
- rassembler les nœuds qui se localisent selon les requis de l'application.

Selon ces critères, le problème de la création des grappes revient à trouver les ensembles G_i qui satisfont les conditions suivantes :

1. $\left(\bigcup_i G_i = V\right) \wedge \left(\forall (i,j)\ \ G_i \cap G_j = \phi\right)$
2. $\forall N_j, N_l \in G_i \quad \left|M_j(t) - M_l(t)\right| \leq sem$
3. $\forall N_j, N_l \in G_i \quad d_{j,l} \leq pas$
4. $\exists N_j \in G_i \qquad E_j^T \leq E_j^{restante}$

$M_j(t)$ est la mesure capturée par le nœud N_j durant la période t. $d_{j,l}$ est la distance entre les nœuds N_j et N_l. E_j^T est l'estimation de la consommation d'énergie par le nœud N_j durant la période future T. $E_j^{restante}$ est l'énergie restante au nœud N_j calculée au moment de la formation des grappes.

La première condition permet de structurer tous les nœuds à l'intérieur des ensembles (grappes) disjoints. La deuxième condition permet d'inclure dans une grappe les nœuds dont les mesures sont susceptibles d'être redondantes selon les requis de l'application (Le paramètre *sem*). Les mesures similaires seront filtrées par les nœuds du réseau. Cette condition peut être satisfaite en utilisant les mesures initiales des capteurs ou les moyennes des mesures des capteurs calculées en fonction des historiques des mesures. Dans ce dernier cas, le capteur devra avoir la capacité de stocker cette moyenne et de la fournir à l'algorithme de la formation des grappes.

La troisième condition permet de rassembler les nœuds qui sont localisés dans des zones dont les périmètres sont déterminés par le paramètre *pas* fourni dans les requis de la QdS de l'application. Une grappe fournira une seule mesure par zone, et ainsi un système de filtrage sera implémenté. La quatrième condition assurera que les grappes seront créées en incluant un nœud qui garantira la couverture de cette zone jusqu'à la fin de la période de la collecte des mesures T prédéterminée par l'application.

Expression des valeurs de QdS et algorithme de création de grappes

L'application exprime ses requis en terme de QdS (fq,sem,T,pas), les types de données dont elle a besoin, ainsi que la fonction d'agrégation qu'elle veut calculer. Elle envoie cette requête écrite en langage décrit précédemment au nœud collecteur S. Afin de créer les grappes, le nœud S transforme cette requête en un message de création des grappes. Ce message contient les éléments suivants :

- X_p,Y_p : les coordonnées du premier nœud p qui a commencé la formation de la grappe. Les valeurs de ces coordonnées sont initialement vides ;
- M_p : la mesure du premier nœud p ;
- fq, sem, T, pas : les éléments de QdS ;
- l_p : la longueur de la grappe, sa valeur initiale est égale à pas ;
- *FlagCouverture* : ce flag spécifie si la grappe contient un nœud qui assurera la couverture de la zone représentée par la grappe pendant toute la durée de la collecte des mesures T ;
- *GrappeId* : cet élément contient un identifiant de la grappe.

Les coordonnées (X_p,Y_p) du premier nœud p qui a commencé la formation de grappe sont utilisées pour que les nœuds puissent vérifier la troisième condition. Le nœud peut calculer la distance qui le sépare du nœud p en connaissant sa position ainsi que la position du nœud p. Un nœud dont les conditions 2, 3 et 4 ne sont pas satisfaites envoie un message au nœud collecteur S pour l'informer des informations de la grappe courante, i.e les données sont incluses dans le message de la création de grappes. Ensuite, il assigne ses coordonnées à (X_p,Y_p) pour commencer la formation d'une nouvelle grappe.

L'élément l_p représente la distance maximale entre le nœud p et les nœuds de la grappe. Il permet de délimiter géographiquement les périmètres des grappes afin de satisfaire le

requis de l'application concernant le paramètre *pas*, et permet aussi de donner une sémantique aux grappes. Ainsi, si l'application reçoit une seule mesure d'une grappe (Filtrage des données), elle sera en mesure d'identifier géographiquement laquelle des zones est couverte par la grappe. La valeur de cet élément est égale à la valeur du paramètre *pas* fourni par l'application. Sa valeur peut être modifiée dans le cas où la quatrième condition n'est pas satisfaite, i.e. l'algorithme favorise la couverture des zones sur la longueur des grappes. L'algorithme de la création des grappes est schématisé à la Figure 3.10. Il doit être exécuté par tous les nœuds du réseau de capteurs.

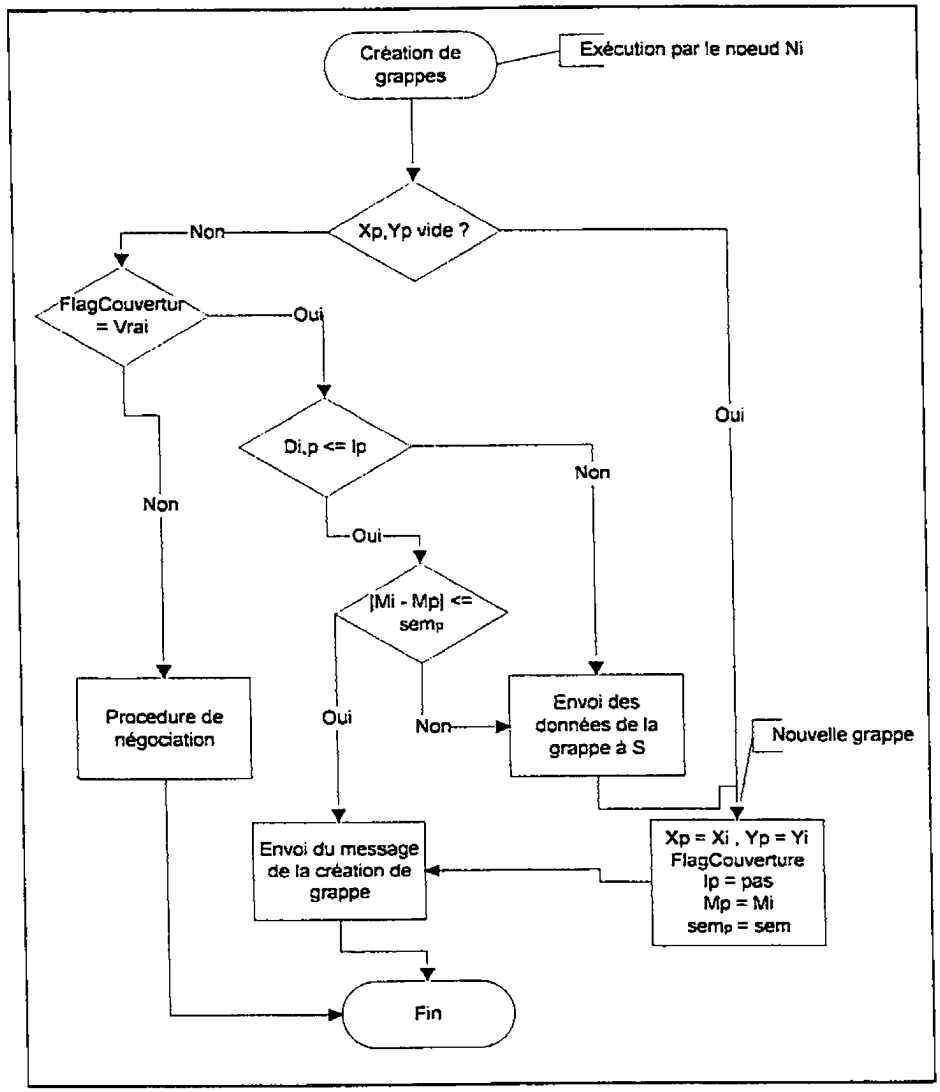

Figure 3.10 Organigramme de l'algorithme de création de grappes

La procédure de négociation

Dans le cas où le premier nœud p réalise que son flag *FlagCouverture* n'est pas égal à *vrai*, il déclenche une procédure de négociation avec ses voisins afin de trouver un nœud qui peut garantir le requis de la couverture des zones, i.e. satisfaire la quatrième condition. Tous les nœuds voisins qui ne sont pas encore affectés à une grappe participent à la procédure de négociation. Un nœud N_i participant, i.e. qui reçoit le message de création de grappe avec *FlagCouverture* = *Faux*, vérifie la couverture avec ses propres énergies. Si le nœud N_i peut assurer la couverture de la grappe, il retourne un message qui contient sa distance $d_{i,p}$ qui le sépare du premier nœud p, ainsi que sa mesure M_i qui doit aider dans la vérification de la deuxième condition. Sinon, le nœud N_i, pour sa part, envoie la requête de création de grappes à son entourage, et ainsi de suite jusqu'à ce que l'algorithme trouve un nœud de couverture de la grappe.

Dans le cas de l'exécution de la procédure de négociation, le premier nœud p attend pendant une certaine durée qui pourrait être déterminée par le nœud collecteur S en fonction de la topologie mise en place et de l'emplacement géographique des nœuds. Après, il calcule la nouvelle longueur de la grappe en fonction des messages retournés : $l_p = MAX\left(pas, \underset{i}{MIN}\left(d_{i,p}\right)\right)$ et la nouvelle incertitude des mesures de la grappe : $sem_p = MAX\left(sem, \underset{i}{MIN}\left(\left|M_i - M_p\right|\right)\right)$. Ensuite, il redistribue la requête de création de grappes avec les nouvelles valeurs des paramètres l_p et sem_p, et avec le paramètre *FlagCouverture* = *Vrai*. Avec ces nouvelles valeurs, l'algorithme garantira que la grappe construite aura au moins un nœud qui couvrira la zone pendant toute la durée de la collecte des données.

Dans le cas particulier où la valeur de paramètre *pas* n'est pas fournie par l'application comme étant un élément de la QdS, l'algorithme ignore la vérification de la troisième condition de la formation de grappes. De la même manière, la deuxième condition est ignorée quand l'application ne fournit pas le paramètre *sem* et la quatrième condition est ignorée quand celle-ci ne fournit pas la valeur du paramètre T qui permet de calculer la consommation d'énergie estimée par les nœuds du réseau.

Dans le cas particulier où le flag *FlagCouverture* n'est jamais égal à *Vrai*, i.e. la couverture des zones ne peut être satisfaite avec les valeurs des paramètres de la QdS, le nœud *S*, en recevant les données sur les grappes, négocie ces valeurs avec l'application.

Exemple de formation de grappes

La Figure 3.11 présente un exemple de création de grappes pour une application qui veut calculer la moyenne des températures avec les requis suivants:

- l'envoi des mesures devra être fait chaque 10 minutes ;
- l'incertitude des mesures est égale à 10% ;
- la longueur des zones considérées comme une seule zone est fixée à un mètre ;
- la durée totale de la collecte est fixée à 100 jours.

L'application envoie au nœud collecteur *S* la requête suivante :

```
SELECT AVG(temp) FROM sensors
      EPOCH DURATION 10
      VALUES WITHIN 10
      ZONES WITHIN 1
      TOTAL DURATION 100
```

Le nœud *S* traduit cette requête en un message de création de grappes contenant les valeurs des paramètres (X_p= '', Y_p = '', M_p= '' , l_p = *1* , sem_p = *10*, *FlagCouverture* = *Faux*). Les nœuds N_3, N_5, N_6 et N_7, qui sont dans l'intervalle de transmission du nœud *S*, reçoivent le message de création de grappes. Le nœud N_7 calcule *FlagCouverture* qui est égal à *Vrai*, assigne ses valeurs à X_p, Y_p *et* M_p , et envoie le message aux nœuds N_1, N_{11} et N_{12}. Dans ce cas, la grappe 1 est formée sans exécuter la procédure de négociation. De la même manière, la grappe 2 est construite.

La formation de la grappe 4 nécessite l'exécution de la procédure de négociation. En effet, le nœud N_5 ne peut pas assurer la couverture de la grappe pendant la durée spécifiée (100 jours) et envoie le message de création de grappes avec *FlagCouverture* = *Faux*. Ainsi, il commence la procédure de négociation. Le nœud N_2 reçoit le message, calcule *FlagCouverture* qui est égal à *Vrai*, et retourne un message au nœud N_5 avec les valeurs des paramètres ($d_{2,5}$ = 0.8 et M_2=23).

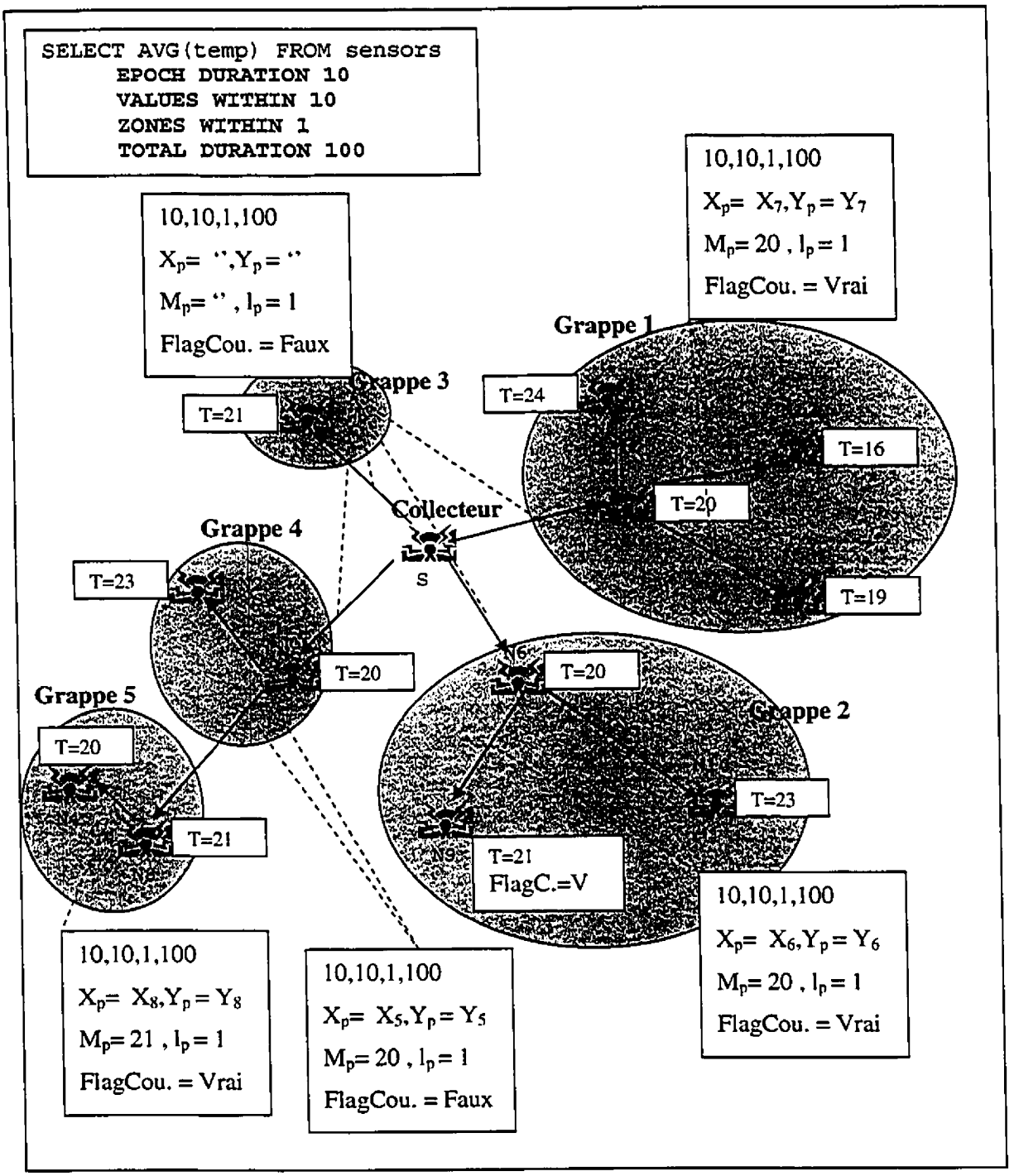

Figure 3.11 Exemple de formation de grappes

Le nœud N_7 procède de la même manière et envoie (($d_{8,5}$ = 1.2 et M_8=21)). Ensuite, le nœud N_5 calcule $l_5 = MAX\left(1, MIN(0.8,1.2)\right) = 1$, calcule $sem_5 = MAX\left(10, MIN(3,1)\right) = 10$ et envoie le message de création de grappe avec les valeurs des paramètres (X_p= X_5 ,Y_p = Y_5, M_5= *20* , l_5 = *1* , sem_5 = *10*, *FlagCouverture* = *Vrai*). Même si le nœud N_8 a participé à la procédure de négociation, il ne fait pas partie de la grappe 4 parce que sa distance ne respecte pas la troisième condition, i.e. $d_{8,5}$ (=1.2) est inférieure à l_5 (=1).

3.2.4. Phase de collection

Après être formées, les différentes grappes devront fournir une mesure collective au nœud collecteur *S* à chaque période de temps. Dans une première phase, la tête de grappe sera statique et déterminée de la manière suivante :

- Si le premier nœud *p* qui a commencé la formation de la grappe peut assurer la couverture de la zone pendant la durée totale de la collecte de mesures, alors la tête de grappe sera le nœud *p* ;
- Sinon, le nœud *p* choisira un nœud de grappe qui peut assurer la couverture lors de l'exécution de la procédure de négociation.

Pour l'exemple précédent, la tête de la grappe 1 est le nœud N_7 parce qu'il est le premier nœud de la grappe et il peut assurer sa couverture. La tête de la grappe 4 est le nœud N_2. Le nœud N_5 le choisit durant la procédure de négociation parce qu'il fait partie de la grappe 4 et peut assurer sa couverture pendant 100 jours.

Durant la phase de *collection*, la tête de grappe se chargera de coordonner la collecte des mesures des nœuds appartenant à sa grappe, de filtrer les données considérées redondantes, de calculer les fonctions d'agrégation et d'envoyer le résultat au nœud collecteur *S* à chaque période de temps.

À la fin de l'exécution de la procédure de création de grappes, le nœud collecteur reçoit les coordonnées des têtes de grappes et leurs longueurs. Il peut ainsi déterminer l'ordre de précédence des grappes durant la phase de la collecte des mesures, i.e. les chemins que les mesures vont emprunter pour arriver au nœud collecteur. Le nœud S communique à chaque tête de grappe, le numéro de grappe qui doit lui envoyer sa mesure collective. Ainsi, une hiérarchie en arbre est définie pour les têtes de grappes. Les grappes qui n'ont pas de

précédents sont appelées les grappes *feuilles*. Les grappes *mères* sont celles qui ont des grappes précédentes appelées *filles*.

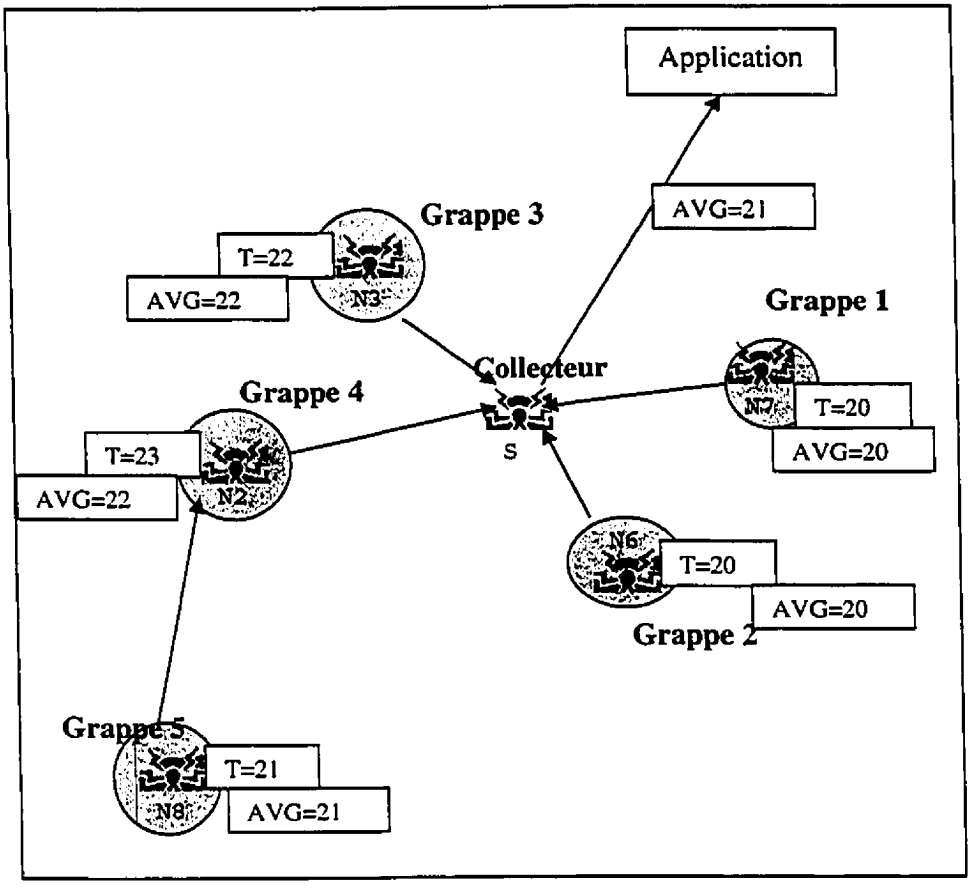

Figure 3.12 Exemple de la phase de collection

À chaque expiration d'une période de temps, les têtes des grappes *feuilles* envoient leurs mesures à leurs grappes *mères*. Celles-ci attendent la réception des mesures de ses grappes filles, calculent les fonctions d'agrégation et envoient les résultats aux grappes suivantes, et ainsi de suite jusqu'au nœud collecteur *S*. Vu la manière dont les grappes sont créées, seules les têtes de grappes envoient leurs mesures. Les données des autres nœuds de grappes sont filtrées puisqu'elles sont considérées comme redondantes et ne satisfont pas les requis de l'application en terme de QdS. Ces nœuds doivent assurer le rôle de passerelle pour les messages qui sont envoyés dans le réseau.

La Figure 3.12 présente la phase de la collection des mesures pour l'exemple illustré à la Figure 3.11. Les grappes 1, 2, 3 et 5 sont des grappes *feuilles* puisqu'elles n'ont pas de grappes *filles*. La grappe 4 est la grappe *mère* de la grappe 5. À chaque période de temps, le nœud N_2, tête de la grappe 4, attend la mesure du nœud N_8, tête de la grappe 5. À la réception de cette mesure, il calcule la moyenne des températures et envoie le résultat de l'agrégation au nœud collecteur. Les nœuds N_3, N_6 et N_7 envoient leurs mesures au nœud collecteur S. Celui-ci calcule la moyenne de toutes les données et transmet le résultat à l'application.

CHAPITRE IV
OPTIMISATION DE LA FORMATION DES GRAPPES

La phase de la distribution de la requête du mécanisme de la collecte des données utilise la notion de grappe pour pouvoir réduire le nombre de messages échangés dans le réseau et ainsi diminuer la dissipation des énergies par les capteurs. Le chapitre III présente une approche répartie de la formation de grappes où tous les nœuds participent à la création de grappes et à la désignation des têtes de grappes. Mais, cette approche n'assure pas une formation optimale des grappes. Par conséquent, nous avons conçu une deuxième approche centralisée où un nœud central utilise un modèle mathématique pour optimiser la formation des grappes.

Dans ce chapitre, nous allons présenter d'une manière sommaire les approches centralisées et réparties de la formation de grappes existantes. Ensuite, nous décrirons la formulation mathématique qui modélise cette optimisation. Puis, nous présenterons une première résolution basée sur *la recherche taboue*. Enfin, nous détaillerons une deuxième approche basée sur une méthode exacte de résolution des problèmes en nombre entier.

4.1. Formation de grappes dans les réseaux de capteurs

Les algorithmes de formation de grappes peuvent être subdivisés en deux catégories : des algorithmes centralisés et des algorithmes répartis. Les algorithmes centralisés se basent sur l'idée qu'un nœud de réseau central possède les informations nécessaires sur tous les nœuds du réseau. Ces informations permettent de former les grappes et de déterminer leurs têtes d'une manière optimale. L'inconvénient majeur de cette catégorie d'algorithmes est la surcharge de communication due au fait que chaque nœud de réseau doit envoyer ses informations au nœud central. L'autre inconvénient est le temps pris par le nœud central pour optimiser la formation de grappes vu que généralement le nœud central exécute un algorithme heuristique qui admet comme paramètres d'entrée les informations sur tous les nœuds.

4.1.1. Les algorithmes centralisés de formation de grappes

Tous les algorithmes rencontrés dans la littérature déterminent les grappes en sachant préalablement le nombre de grappes à former. Dans certains cas, les têtes de grappes sont déterminées dans une première phase et les autres nœuds sont assignés aux différentes grappes dans une deuxième phase. Généralement, ces algorithmes partent du fait que la formation de grappes est considérée comme une partition de graphe avec des contraintes particulières qui rend le problème NP-Complet [BOL85].

Ghiasi, et al [GHI02], et Kanugo et al. [KAN02] ont basé leurs approches de formation de grappes sur le fait que ce problème peut être modélisé par celui de "k-means clustering". Par conséquent, leurs algorithmes sont basés sur les heuristiques de la résolution du problème de « k-means clustering » , ce problème qui se résume de la manière suivante :

> *Étant donné un ensemble P de n points dans un espace de dimension d (R^d), et un nombre entier k, le problème est de déterminer un ensemble de k points dans R^d, appelés centres, qui minimise la moyenne des carrés des distances euclidiennes entre chaque point et le centre le plus proche.*

Heinzelman *et al.* proposent une version centralisée de leur protocole de collecte de données basé sur la formation de grappes LEACH [HEI02] afin de produire une meilleure répartition des têtes de grappes sur l'ensemble de réseau. Ils ont proposé cette version centralisée vu que leur protocole réparti ne garantit pas un meilleur contrôle sur l'emplacement et le nombre de têtes de grappes. Dans une première phase, les nœuds envoient les informations concernant leur emplacement ainsi que leurs niveaux d'énergie au nœud collecteur. Celui-ci utilise ces niveaux d'énergie pour déterminer les têtes de grappes. Ensuite, il détermine les grappes optimales qui minimisent l'énergie utilisée par les autres nœuds pour envoyer leurs données aux têtes de grappes, et ceci en minimisant la somme totale des carrés des distances entre les nœuds et les têtes de grappes les plus proches. Les auteurs utilisent les résultats de Agarwal *et al.* qui proposent des algorithmes permettant de résoudre le problème de formation de grappes connaissant un ensemble de nœuds centres [AGA02]. Le problème se résume de la manière suivante:

Étant donné un ensemble de n point dans un espace métrique de dimension d (R^d, ρ), et un entier positif k <= n. k-clustering de S est la répartition Σ de S en k sous ensembles $S_1,...S_k$. La taille de la grappe S_i est la distance maximale (suivant la métrique ρ) entre un point fixe c_i, appelé centre de la grappe S_i, et un point de S_i. La taille de la répartition Σ est la taille maximale de ses grappes. Le problème est de trouver une répartition dont la taille est minimale.

Dans notre cas, ces résultats ne peuvent pas être utilisés, vu que le nombre de grappes n'est pas prédéterminé et que les têtes de grappes ne sont pas connues préalablement à la formation de grappes.

4.1.2. Les algorithmes répartis de formation de grappes

La deuxième catégorie contient les algorithmes répartis sur tous les nœuds. Tous les nœuds exécutent l'algorithme de la formation de grappes et la détermination des têtes de grappes. L'inconvénient de ces algorithmes est que généralement les nœuds ont une vision restreinte à leur entourage. Par conséquent, les grappes ne sont pas formées d'une manière optimale. Le premier algorithme que nous avons présenté fait partie de cette catégorie.

Dans [MOU06], Moussaoui *et al.* proposent un algorithme de formation de grappes basé sur trois critères :

1) le nombre de nœuds d'une grappe qui équilibre leurs tailles ;

2) les distances entre les nœuds qui permettent de réduire la consommation d'énergie ainsi que les interférences entre les nœuds ;

3) le niveau d'énergie de chaque nœud pour pouvoir répartir la charge d'une manière raisonnable.

L'algorithme de la formation de grappes est exécuté par tous les nœuds du réseau. La tête de grappe est élue parmi les nœuds de la grappe suivant leurs niveaux d'énergie. Les nœuds choisissent celui qui a un maximum d'énergie restante vu que la tête de grappe effectue des tâches qui nécessitent un niveau d'énergie élevé.

LEACH [HEI02] est un protocole de collecte de données qui utilise la notion de grappes pour pouvoir prolonger la durée de vie des réseaux de capteurs. Dans une première phase, certains nœuds du réseau s'élisent eux-mêmes comme têtes de grappes en se basant sur des probabilités prédéterminées et l'historique des têtes de grappes. Ensuite, les têtes de grappes collectent les données des autres membres de grappes, effectuent des agrégations de données, et communiquent les résultats au nœud collecteur jusqu'à ce qu'une nouvelle procédure de formation de grappes soit déclenchée. La formation des grappes s'effectue de la manière suivante : après avoir effectué les élections des têtes de grappes, celles-ci envoient un message aux autre nœuds voisins contenant leurs identifiants. En se basant sur la puissance de signal reçu, les autres nœuds déterminent leurs grappes de manière à minimiser l'énergie nécessaire à la communication avec les têtes de grappes. Cette version de protocole souffre de deux inconvénients triviaux. D'une part, le protocole ne garantit pas une répartition optimale de la fonction de tête de grappe et l'énergie restante ne fait pas partie du mécanisme de l'élection des têtes de grappes. D'autre part, ce protocole ne peut pas assurer une meilleure extensibilité dans le cas où le réseau déploie un grand nombre de nœuds. Les auteurs supposent que les têtes de grappes devront acheminer leur message directement au nœud collecteur.

L'approche de Younis et al. [YOU04] est dite hybride : dans une première phase, les têtes de grappes sont choisies d'une manière aléatoire suivant leur énergie résiduelle. Puis, les autres nœuds choisissent leurs grappes tout en minimisant le coût de la communication. Les têtes de grappes sont choisies en se basant sur l'énergie résiduelle et d'autres paramètres (par exemple, la proximité des nœuds avec leurs entourages, la connectivité de réseau). La formation des grappes a pour but de maximiser la durée de vie du réseau et assurer une meilleure extensibilité et une bonne balance de charge entre les nœuds. Ils supposent qu'un nœud a un nombre connu de niveaux de puissance de transmission. Celle-ci est utilisée pour définir le rayon d'une grappe et le niveau moyen de puissance utilisée pour la communication à l'intérieur des grappes.

Raghuwanshi et al. définissent une grappe comme étant un ensemble de nœuds qui se situent à une distance d'un hop entre eux et qui collaborent pour compléter une tâche en commun [RAG03]. Dans leur approche, les nœuds découvrent leur entourage dans une phase initiale et commencent la formation des grappes. Initialement, le nœud du milieu dans une

grappe devient la tête de celle-ci. Chaque nœud choisit de devenir membre de la grappe qui a la tête la plus proche du nœud selon la puissance du signal de transmission. Les auteurs justifient le choix de la puissance de signal au lieu de la distance spatiale entre les nœuds par le fait que les capteurs peuvent être déployés dans des environnements qui perturbent la transmission des signaux.

4.2. Notre approche centralisée de formation de grappe

Dans la section 3 du chapitre 3, nous avons expliqué les critères de la formation de grappes. L'idée générale de notre approche est de pouvoir rassembler dans une seule grappe les nœuds du réseau susceptibles de prendre des mesures considérées redondantes par rapport au besoin des applications, de maximiser la couverture de toutes les zones où le réseau est déployé et de maximiser la durée de vie des capteurs en minimisant la consommation d'énergie. La formulation mathématique des critères de formation des grappes est la suivante :

1. $\left(\bigcup_i G_i = V\right) \wedge \left(\forall (i,j)\ G_i \cap G_j = \phi\right)$ (4.1)
2. $\forall N_j, N_l \in G_i \quad \left|M_j(t) - M_l(t)\right| \leq sem$ (4.2)
3. $\forall N_j, N_l \in G_i \quad d_{j,l} \leq pas$ (4.3)
4. $\exists N_j \in G_i \qquad E_j^T \leq E_j^{restante}$ (4.4)

Les différents termes de cette formulation sont expliqués dans la section 3 du chapitre 3. Ce problème est un problème de partitionnement d'un hypergraphe avec des contraintes sur les hyper arêtes comme nous allons expliquer par la suite.

4.2.1. Définitions générales de la répartition des hypergraphes

Définition 1 [GON85]: Soit un ensemble $S = \{s_1, s_2, \ldots, s_n\}$ et une famille $E = \{E_1, E_2, \ldots, E_m\}$ de parties de S. E constitue un hypergraphe sur S si nous avons :

$$E_j \neq \Phi \qquad \text{j de 1 à m}$$

$$\bigcup_j E_j = S$$

Le couple $H = (S, E)$ s'appelle un *hypergraphe*. Les éléments $s_1,s_2, \ldots ,s_n$ s'appellent les sommets de l'hypergraphe, et les éléments $E_1,E_2, \ldots ,E_m$ s'appellent les arêtes de l'hypergraphe, ou les *hyperarêtes*.

Définition 2 [GON85]: Étant donné un hypergraphe $H = (S,E)$ avec $S = \{s_1,s_2, \ldots ,s_n\}$ et $E = \{E_1,E_2, \ldots ,E_m\}$. Étant donné un coût $c_i = c(E_i)$ associé à chaque sous ensemble E_i. Le problème de répartition de H consiste à trouver une sous-famille F de E qui répartit l'ensemble S tel que :

$$\bigcup_{E_j \in F} E_j = S$$

$$\forall E_k \in F, \forall E_j \in F, E_k \cap E_j = \Phi \quad si\ k \neq j$$

et en minimisant le coût total $\sum_{E_j \in F} c_j$

Soit $A = (a_{ij})$ *la matrice d'incidence* de l'hypergraphe H définie par :

$$a_{ij} = \begin{cases} 1 & si\ s_i \in E_j \\ 0 & sinon \end{cases}$$

Le sous-ensemble F de E peut être représenté par son *vecteur caractéristique* $x = (x_1, \ldots, x_n)$ tel que :

$$x_j = \begin{cases} 1 & si\ E_i \in F \\ 0 & sinon \end{cases}$$

Alors, x représente un *partitionnement* si et seulement si :

$$\sum_{j=1}^{n} a_{ij} x_j = 1 \quad i\ de\ 1\ à\ m$$

$$x_j = 0\ ou\ 1 \quad j\ de\ 1\ à\ n$$

Ainsi, le problème de partition d'un hypergraphe à coût minimum se formalisera comme étant la recherche de la solution en nombre entier du problème linéaire suivant :

$$\begin{cases} \min z = \sum_{j=1}^{n} c_j x_j \\ avec \sum_{j=1}^{n} a_{ij} x_j = 1 \quad i \; de \; 1 \; à \; m \\ x_j = 0 \; ou \; 1 \qquad\qquad j \; de \; 1 \; à \; n \end{cases} \tag{4.5}$$

Ce problème est connu comme étant un problème NP-Complet [GON85].

4.2.2. Formulation de la formation de grappes

Étant donné la définition de la problématique que nous avons définie dans les sections précédentes et la définition de partitionnement d'un hypergraphe, le problème de trouver l'ensemble de grappes qui satisfont les critères de (4.1) à (4.4) et qui minimisent la consommation d'énergie totale du réseau consiste à trouver un partitionnement d'un hypergraphe qui minimise le coût total. Pour cette raison, il faut tout d'abord définir les sommets et les arêtes de l'hypergraphe, trouver le coût de chaque arête et formaliser les critères ((4,1) ... (4,4)). Enfin, et puisque le problème est NP-Complet, il faut trouver une heuristique qui permet d'approcher la solution optimale.

Modèle mathématique général

Afin de formaliser le quatrième critère (4.4) de la formation des grappes comme étant une contrainte du problème du partitionnement en nombre entier, nous avons introduit une nouvelle variable qui représente le fait que le nœud peut assurer la couverture de sa zone durant toute la durée de l'exécution de la requête. Cette variable s'écrit de la manière suivante :

$$e_i = \begin{cases} 1 \; si \; E_i^T \leq E_i^{\text{restante}} \\ 0 \; sinon \end{cases}$$

Alors le critère de l'équation (4.4) peut s'écrire de la façon suivante :

$$\sum_{i=1}^{m} a_{ij} e_i \geq 1 \quad j \; de \; 1 \; à \; n \tag{4.6}$$

Expression du coût d'une grappe

Le coût d'une grappe (celui de l'arête de l'hypergraphe) doit prendre en considération le fait que le but de notre problème est de minimiser la consommation d'énergie et de prolonger la durée de vie de réseau. Ainsi, les grappes faisant partie de la solution optimale devront, d'une part, minimiser le nombre de messages échangés dans le réseau, et d'autre part, minimiser les énergies de capture, de traitement et d'envoi. Pour ce faire, nous considérons les deux volets : le nombre de nœuds dans la grappe et la localisation de la tête de grappe. Les grappes devront être choisies de manière à contenir le plus grand nombre de nœuds. En effet, plus la grappe contient de nœuds, moins sont nombreux les messages générés et moins sont élevées les énergies de capture et de traitement. Ceci est dû au fait que les nœuds de la grappe qui ne jouent pas le rôle de la tête de grappe ne participent ni aux captures des mesures ni aux traitements.

Le deuxième volet qu'il faut prendre en considération dans la détermination des coûts des grappes est celui des localisations des têtes de grappes. Nous rappelons que le long de l'exécution de la requête, les têtes de grappes capturent les mesures, les envoient aux autres têtes de grappes à travers les autres nœuds, et participent aux différents calculs des agrégats. Par conséquent, plus les têtes des grappes sont proches du nœud collecteur en terme de nombre de nœuds intermédiaires, moins d'énergie est consommée pour envoyer les mesures. Ainsi, nous proposons la fonction de coût c_j de la grappe j comme suit :

$$c_j = f_1(n_j) + f_2(t_j)$$

où n_j est le nombre de nœuds dans la grappe j et t_j est le nœud tête de la grappe j.

Nous commençons par le deuxième terme; $f_2(t_j)$ présente le deuxième volet de la fonction de coût. Du fait que nous voulons minimiser la consommation d'énergie, cette partie de coût est une sommation :

- de l'énergie de capture des mesures par la tête de grappe j ;
- des énergies de traitement consommées par les différentes têtes de grappes qui font partie du chemin reliant la tête de la grappe j au nœud collecteur ;
- de l'énergie totale consommée par l'acheminement des messages de la tête de la grappe j vers le nœud collecteur.

En prenant en considération la formule 7 du chapitre II, elle pourra s'écrire de la manière générale suivante :

$$f_2(t_j) = hop_{js} E_{txelec} + \sum_{i=j}^{\lambda} \left(\varepsilon_{amp} d_{i,i+1}^{\alpha} + N_{i,i+1}^{\alpha} E_{rxelec} \right) + \sum_{i=j}^{\lambda} \left(E_{TraitementTête}^{i} \right) + E_{Capture}^{j}$$

où $\left(E_{TraitementTête}^{i}\right)$ est l'énergie consommée par une tête de grappe i qui se trouve dans le chemin permettant de communiquer les mesures de la tête de la grappe j au nœud collecteur. $E_{Capture}^{j}$ est l'énergie consommée par la tête de la grappe j lors de la capture des mesures. Nous rappelons les significations de quelques termes : hop_{js} est le nombre de nœuds qui séparent le nœud j du nœud *collecteur* s. Cet ensemble de nœuds présente la route que le message emprunte lors de sa transmission du nœud j vers le nœud s, et dépend des protocoles de routage mis en place. $d_{i,i+1}$ est la distance entre deux nœuds successifs dans cette route. $N_{i,i+1}$ est le nombre de nœuds voisins qui pourront sur-écouter les messages envoyés du nœud i vers le nœud $i+1$.

Certes, cette formule est générale et prend en considération tous les types d'énergies consommées; mais elle est complexe et son calcul sera difficile dans la pratique vu qu'elle dépend des protocoles de routage mis en place. Par conséquent, il s'avère nécessaire de la simplifier et l'approximer par une formule moins complexe. L'énergie de la transmission des messages est toujours considérée comme prépondérante par rapport aux autres consommations d'énergie (e.g. l'énergie de traitement et l'énergie de réception des messages) [LIA07]. En ce qui concerne les termes reliés au nombre de hops existant entre le nœud j tête de grappe et le nœud collecteur s, nous considérons que la distance entre ces deux nœuds donne une bonne approximation de ces termes vu que plus le nœud est éloigné du nœud collecteur, plus le nombre de hop augmente. En plus, dans un réseau dense, le protocole de routage peut trouver facilement une route dont la somme entre les distances des nœuds intermédiaires est proche de la distance entre le nœud tête de grappe et le nœud collecteur. La Figure 4.1 illustre cette approximation. Enfin, en considérant uniquement la distance entre le nœud j et le nœud s, cette approximation de coût est parfaitement valide dans le cas d'un modèle de réseau où les nœuds communiquent directement avec le nœud collecteur. Par conséquent, le terme $f_2(t_j)$ est approximé par $f_2(t_j) = \delta\, d_{j,s}^{\alpha}$ où δ est un coefficient positif.

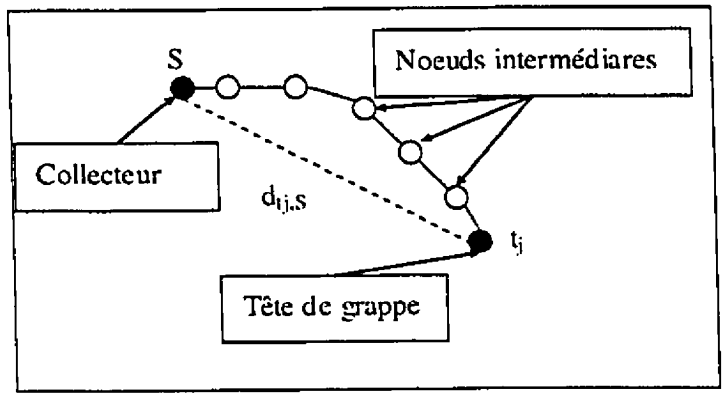

Figure 4.1 Approximation des distances

Le premier terme de la fonction de coût d'une grappe est $f_1(n_j)$. Il présente le gain en consommation d'énergie qu'une grappe fait épargner en filtrant les messages des nœuds de la grappe qui ne jouent pas le rôle de la tête de grappe. L'énergie gagnée par un nœud de la grappe peut être approximé de la même manière que l'énergie consommée par la tête de grappe expliquée précédemment. Elle est approximée par $\beta\, d_{j,s}^{\alpha}$. D'où l'approximation du premier terme de la fonction de coût par $f_1(n_j) = -\beta \sum_{i=1}^{n_j} d_{\theta i,s}^{\alpha}$. θi est l'indice d'un nœud de la grappe j. (Grappe j = $\{N_{\theta i} / i$ entre 1 et n$\}$) Le signe moins dans cette formule représente le fait qu'à chaque nœud appartenant à la grappe, le mécanisme gagne sur la consommation d'énergie. β est un coefficient positif. Par conséquent, le coût total d'une grappe peut être approximé par la formule suivante :

$$c_j = \delta\, d_{j,s}^{\alpha} - \beta \sum_{i=1}^{n_j} d_{\theta i,s}^{\alpha} \qquad (4.7)$$

D'après les relations (4.2) (4.3) (4.5) (4.6) et (4.7), le programme linéaire en nombre entier général peut s'écrire de la manière suivante :

$$\begin{cases} \min z = \sum_{j=1}^{n} \left(\delta\, d^{\alpha}_{j,s} - \beta \sum_{i=1}^{n_j} d^{\alpha}_{ik,s} \right) x_j \\ avec \sum_{j=1}^{n} a_{ij} x_j = 1 & i\ de\ 1\ à\ m \\ x_j \in \{0,1\} & j\ de\ 1\ à\ n \\ |M_b(t) - M_c(t)| \le sem & \forall s_b, s_c \in E_j,\ j\ de\ 1\ à\ n \\ d_{b,c} \le pas & \forall s_b, s_c \in E_j,\ j\ de\ 1\ à\ n \\ \sum_{i=1}^{m} a_{ij} e_i \ge 1 & j\ de\ 1\ à\ n \end{cases} \quad (4.8)$$

Le problème linéaire général (4.8) contient un nombre fulgurant de variables et de contraintes. En effet, le nombre de variables correspondent au nombre des arêtes de l'hypergraphe considéré. Une approximation naïve de ce nombre, qui est aussi une majoration, est la cardinalité de l'ensemble des parties $P(S)$ de l'ensemble S. Si la cardinalité de S est égale à m, alors la cardinalité de $P(S)$ est égale à 2^m. Par exemple, si le réseau de capteurs contient 1000 nœuds, alors la cardinalité de $P(S)$ est 10^{300}. Le nombre de contraintes égale à $(m + 2n + mn)$ qui est aussi trop élevé. D'où l'idée de choisir un hypergraphe avec un ensemble d'arêtes restreint qui permet de réduire le nombre de variables et le nombre de contraintes.

Définition de l'hypergraphe

Pour simplifier le modèle (4.8) décrit dans la section précédente, nous définissons l'hypergraphe $H^r = (S, E^r)$ de la manière suivante. L'ensemble S des sommets de l'hypergraphe est l'ensemble des nœuds du réseau. L'ensemble E^r des arêtes de l'hypergraphe est l'ensemble réduit des parties de S qui satisfont les critères (4.2), (4.3) et (4.4). Si le partitionnement est calculé sur cet ensemble restreint, alors c'est clair que les trois dernières contraintes du modèle (4.8) vont disparaître du modèle vu que tous les sous-ensembles de E^r vont les satisfaire.

Pour construire l'ensemble des éléments de E^r, nous définissons un nouveau graphe $G^r = (S, U^r)$. L'ensemble S des sommets de G^r est l'ensemble des nœuds du réseau. Une arête est définie entre deux nœuds i et j si est seulement si les deux nœuds satisfont les deux contraintes suivantes :

$$|M_i(t) - M_j(t)| \leq sem$$
$$d_{i,j} \leq pas$$

L'ensemble réduit E' des arêtes de l'hypergraphe est défini comme étant l'ensemble des cliques du graphe G' contenant au moins un nœud qui satisfait le critère de la couverture de zone (4.4). De cette manière, l'ensemble E' ne contient que des sous-ensembles de S qui satisfont les critères (4.2), (4.3) et (4.4) de la formation des grappes. Ainsi, le problème de formation de grappes se réduit au problème (4.9) avec moins de contraintes. Le paramètre n' dans le problème est égal à la cardinalité de E'. ($n' = |E'|$).

$$\begin{cases} \min z = \sum_{j=1}^{n'} \left(\delta\, d_{j,s}^{\alpha} - \beta \sum_{i=1}^{n_c} d_{\bar{\alpha},s}^{\alpha} \right) x_j \\ avec \sum_{j=1}^{n'} a_{ij} x_j = 1 \qquad i\ de\ 1\ à\ m \qquad (4.9) \\ x_j \in \{0,1\} \qquad\qquad j\ de\ 1\ à\ n' \end{cases}$$

4.2.3. Un exemple de réseau

Le long de cette section, nous allons prendre un exemple de réseau de capteurs afin d'expliquer nos différents algorithmes. Le réseau est composé de sept nœuds (n_1, ..., n_7) et ne contient que quatre nœuds (n_1, n_3, n_4 et n_7) qui peuvent assurer la couverture de leurs zones selon le critère (4.4) de la formation de grappes. La Figure 4.2 illustre le graphe G' du réseau avec ses différentes cliques.

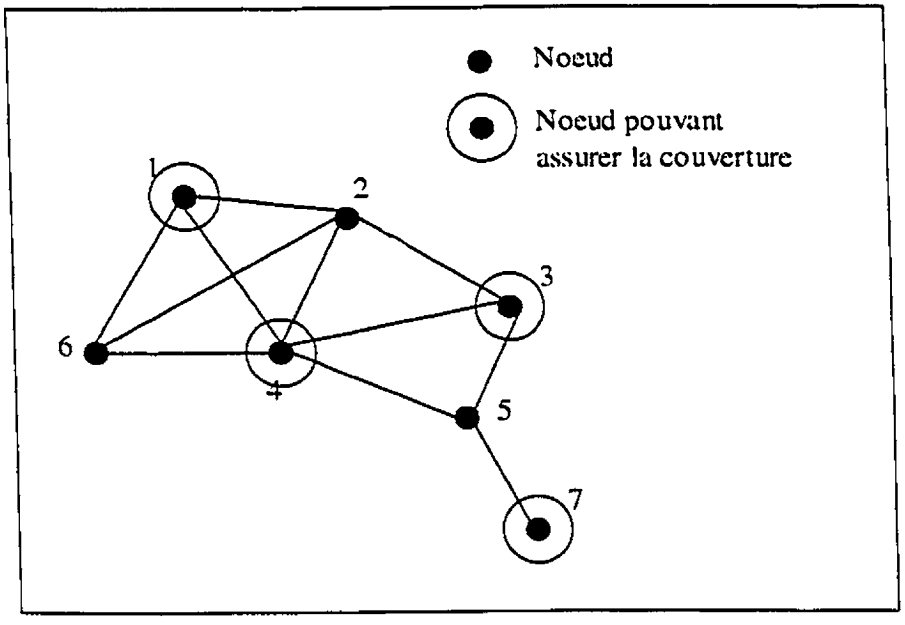

Figure 4.2 Exemple de graphe G^r

4.2.4. La résolution du problème avec la recherche taboue (RT)

Le problème de partitionnement de l'hypergraphe que nous avons défini dans les sections précédentes est un problème NP-Complet. Cette classe de problèmes est caractérisée par le fait qu'il n'existe à présent aucun algorithme qui trouve une solution optimale dans un temps polynomial. Généralement, des méthodes heuristiques et méta-heuristiques sont appliquées pour trouver une bonne solution proche de l'optimum dans un temps raisonnable. La méthode appelée *la recherche taboue* (RT) fait partie de ce type de méthodes. Dans cette section nous allons tout d'abord présenter d'une manière sommaire cette méthode. Ensuite, nous allons expliquer l'adaptation de RT à notre problème d'optimisation.

Les principes de base de la recherche taboue

La recherche taboue est une méthode de recherche locale. Elle peut être vue comme une technique itérative qui explore un ensemble de solutions, noté χ, en appliquant des mouvements d'une solution s vers une autre sclution s' faisant partie du voisinage $N(s)$ de s. Ces mouvements sont appliqués avec l'objectif d'atteindre une « bonne » solution (optimale ou proche de l'optimum) en évaluant une fonction objective $f(s)$. La méthode peut accepter des solutions qui détériorent la fonction objective. Elle utilise une mémoire dynamique,

appelée *liste taboue,* notée *T,* qui contient des solutions déjà visitées et qui sont interdites parmi celles des voisinages des solutions analysées. Cette liste, qui est actualisée à chaque itération, permet de restreindre le voisinage *N(s)* et ainsi d'éviter les cycles. Par conséquent, les solutions qui sont analysées dans une itération sont celles appartenant à *N(s) – T.* L'algorithme général de la recherche taboue est illustré à la Figure 4.3 .

```
1.  Initialisation : Choisir une solution initiale s dans χ.
2.  s* := s (s* est la meilleure solution obtenue jusqu'ici)
3.  T := Ø (T est la liste taboue)
4.  Continuer := vrai
5.  Tant que Continuer faire
6.          Si une condition d'arrêt est rencontrée
7.                  alors Continuer := faux
8.          Sinon
9.                  Générer V = N(s)
10.                 Trouver la meilleure solution s' dans V (f(s')=min f(s) avec s ∈ V - T)
11.                 s := s'
12.                 Mettre à jour T
13.                 Si f(s') < f(s*) alors s* := s'
14.                 Fin si
15.         Fin si
16. Fin tant que
```

Figure 4.3 L'algorithme général de la recherche taboue

La liste taboue est construite en ajoutant des solutions visitées à chaque itération jusqu'à ce que la taille de la liste atteigne une certaine limite. Après cette limite, les solutions les plus récentes remplacent les plus anciennes. Une solution peut sortir de la liste taboue si elle satisfait certains critères, appelés critères *d'aspiration.* Afin de rendre la méthode plus intelligente, deux processus peuvent être rajoutés à la méthode : l'intensification et la diversification. Le processus d'intensification consiste à dégager les propriétés communes à toutes les bonnes solutions visitées et à utiliser ces propriétés pour trouver les bonnes solutions dans les itérations futures. Le processus de la diversification est une notion complémentaire de l'intensification qui consiste à diversifier les régions de la recherche dans l'espace des solutions. Ainsi, la recherche taboue utilise l'intensification pour accroître la

recherche dans les régions prometteuses et la diversification pour considérer des régions opposées. Une description générale de la méthode peut être trouvée dans [GLO93].

Les conditions d'arrêt sont nombreuses et les plus utilisées sont une combinaison des conditions suivantes :

- Une solution optimale est atteinte ;
- $N(s) - T = \emptyset$;
- Un maximum de nombre d'itérations est atteint ;
- Un maximum de nombre d'itérations qui n'améliorent pas la bonne solution est atteint.

Adaptation de la recherche taboue au problème de formation de grappes

Afin d'adapter la recherche taboue à la résolution du problème combinatoire que nous avons défini, les étapes suivantes devront être suivies :

1. Trouver l'algorithme permettant d'obtenir la solution initiale ;
2. Définir le voisinage $N(s)$ d'une solution s: L'ensemble V des $m(s)$ où $m(.)$ est le mouvement qui permet de passer d'une solution s à une autre solution s' définie dans son voisinage ;
3. Décrire le type et la taille de la liste taboue préconisée ;
4. Définir les critères d'aspiration ;
5. Proposer des améliorations de la méthode : Intensification et diversification.

Algorithme permettant d'obtenir la solution initiale

L'objectif de cet algorithme est de pouvoir trouver une solution initiale à notre problème. Le choix d'une bonne solution initiale permet de se rapprocher de la meilleure solution dans un temps minimal. L'algorithme que nous préconisons est présenté à la Figure 4.4. Nous commençons par trier les nœuds actifs selon leurs degrés décroissants. Dans la suite, nous appelons un *nœud actif* un nœud de réseau qui peut assurer la couverture de sa zone selon le critère (4.4) de la formation de grappes. À chaque itération, nous choisissons un nœud actif qui n'est pas encore couvert par les éléments de la solution initiale F_0. Nous ajoutons dans la solution initiale F_0 une clique de taille maximale contenant le nœud actif et

tous les autres nœuds qui n'appartiennent à aucun élément de F_0. Ce nœud actif est élu tête de cette grappe formée.

```
1. Initialisation :
2. Ordonner les nœuds actifs suivant leurs degrés
3. F0 := Ø F0 est la solution initiale)
4. Pour i = 0 à m' faire (m' est le nombre de nœuds actifs)
5.        Si le nœud i n'appartient à aucune clique de F0 alors
6.                Formation d'une nouvelle grappe g
7.                Tête de g = nœud i
8.                g = clique de taille maximale contenant i
                  et les nœuds qui n'appartiennent à aucun
                  élément de F0
9.                F0 = F0 U g
10. Fin si
11. Fin pour
```

Figure 4.4 Algorithme de détermination de la solution initiale pour la recherche taboue

La Figure 4.4 illustre cet algorithme. Si nous considérons l'exemple introduit dans la section 4.2.3, l'algorithme commence par le nœud 4 qui est actif et son degré égale 5. La grappe trouvée est {1,2,4,6} avec le nœud 4 jouant le rôle de la tête. F_0={{1,2,4,6}}. La deuxième itération choisit le nœud actif 3 qui n'est pas encore couvert. La clique de taille maximale, formée du nœud 3 et les nœuds qui ne sont pas encore couverts par F_0, forme la grappe {3,5}. F_0= {{1,2,4,6}, {3,5}}. La dernière grappe trouvée est {7}. F_0={{1,2,4,6}, {3,5},{7}}. Les têtes de ces grappes sont les nœuds 1, 3 et 7.

Cet algorithme ne garantit pas que chaque nœud ordinaire soit affecté à une grappe. De ce fait, si un nœud *i* n'appartient à aucune grappe à la fin de l'algorithme, il est affecté à une grappe dont un nœud est adjacent au nœud *i*. Ainsi, la solution initiale peut être non réalisable, i.e. une grappe ne forme pas une clique. Pour évaluer cette solution, nous proposons d'appliquer une pénalité qui sera détaillée un peu plus loin lors de l'analyse des voisinages d'une solution.

Définition des voisinages N(s) d'une solution s

La définition de voisinage *N(s)* est une étape cruciale dans la recherche taboue vu qu'elle détermine la qualité de la solution trouvée et influence le temps d'exécution de la méthode. Afin de définir ce voisinage, nous distinguons entre deux types de mouvements : un mouvement impliquant un nœud actif et un autre impliquant un nœud ordinaire, i.e. un nœud qui ne peut pas assurer la couverture de sa zone. Cette distinction est justifiée par le fait que le nœud actif peut jouer le rôle de la tête de grappe, et ainsi il peut former une nouvelle grappe. Nous introduisons un troisième mouvement impliquant une tête de grappe et qui permet d'éliminer une grappe de la solution.

Mouvement impliquant un nœud ordinaire :

Soit *s* la solution dont on veut calculer le voisinage *N(S)*. Nous définissons le mouvement $m(\alpha,t)$ où α est un nœud ordinaire et l'indice t présente la tête d'une grappe de la manière suivante : Le nœud α qui était affecté à la grappe E_i dans *s*, se réaffecte à une autre grappe E_j dont la tête t est adjacente au nœud α. Cette affectation pourra engendrer une grappe qui ne forme par nécessairement une clique. Lors de cette nouvelle affectation, le nœud α peut ne pas avoir de lien avec un ou plusieurs nœuds de la grappe. Nous définissons un coût qui pénalise les mouvements impliquant des grappes non réalisables par le terme $P_{(\alpha,j)}$ qui présente le nombre de nœuds de cette grappe n'ayant pas une arête avec le nœud α. Ainsi la fonction permettant d'évaluer une solution obtenue lors du mouvement $m(\alpha,t)$ dans le voisinage *N(s)* s'écrit de la manière suivante :

$$f'(\alpha,t) = \sum_{j=1}^{|s|} c_j x_j + \sigma P_{(\alpha,t)} \qquad (4.10)$$

Le coefficient σ doit être supérieur à 1 et élevé de manière à ce que le choix de la solution soit le plus possible réalisable. Il doit être ajusté d'une manière dynamique. La Figure 4.5 illustre le mouvement $m(5,7)$ appliqué sur l'exemple de la section précédente. Le nœud 5 est adjacent aux deux têtes de grappes 3 et 7. Il était affecté à la grappe dont la tête est 3. Le mouvement $m(5,7)$ consiste à l'affecter à la grappe dont la tête est le nœud 7. La solution engendrée est {{1,2,4,6},{3},{5,7}}. Vu que tous les nœuds de la grappe {5,7} ont une connexion avec le nœud 5, la pénalité est égale à zéro ($P_{(5,7)} = 0$) dans cette itération.

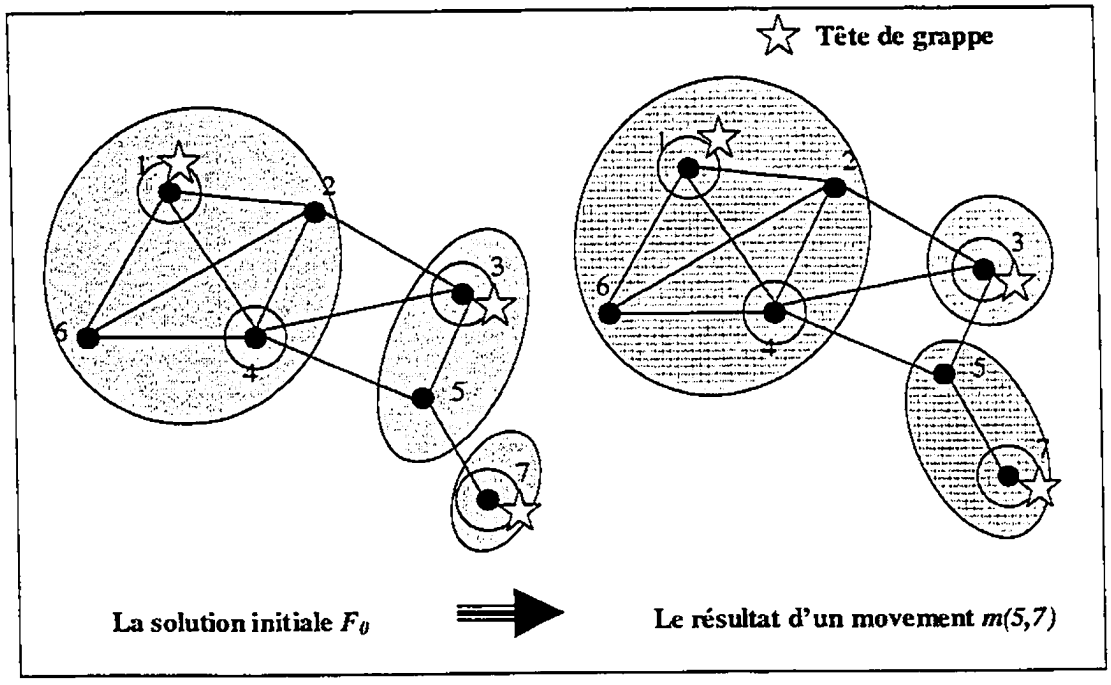

Figure 4.5 Exemple de mouvement impliquant un nœud ordinaire

Mouvement impliquant un nœud actif:

Le deuxième type de mouvement que nous proposons concerne un nœud actif α. Celui-ci a la particularité de jouer le rôle d'une tête de grappe, et ainsi il peut former une nouvelle grappe. Le mouvement $m(\alpha,.)$ consiste à :

1. réaffecter le nœud α à une grappe dont la tête est adjacente à α. Ce mouvement est le même que celui que nous appliquons à un nœud ordinaire, vu qu'un nœud actif peut appartenir à une grappe sans nécessairement jouer le rôle de la tête ;
2. élire le nœud α pour devenir tête de la grappe où il est affecté. L'ancienne tête de grappe continuera à être membre de cette grappe sans jouer le rôle de la tête. Le coût de la grappe change vu qu'il dépend de la distance entre le nœud collecteur et la tête de grappe.

De cette manière, le mouvement engendre un voisinage $N(S)$ varié en analysant de nombreuses combinaisons de grappes et de tête de grappes. La Figure 4.6 illustre un exemple de mouvement impliquant le nœud actif 4. Celui-ci est affecté à la grappe dont la

tête est le nœud 3. La pénalité est égale à zéro vu que le nœud 4 a une connexion avec le nœud 5.

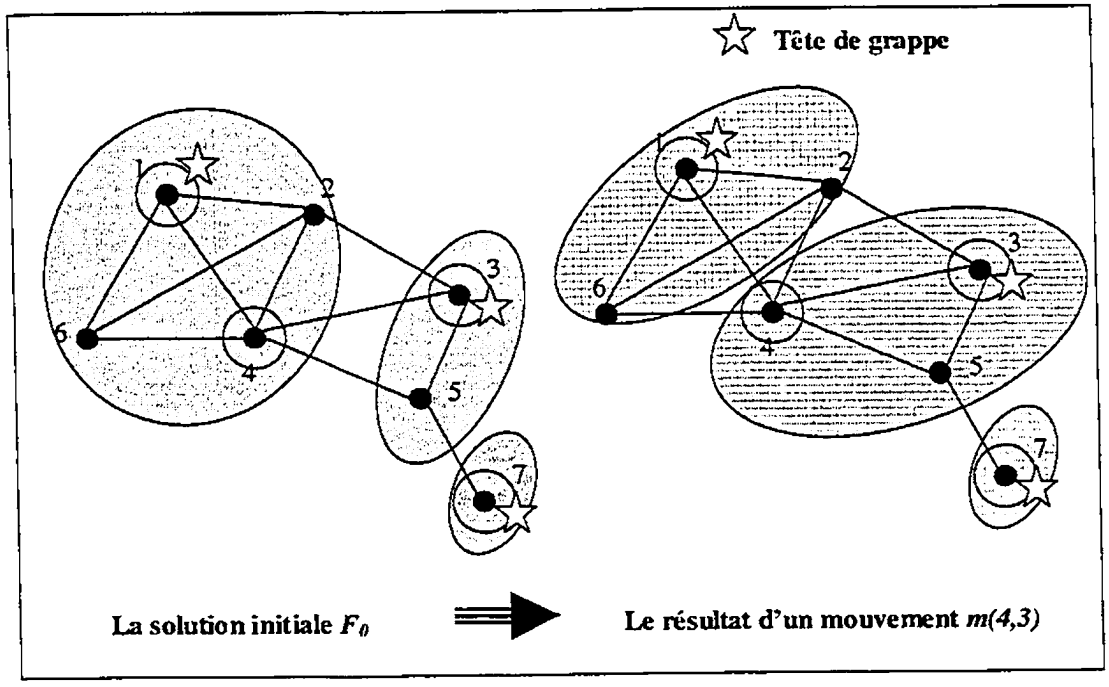

Figure 4.6 Exemple de mouvement impliquant un nœud actif

Mouvement impliquant une tête de grappe

Le troisième type de mouvement que nous proposons implique une tête de grappe existante. Dans le cas où une grappe ne contient que le nœud α qui joue le rôle de sa tête, le mouvement $m(\alpha,t)$ consiste à éliminer cette grappe de la solution et à affecter le nœud α à une grappe G_j dont la tête t est adjacente à α (Le mouvement est le même que celui impliquant un nœud ordinaire). Ainsi, ce mouvement permet d'analyser des solutions avec différent nombre de grappes. Par exemple, le mouvement $m(7,3)$ donne une solution avec deux grappes ({1,2,4,6},{3,5,7}).

Calcul des gains des solutions

Nous avons défini précédemment la fonction $f'(\alpha,t)$ qui permet d'évaluer la solution dans le voisinage $N(s)$ obtenue par le mouvement $m(\alpha,t)$. Cette fonction contient un premier terme présentant le coût des grappes et un deuxième terme présentant les pénalités. À chaque itération, il suffit de calculer le gain d'une solution par rapport à la solution de base sans

recalculer à chaque fois la fonction f', vu qu'un mouvement n'affecte que deux grappes dans le pire des cas. Ainsi, nous définissons la fonction de gain suivante :

1- Dans le cas d'une réaffectation de nœud α d'une grappe g_i dont la tete est t_i à une grappe g_j dont la tête est t_j, le gain est égal à :

$$gain = \sigma\ (P_{(\alpha,tj)} - P_{(\alpha,ti)})$$

$P_{(\alpha,tj)}$ est la pénalité associée à l'affectation du nœud α à la grappe g_j. Elle est égale au nombre de nœuds appartenant à la grappe g_j qui n'ont pas une arête avec le nœud α. Ce gain ne contient pas les coûts des grappes g_i et g_j vu qu'ils seront neutralisés mutuellement.

2- Dans le cas d'une élection de nœud i pour devenir une tête de la grappe à laquelle il appartient, le gain est égal à :

$$gain = (\delta\, d^{\alpha}_{i,s} - \beta d^{\alpha}_{\lambda,s}) - (\delta\, d^{\alpha}_{\lambda,s} - \beta d^{\alpha}_{i,s})$$

Le nœud λ est la tête de la grappe dans la solution s. Ce résultat est facilement vérifié en prenant en considération les équations 4.9 et 4.10. La différence entre les pénalités de ce mouvement est nulle vu que le nœud i ne change pas de grappe.

Liste taboue et critères d'aspiration

La recherche taboue accepte des solutions qui détériorent la meilleure solution trouvée. L'effet de cette caractéristique est d'engendrer des cycles dus à la visite des solutions déjà analysées. La liste taboue sert à restreindre le voisinage des solutions afin d'éviter ces cycles. Elle contient des attributs sur des solutions qui ne doivent pas être traitées ou bien sur des mouvements interdits. La taille de la liste taboue dépend de deux éléments : l'objet mémorisé et le nombre d'objets. Dans notre adaptation, nous proposons deux listes taboues : une liste pour l'affectation des nœuds et une deuxième liste pour les élections des nœuds actifs. La première liste taboue sert à interdire tout mouvement qui consiste à réaffecter un nœud à la même tête de grappe. Après chaque mouvement *(α,t) (t : nœud tête de grappe où α vient de se réaffecter)*, nous rajoutons à cette liste taboue la paire (α, t). La deuxième liste taboue permet d'interdire la réélection d'un nœud actif dans une même grappe. Après chaque mouvement (α,t) faisant réélire le nœud α dans la grappe g_i, nous rajoutons à la liste taboue

deux paires de nœuds. La première paire (α,tête de la grappe g_i) pour interdire le mouvement (α,t) et la deuxième paire (tête de la grappe g_i, α) pour interdire le mouvement inverse.

Une autre amélioration que nous proposons est reliée à l'initialisation de la liste taboue. En effet, les itérations peuvent retourner à la solution initiale. De ce fait, nous initialisons la liste taboue par toutes les affectations lors du calcul de la solution initiale.

Une liste taboue avec une petite taille engendre une grande fréquence de cycles, et, si la taille est grande, la qualité de la solution est détériorée vu l'interdiction d'un grand nombre de mouvements. Une taille adéquate est celle qui oscille entre ces deux extrêmes. Elle devra aussi être dynamique et dépendra de la taille de problème e.g. le nombre de nœuds dans notre cas. Nous proposons de définir trois valeurs en fonction du nombre de nœuds : petite, moyenne et grande. Nous analyserons cette taille au chapitre V.

La liste taboue peut restreindre le voisinage analysé d'une manière dramatique. Elle peut même empêcher de visiter des solutions attractives. De ce fait, la recherche taboue permet de violer les règles de la liste taboue en définissant des *critères d'aspiration*. Dans une majorité de problème, les critères d'aspiration prennent la forme suivante : si un mouvement $m(...)$ appartenant à la liste taboue engendre une solution qui améliore la meilleure solution trouvée, alors ce mouvement est pris en considération.

Améliorations de la recherche taboue : Intensification et diversification

C'est ce que nous pouvons appeler la mémoire à long terme de RT. En analysant les solutions déjà visitées, nous pouvons dégager quelques règles permettant d'intensifier ou de diversifier le voisinage. Parmi les techniques : l'alternance entre deux ou plusieurs pénalités afin d'analyser les différentes violations de contraintes relaxées, ou la définition d'autre mouvement permettant de balayer une large gamme de voisinage.

Le coefficient de la pénalité est déterminé en fonction de nombre de fois que la contrainte de clique est violée. Cette tactique permettant de déterminer dynamiquement les coefficients des pénalités entre dans le cadre des méthodes appelées les oscillations stratégiques [GLO93].

4.2.5. Résolution du problème avec une deuxième méthode

Dans cette section, nous proposons une deuxième méthode de la résolution du problème linéaire en nombre entier (4.9). L'idée de base de cette nouvelle méthode est de

limiter l'ensemble de variables et de contraintes de ce problème en définissant un ensemble de grappes réduit et de résoudre ce problème avec un algorithme basé sur une méthode exacte, par exemple le simplexe. L'objectif est de pouvoir comparer la qualité des solutions trouvées par la méthode basée sur la recherche taboue.

Le problème 4.9 a pour objectif de trouver un partitionnement de l'hypergraphe. Par conséquent, il contient des contraintes de type « égalité ». Cependant, l'utilisation des contraintes de ce type dans les problèmes d'optimisation augmente leur difficulté vu que ces contraintes seront toujours saturées. De plus, il faut déterminer un ensemble de grappes réalisables très étendu afin d'être capable de trouver des partitionnements. De ce fait, nous proposons dans cette nouvelle approche de trouver un recouvrement par Cplex au lieu de trouver un partitionnement et de raffiner la solution trouvée dans une phase post optimisation. Donc, cette nouvelle approche de la résolution du problème se déroule en trois phases :

1. La phase initiale où les grappes réalisables sont déterminées ;
2. La phase d'optimisation où Cplex calcule et retourne le recouvrement de l'hypergraphe ;
3. La phase post optimisation où un algorithme permet de calculer le partitionnement de l'hypergraphe à partir du recouvrement trouvé dans la phase précédente.

Dans la suite de cette section, nous allons, premièrement, définir l'ensemble de grappes réalisables qui va constituer l'ensemble des variables et, deuxièmement, présenter l'algorithme de la détermination du partitionnement dans la phase post optimisation.

Phase initiale : La détermination d'un ensemble de cliques réalisables du graphe G^r

Une clique dans un graphe est un sous graphe complet (il existe une arête entre chaque paire de nœuds appartenant à ce sous ensemble). Dans un graphe dense où une majorité de nœuds est connectée entre elles, l'ensemble de toutes les cliques pourra avoir une cardinalité très grande. Tel qu'il a été défini, le graphe G^r ne présente pas cette caractéristique. En effet, pour que deux nœuds soit adjacents, il faut qu'ils se situent entre eux à une distance inférieure à celle définie par l'application et avoir des mesures initiales dont la différence est

inférieure à un seuil. Dans le cas où la distance définie par l'application est grande, les mesures interdiront la construction d'un nombre fulgurant d'arêtes. L'autre mécanisme qui réduira le nombre de cliques réalisables est la contrainte de la couverture. En effet, une clique n'est considérée que si elle contient au moins un nœud qui peut assurer la couverture de la zone. Pour cette raison, l'algorithme de la détermination d'un ensemble de cliques réalisables ne cherche les cliques qu'à partir des nœuds actifs i.e. les nœuds pouvant assurer la couverture de leurs zones. Nous devrons mentionner que l'algorithme ne détermine qu'un ensemble de cliques réalisables qui seront considérées dans la phase suivante. L'algorithme ne détermine pas les cliques réalisables.

```
1.  Initialisation
2.  Ncouv := L'ensemble des nœuds actifs
3.  U^r := Ø
4.  Pour tout nœud i dans Ncouv faire
5.    VTmp = Tous les nœuds du voisinage de i
6.    Tant que VTmp n'est pas vide faire
7.          E_k = {i, le premier nœud dans VTmp}
8.          Construire une clique de taille maximale à partir de E_k
9.          Si E_k ne fait pas partie de U^r alors
10.               U^r := U^r ∪ E_k
11.         Fin si
12.         VTmp = VTmp - E_k
13.   Fin tant que
14. Fin pour
```

Figure 4.7 Algorithme de détermination d'un ensemble de cliques réalisables

La Figure 4.7 illustre l'algorithme de détermination d'un ensemble de cliques réalisables. Pour construire les sous-ensembles complets E_k de taille maximale contenant le nœud i, nous analysons tout le voisinage de i (l'ensemble de nœuds ayant des arêtes avec le nœud i dans le graphe G'). Nous commençons par construire une clique E_1 dont la taille est maximale. Ensuite, nous construisons une deuxième clique E_2 de taille maximale pour les

nœuds qui ne sont pas couverts par la clique E_1 à partir d'un autre nœud adjacent au nœud i et qui n'appartient pas à E_1, et ainsi de suite jusqu'à ce que chaque nœud adjacent au nœud i appartienne à au moins une clique. De cette manière, nous assurons que l'algorithme retourne les cliques couvrant tous les nœuds voisins de i et qui ont les plus grandes cardinalités. Nous rappelons qu'une clique de taille maximale est une clique dont l'ajout d'un autre nœud donne un sous-graphe qui n'est pas complet. La Figure 4.8 illustre un exemple d'itération de l'algorithme. La première itération trouve la première clique E_1 avec 5 nœuds incluant le nœud i. La deuxième itération trouve la deuxième clique E_2 avec 3 nœuds.

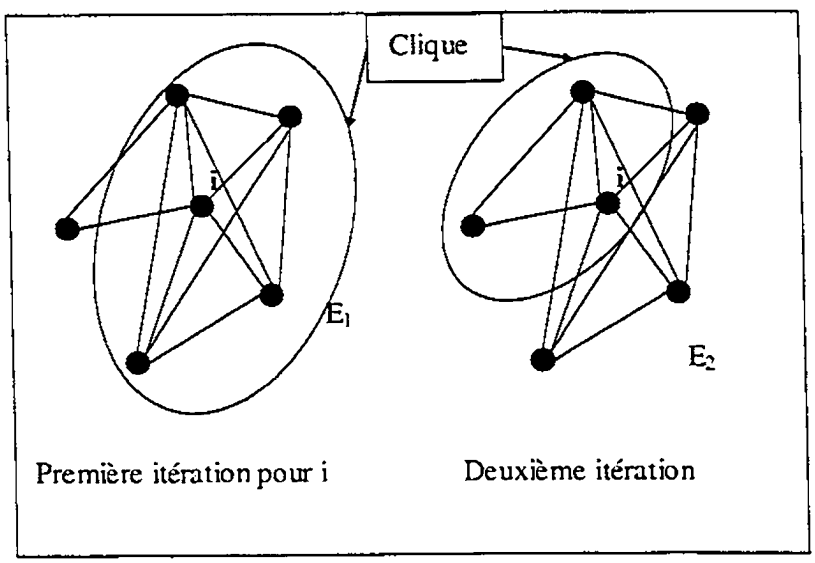

Figure 4.8 Exemple de la détermination des cliques réalisables

Dans le pire des cas, la complexité de l'algorithme de la détermination des grappes réalisables est évaluée à $O(m^3)$ où m est le nombre de nœuds du réseau. En effet, cette complexité peut être rencontrée si tous les nœuds du réseau sont actifs et si le graphe G' est fortement connecté i.e. chaque deux sommets du graphe sont connectés par une arête. C'est seulement dans ce cas que l'algorithme considérera tous les sommets du graphe G' dans la première boucle. La deuxième boucle se termine lors de la première itération vu que tous les nœuds du voisinage de i seront couverts par la première et la seule clique trouvée. Cependant, la construction d'une clique de taille maximale se fera en $((m-2)(m-1)/2)$. Cette valeur peut être facilement calculée : initialement E_k contient le nœud i et un nœud de

voisinage de i, chaque nœud restant du réseau vérifie la connectivité avec les éléments de E_k parce qu'il fait partie du voisinage de i; La taille de E_k augmentera de 1 à chaque vérification. Donc, la vérification va nécessiter un nombre d'itérations égale à 1 + 2 + ...+ m-2. D'où le résultat. Par conséquent, le nombre total d'itérations pour ce pire cas est ($m(m-2)(m-1)/2$).

Phase post optimisation : La détermination de partitionnement

La phase post optimisation est la troisième phase de cette approche de résolution du problème de formation de grappes. L'objectif de cette phase est de construire un partitionnement de l'hypergraphe à partir de son recouvrement trouvé lors de la phase d'optimisation. L'ensemble des grappes trouvées par Cplex est noté *RCV*. La Figure 4.9 présente l'algorithme de la phase post optimisation. Il est subdivisé en trois parties :

1. Les lignes de 2 à 9 assurent qu'un nœud n'est couvert que par une seule grappe. Elles éliminent les nœuds qui sont déjà couverts, des grappes dont les coûts sont élevés ;
2. Les lignes de 12 à 18 éliminent de *RCV* chaque grappe dont la tête et la liste des noeuds sont déjà couvertes par les grappes qui ont des coûts moins élevés ;
3. Les lignes de 20 à 27 traitent les grappes qui ont une tête commune. Elles trouvent, parmi les nœuds actifs couverts par deux grappes qui ont la même tête, celui qui a la distance minimale et l'élisent comme tête de sa grappe.

L'algorithme ordonne les grappes selon leurs coûts croissants avant chaque étape. De cette manière, l'algorithme favorise la construction des grappes qui ont des coûts moins élevés. La complexité de cet algorithme est égale à $O(nlog(n) + mn^2)$, où n est le nombre des grappes constituant le recouvrement trouvé par Cplex et m est le nombre maximum des éléments appartenant à une de ces grappes et qui peut être égale au nombre de nœuds de réseau dans le cas où celui-ci est fortement connecté. Cette formule de complexité est due au fait que :

- l'ordre des éléments d'un vecteur est égal à $O(xlogx)$ où x est le nombre des éléments du vecteur. L'ordre s'effectue trois fois.
- Les trois étapes exécutent les deux premières boucles en $½n(n-1)$ et la dernière boucle se fait au maximum m fois.

```
1.  Ordonner les grappes de RCV selon leurs coûts croissants
2.  Pour i=0 jusqu'à la taille de RCV faire
3.          Pour j=i+1 jusqu'à la taille de RCV faire
4.                  Pour chaque nœud dans G_i faire
5.                          Si le nœud appartient à G_j alors
6.                                  Eliminer le nœud de G_j
7.                          Fin si
8.                  Fin pour
9.          Fin pour
10. Fin pour
11. Ordonner les grappes de RCV selon leurs coûts croissants
12. Pour i=0 jusqu'à la taille de RCV faire
13.         Pour j=0 jusqu'à i-1 faire
14.                 Si la tête et les nœuds de G_i sont couverts par l'une des G_j alors
15.                         Eliminer la grappe G_i de RCV
16.                 Fin si
17.         Fin pour
18. Fin pour
19. Ordonner les grappes de RCV selon leur coût croissant
20. Pour i=0 jusqu'à la taille de RCV faire
21.         Pour j=i+1 jusqu'à la taille de RCV faire
22.                 Si G_i et G_j ont la même tête alors
23.                         Si G_j est vide alors éliminer G_j de RCV
24.                         Sinon trouver parmi les nœuds actifs couverts par G_i et G_j
                            celui qui a la distance minimale et l'élire comme tête
                            de sa grappe
25.                 Fin si
26.         Fin pour
27. Fin pour
```

Figure 4.9 Algorithme de la phase de post optimisation

CHAPITRE V
ÉVALUATION DES PERFORMANCES ET RÉSULTATS

Après avoir conçu deux approches de formation des grappes, il est nécessaire d'évaluer leurs performances afin de quantifier la qualité de ces approches. Les algorithmes répartis ont été simulés avec des réseaux de capteurs avec différentes tailles et différentes topologies. Pour ce faire, l'environnement de simulation OMNET++ [OMNET] a été choisi. Les algorithmes centralisés ont été implémentés à l'aide du langage C++ et les résultats ont été obtenus sur un serveur roulant le système d'exploitation Linux et offrant les services de la librairie BGL (Boost Graph Library) et les services de la version 10.1.1 de CPLEX.

Dans ce chapitre, nous allons présenter tout d'abord les résultats de l'approche répartie et leurs analyses. Ensuite, nous détaillerons la manière avec laquelle nous avons implémenté les algorithmes centralisés et nous commenterons les différents résultats obtenus. Une comparaison des deux approches conclura le chapitre.

5.1. Résultats de la simulation de l'approche répartie

Afin d'analyser les performances de notre approche répartie, nous avons simulé le comportement des algorithmes dans un réseau de capteurs. Les résultats de notre approche sont comparés à ceux de l'agrégation légère TAG décrite dans la section 3.3.1. Nous avons choisi cet algorithme en raison de son importance dans les réseaux de capteurs. En effet, il constitue l'algorithme de la collecte de données dans le système d'exploitation TinyOS. Ce système d'exploitation est le système le plus utilisé par la communauté scientifique pour l'implémentation des applications dans les réseaux de capteurs.

Nous avons opté pour le simulateur OMNET++ qui offre un environnement convivial et facile pour ce type de réseaux. Le choix de ce simulateur est justifié par plusieurs raisons. D'une part, OMNET++ a été validé par plusieurs chercheurs dans le domaine des réseaux de capteurs. D'autre part, c'est un logiciel à source libre qui utilise le langage C++ pour implémenter le comportement des nœuds, et, par conséquent, il s'est avéré adéquat pour notre étude. Le premier réseau qui a été simulé comporte 9 nœuds avec le nœud collecteur. L'application envoie une requête permettant de calculer la moyenne des températures captées par les nœuds du réseau. La requête utilisée est celle décrite dans l'exemple de la

Figure 4.14 du chapitre 4. Nous nous sommes intéressés à la consommation d'énergie et nous l'avons prise comme indicateur de performance dans notre étude.

5.1.1. Validation de l'implémentation et analyse des résultats dans un réseau de petite taille

La consommation d'énergie est étudiée dans les deux phases des deux algorithmes : la phase de la distribution et la phase de la collecte. La Figure 5.1 montre la consommation d'énergie durant la phase de la distribution dans les différents nœuds du réseau. En analysant cette figure, nous constatons que, dans notre approche, les nœuds consomment plus d'énergie que dans la méthode TAG. Ceci est dû, d'une part, au fait que les messages échangés dans notre approche sont plus volumineux. D'autre part, la manière de former les grappes dans notre approche peut générer un plus grand nombre de messages.

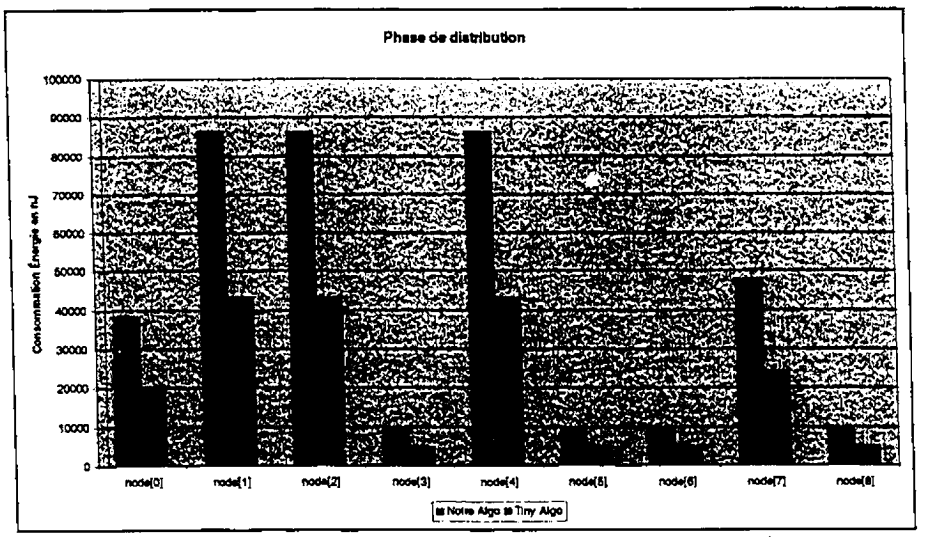

Figure 5.1 Consommation d'énergie : Phase de distribution

La Figure 5.2 montre la consommation d'énergie dans la phase de la collecte des données par les différents nœuds du réseau. L'analyse révèle que, dans notre approche, les nœuds consomment une quantité d'énergie très inférieure comparée à celle consommée dans le cas de l'agrégation TAG. Ceci est dû au fait que notre approche utilise un mécanisme de filtrage qui optimise le nombre de messages échangés par les nœuds. En effet, certains nœuds (nœuds 3, 5 et 6) ne participent pas à la phase de la collecte de données.

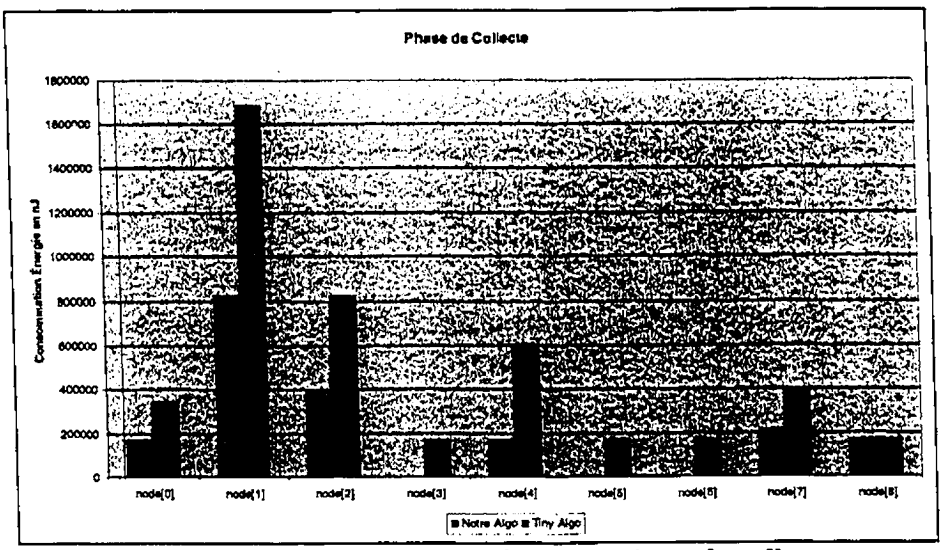

Figure 5.2 Consommation d'énergie : Phase de collecte

Enfin, en combinant les deux consommations d'énergie dans les deux phases, nous constatons que notre approche performe mieux que l'agrégation TAG. La Figure 5.3 illustre la consommation d'énergie dans les deux phases par les différents nœuds du réseau. Ceci s'explique par le fait que la phase de la distribution est une phase transitoire qui ne s'exécute qu'une seule fois lors du lancement de la requête, et que la phase de la collecte présente une très grande importance du point de vue de la consommation d'énergie. En effet, c'est la phase qui s'exécute d'une manière répétitive durant toute la durée de la requête.

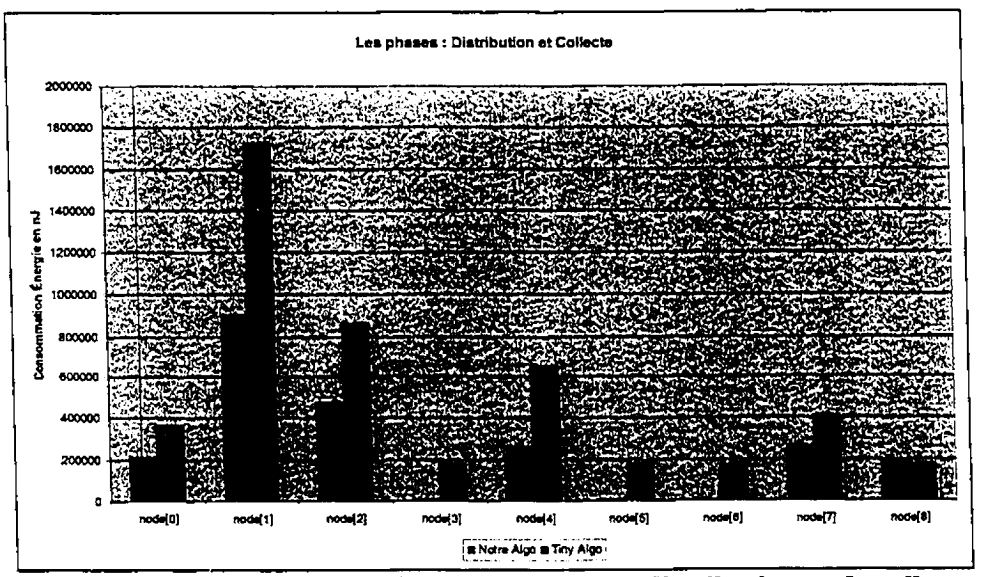

Figure 5.3 Consommation d'énergie : Phases de distribution et de collecte

5.1.2. Analyse des résultats de l'approche répartie

Les performances de l'algorithme sont analysées en utilisant des réseaux de capteurs avec différentes topologies et différents nombres de nœuds. Deux topologies sont analysées : La topologie *en ligne* et la topologie *en carré*. Dans la topologie en ligne, les nœuds sont déployés sur une ligne. Cette topologie peut présenter certains types d'applications où le déploiement des capteurs se fait sur une ligne. Par exemple, des capteurs déployés pour mesurer la qualité de l'eau le long d'un fleuve. La Figure 5.4 illustre un exemple de réseau de capteurs avec la topologie en ligne. La deuxième topologie considérée est dite en carré vu que les nœuds sont localisés sur les sommets des carrés dans un espace donné. La Figure 5.5 illustre un exemple de réseau de capteurs avec la topologie en carré. Ces topologies sont choisies d'une manière simple vu les limitations de l'environnement de simulation où les connexions entre les nœuds se définissent une à une dans un fichier texte.

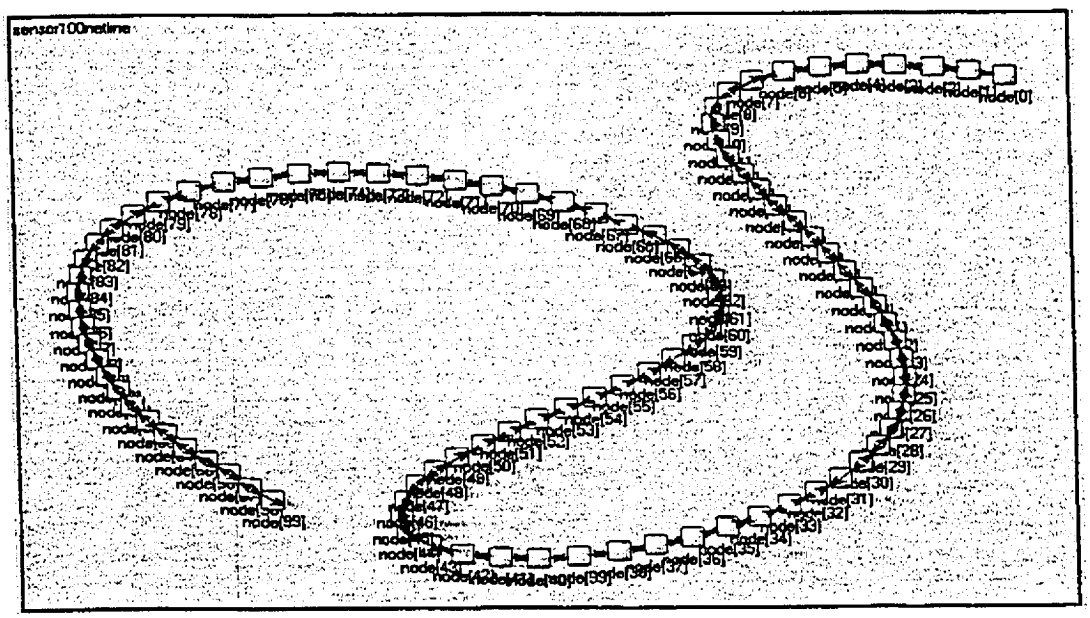

Figure 5.4 Exemple de topologie en ligne

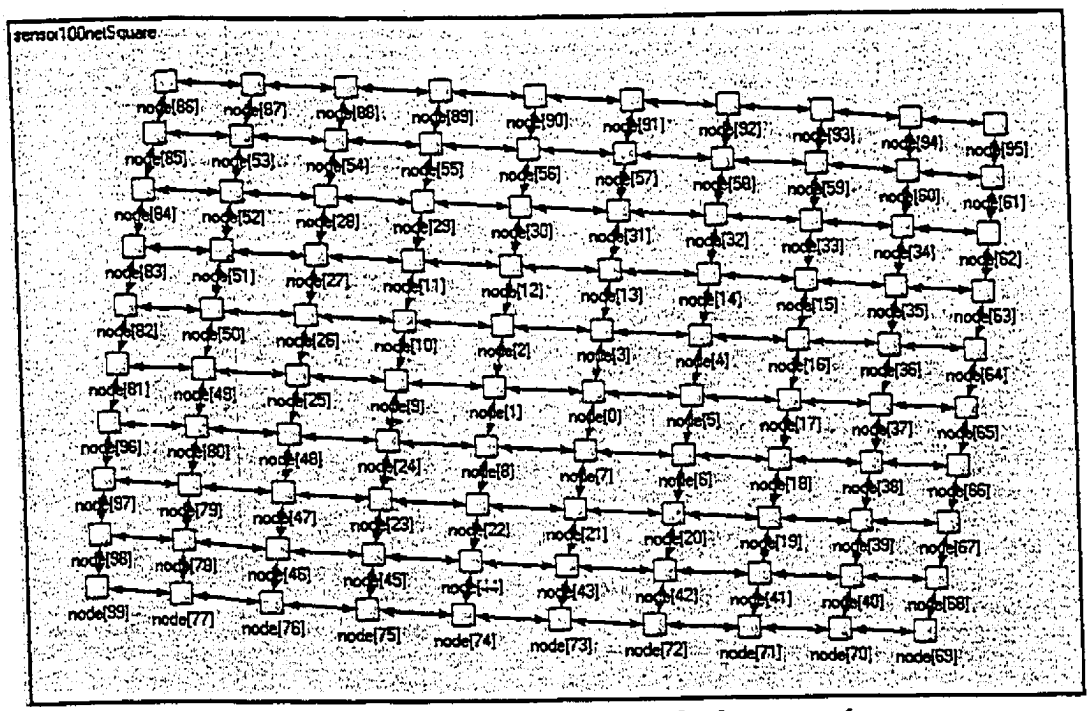

Figure 5.5 Exemple de topologie en carré

Nous avons considéré des réseaux de capteurs dont la taille varie entre 10 et 100 nœuds. Ce nombre restreint de capteurs est dû à la même raison évoquée dans le paragraphe précédent i.e. l'environnement de simulation ne fournit pas une manière facile de construire les nœuds et les liaisons entre les nœuds.

La première remarque que nous avons faite est que les conclusions sont identiques pour les deux topologies considérées. Par conséquent, nous n'allons détailler dans les prochains paragraphes que les résultats obtenus dans le cas de la topologie en carré. La Figure 5.6 illustre l'énergie totale consommée en faisant varier le nombre de nœuds du réseau. Pour les deux algorithmes, l'énergie totale consommée devient de plus en plus grande en augmentant le nombre de nœuds. Ce résultat est trivial. Mais, le gain en consommation d'énergie est plus élevé dans notre algorithme comparé à celui de l'algorithme TAG. En effet, en augmentant la taille de réseau en la multipliant par quatre (i.e. d'un réseau avec une taille de 25 nœuds à une taille de 100 nœuds), la consommation de l'énergie dans notre algorithme est multipliée par 7 alors que celle de l'algorithme TAG est multipliée par plus de 8 fois. Par conséquent, notre approche répartie se comporte avec

l'extensibilité du réseau d'une manière plus efficace que l'approche TAG. Ceci est dû au fait que notre algorithme permet de filtrer les messages plus efficacement.

Il faut noter que dans notre algorithme la consommation de l'énergie dans un réseau avec 100 nœuds est moins élevée que celle dans un réseau avec 80 nœuds. Ceci s'explique par le fait que la formation des grappes se fait d'une manière non optimale et que l'approche répartie peut se comporter d'une manière imprévisible, vu que les nœuds ont une visibilité restreinte sur leur environnement.

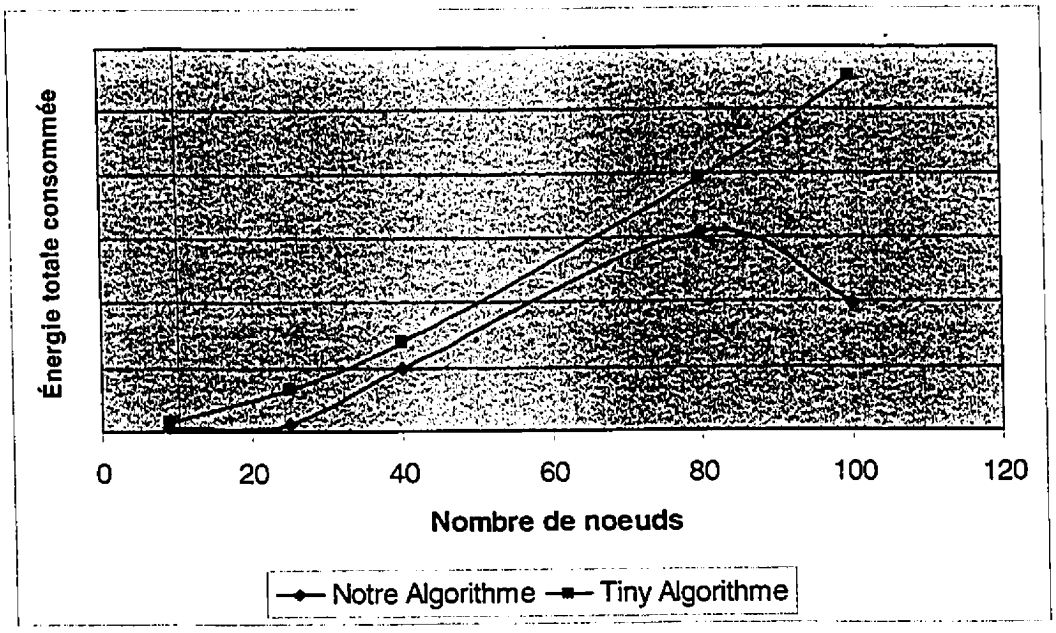

Figure 5.6 Impact du nombre de nœuds sur la consommation d'énergie

La Figure 5.7 présente l'impact de la valeur du paramètre *pas* sur la consommation d'énergie en variant la taille du réseau. Nous n'avons évalué que les performances de notre algorithme vu que TAG n'utilise pas ce paramètre. Les résultats révèlent que l'impact de ce paramètre n'est ressenti que dans le cas des réseaux de grande taille. Dans les autre cas, l'impact est invisible ou peu visible. En effet, il n'y pas d'impact pour le réseau contenant 9 nœuds, et l'impact est minime pour le réseau qui contient 25 nœuds. Par contre, le paramètre *pas* a permis de diminuer la consommation d'énergie de sept fois en augmentant sa valeur de quatre fois pour un réseau contenant 100 nœuds. Ceci est dû à la signification de ce paramètre qui permet à une application de spécifier le degré de précision des mesures le long de la superficie considérée. Dans le cas du réseau étudié, les nœuds sont répartis sur la

superficie d'une manière uniforme (i.e. les distances entre les nœuds sont identiques). Donc, dans le cas du réseau avec un grand nombre de nœuds, l'augmentation de la valeur du paramètre *pas* implique la formation des grappes contenant un grand nombre de nœuds. Par conséquent, le mécanisme de filtrage permet de réduire efficacement le nombre de messages et ainsi la consommation d'énergie.

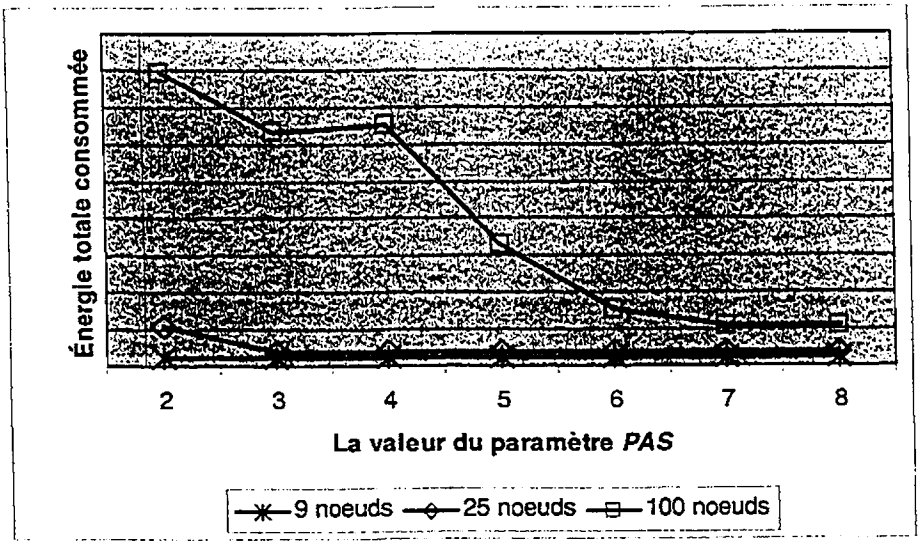

Figure 5.7 Impact du paramètre '*pas*' sur la consommation d'énergie

D'une manière générale, cette étude nous a permis de conclure d'une part, que notre approche est plus efficace que l'algorithme TAG en terme de la consommation d'énergie et, d'autre part, que le paramètre '*pas*' n'a d'effet que pour les réseaux de grandes tailles.

5.2. Résultats de l'implémentation de l'algorithme centralisé

Nous avons proposé dans le chapitre précédent une approche centralisée de la formation de grappe. Nous avons formulé ce problème en un problème linéaire en nombre entier et nous avons présenté deux manières de résoudre ce problème : une première basée sur la recherche taboue et une deuxième basée sur une méthode exacte (simplexe du CPLEX). Dans cette section, nous allons présenter l'implémentation des algorithmes centralisés et nous analyserons les résultats obtenus par la recherche taboue tout en évaluant la qualité de ces résultats avec notre approche basée sur la résolution avec CPLEX.

5.2.1. Implémentation des algorithmes de la recherche taboue

Les algorithmes de la recherche taboue ont été implémentés avec le langage de programmation C++ dans l'environnement (Visual C++ 2005 Express Edition). Nos algorithmes utilisent la notion de graphe. De ce fait, et vu la sensibilité de la manipulation des structures présentant les graphes, nous avons opté pour l'utilisation d'une librairie reconnue par la communauté scientifique. En effet, les données des graphes devront être stockées d'une manière optimale pour pouvoir réduire l'espace mémoire utilisé et minimiser le temps d'accès à ces données. Nous avons utilisé la librairie BGL (Boost Graph Library) [JER02] dont le code est fourni librement pour construire et manipuler les graphes. BGL fournit une interface générique qui permet l'accès aux structures des graphes tout en cachant la complexité de l'implémentation. C'est une interface ouverte permettant l'interopérabilité avec les autres librairies implémentant cette interface. BGL fournit aussi un ensemble de classes des graphes qui implémentent cette interface.

BGL fournit un ensemble d'algorithmes qui peuvent se subdiviser en deux catégories : des algorithmes de noyau et des algorithmes de graphes. Les algorithmes de noyau sont génériques et contiennent la recherche en largeur, la recherche en profondeur et la recherche à coût uniforme. Le deuxième type d'algorithmes contient une large gamme d'algorithmes de graphes connus dans la littérature. Par exemple, on y trouve l'algorithme du plus court chemin de Dijkstra, de Bellman-Ford et l'algorithme d'arbre de recouvrement à coût minimum de Kruskal.

BGL permet de présenter les graphes à l'aide de plusieurs types de structures. La librairie fournit essentiellement deux classes de graphes : *adjacency_list* et *adjacency_matrix*. La première classe présente un graphe à l'aide de sa liste d'adjacence. Elle est paramétrable et ainsi son utilisation est optimisée dans différentes situations. Par exemple, le graphe peut être orienté ou non, il peut contenir des arêtes parallèles ou non. Dans ce type de structure, l'ajout et la suppression d'un sommet sont optimisés par rapport à l'usage de l'espace mémoire. Le deuxième type de classes permet de présenter le graphe avec sa matrice d'adjacence. Elle stocke les arêtes à l'aide d'une matrice de taille $|V|x|V|$ ($|V|$ est le nombre de sommets). Les cellules de cette matrice représentent les arêtes. Elle est appropriée dans le cas des graphes denses dont le nombre d'arêtes avoisine $|V|^2$.

Dans notre cas, nous avons utilisé la première classe (*adjacency_list*) qui s'avère la plus adéquate vu que le nombre d'arêtes n'est pas très élevé. Notre graphe n'est pas orienté et il n'admet pas d'arêtes parallèles. Les données des nœuds du réseau permettant de construire le graphe sont stockées dans un fichier texte contenant les coordonnées de chaque nœud, sa mesure initiale et un booléen représentant le fait que le nœud est actif ou non.

5.2.2. Implémentation des algorithmes pour CPLEX

La deuxième méthode centralisée proposée pour la résolution du partitionnement des nœuds en grappes s'exécute en trois parties et est basée sur une méthode exacte de résolution des problèmes linéaires. La première partie consistant à déterminer les grappes réalisables est implémentée dans le même environnement où le graphe G' est construit pour la recherche taboue. En effet, c'est le même graphe G' qui sert pour les deux méthodes de résolution. La deuxième partie consistant à déclarer les variables du problème linéaire, les contraintes et la fonction objective utilise les APIs fournis par ILOG CPLEX 10.1 [ILOG]. Celui-ci est un optimiseur commercial permettant de résoudre une grande variété de problèmes linéaires ou quadratiques avec un grand nombre de variables dans des temps d'exécution raisonnables.

La phase post optimisation utilise les résultats de l'optimiseur afin de trouver un partitionnement de l'hypergraphe. Afin d'exécuter les trois parties de cette méthode de résolution, un serveur roulant CPLEX est utilisé. Le système d'exploitation installé sur le serveur est Linux (Red Hat 3.4.4-2). Le serveur est doté d'une capacité mémoire de 1Go et d'un processeur Intel(R) Pentium(R) 4 1.80GHz.

5.2.3. Résultats et analyses

Les implémentations des deux méthodes de résolution du problème linéaire ont été validées en considérant des réseaux avec de petites tailles afin de déceler les erreurs et vérifier manuellement les résultats obtenus. Ensuite, nous avons étudié les coûts des solutions obtenues par la recherche taboue. Puis, nous avons évalué leur qualité en la comparant avec celle des solutions obtenues par la deuxième méthode basée sur CPLEX. Enfin, les deux approches répartie et centralisée ont été comparées à l'aide des coûts des grappes formées par les deux approches.

Validation des implémentations avec un réseau simple : Recherche taboue et CPLEX

Afin de valider les implémentations de nos algorithmes, nous avons considéré un réseau de capteurs avec un nombre réduit de nœuds. Le réseau contient huit capteurs dont les données initiales sont présentées dans le Tableau 5.1. Celui-ci contient les coordonnées des nœuds, leurs mesures initiales et un indicateur présentant le fait que le nœud peut assurer la couverture ou non de sa zone (i.e. nœud actif ou ordinaire). Le nœud collecteur se situe au centre de repère des coordonnées. Nous supposons que l'application spécifie la valeur du paramètre *pas* égale à 10m et la valeur du paramètre *sem* (seuil des erreurs de mesures) égale à 10 unités de mesure. Les données initiales des nœuds et les valeurs des paramètres *pas* et *sem* sont choisies de manière à créer un bon exemple de validation de nos implémentations. L'algorithme de la construction du graphe G' donne le résultat illustré à la Figure 5.8.

Tableau 5.1 Données initiales des nœuds pour le réseau de validation

Nœud#	Coordonnées	Mesure	Statut	Nœud#	Coordonnées	Mesure	Statut
0	(0,0)	0	Actif	5	(-1,-1)	2	Ordin.
1	(1,-1)	4	Ordin.	6	(2,-4)	24	Actif
2	(2,-2)	13	Actif	7	(2.5,-3)	25	Actif
3	(1,-2)	6	Actif	8	(3,-4)	26	Actif
4	(1.5,-3)	15	Ordin.				

Pour calculer le coût des grappes, nous avons affecté aux coefficients δ et β la valeur 1. Nous rappelons que le coût d'une grappe j est calculé à l'aide de l'équation (4.7) ($c_j = \delta\, d_{j,s}^{\alpha} - \beta \sum_{i=1}^{n_j} d_{i,s}^{\alpha}$) présentée au chapitre précédent. La valeur du coefficient α (le degré d'atténuation de la propagation de signal) est égal à 2. La valeur initiale du coefficient σ est 1. Nous rappelons que ce coefficient est associé à la pénalité de la violation de la contrainte de clique par une solution. La valeur de la pénalité d'une solution est calculée à l'aide de l'équation 4.10 expliquée au chapitre IV.

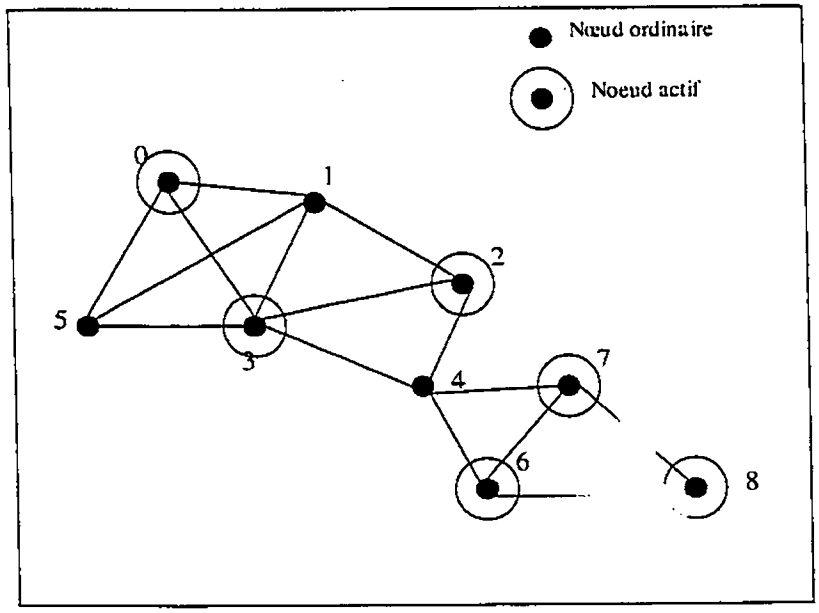

Figure 5.8 Graphe G^r et solution initiale du réseau de validation

Dans une première phase, nous allons présenter les résultats trouvés par les algorithmes implémentés dans le cadre de la résolution du problème à l'aide de la recherche taboue. La solution initiale trouvée par l'algorithme présenté au chapitre IV donne trois grappes {0,1,3,5},{2,4} et {6,7,8} avec les nœuds 0, 2 et 6 respectivement tête de ces grappes. La Figure 5.9 illustre les grappes de la solution initiale. Le coût de cette solution initiale est égal à -32.5. Sa pénalité est nulle vu que toutes ses grappes sont des cliques. Par conséquent, la solution initiale respecte toutes les contraintes de notre problème.

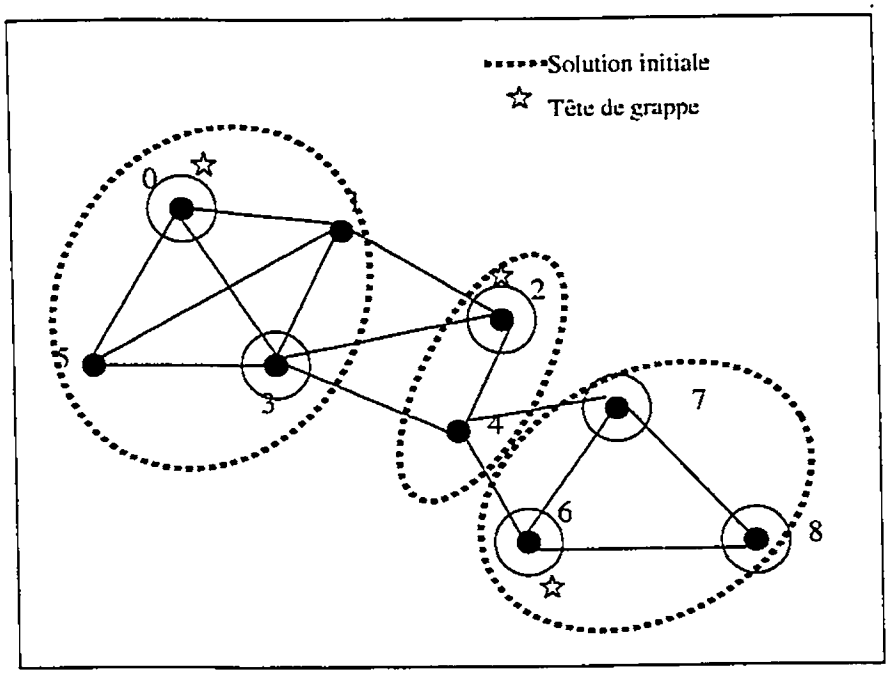

Figure 5.9 Solution initiale du réseau de validation

Nous avons défini trois conditions d'arrêt des itérations :

1. Tous les mouvements sont interdits ;
2. Un nombre maximum d'itérations est atteint ;
3. Un nombre maximum d'itérations où la meilleure solution n'est pas améliorée est atteint.

La première condition permet d'arrêter les itérations quand l'intersection du voisinage de la solution traitée et les éléments des deux listes taboues est vide, c'est-à-dire que tous les mouvements sont interdits. La deuxième condition permet de fixer le nombre d'itérations. Cette condition est appropriée dans notre cas vu que la phase de formation des grappes devrait être limitée dans le temps et que le nœud collecteur devra contrôler ce temps. Nous rappelons que la formation des grappes est une première phase qui s'exécute avant la collecte des données. Par conséquent, en contrôlant le nombre d'itérations, le nœud collecteur contrôle ainsi le temps d'exécution de l'algorithme. La troisième condition permet d'arrêter les itérations dans le cas où l'algorithme trouve une bonne solution dans les premières itérations et il est en train de s'éloigner de la solution optimale. Elle permet donc de gagner du temps d'exécution surtout que notre algorithme de détermination de la solution initiale est conçu de telle manière à donner de bonnes solutions.

Comme nous l'avons expliqué au chapitre précédent, la taille de la liste taboue devra être dynamique et oscille entre les deux valeurs extrêmes : la valeur minimale générant des cycles et la valeur maximale empêchant d'obtenir la meilleure solution. Pour cet exemple simple, nous avons fixé la taille des deux listes taboues au nombre de nœuds du réseau qui est de 9. Nous avons remarqué que ce nombre génère un cycle.

La valeur du coefficient de la pénalité σ est initialisée à 1. Après chaque violation de la contrainte de clique par les grappes d'une solution choisie à une itération, nous doublons la valeur de σ pour s'éloigner des solutions non réalisables. Afin d'assurer la diversification de la recherche taboue, nous faisons osciller cette valeur en diminuant sa valeur dans le cas où la solution choisie à la fin d'une itération ne viole pas cette contrainte (i.e. pénalité de la solution égale à zéro).

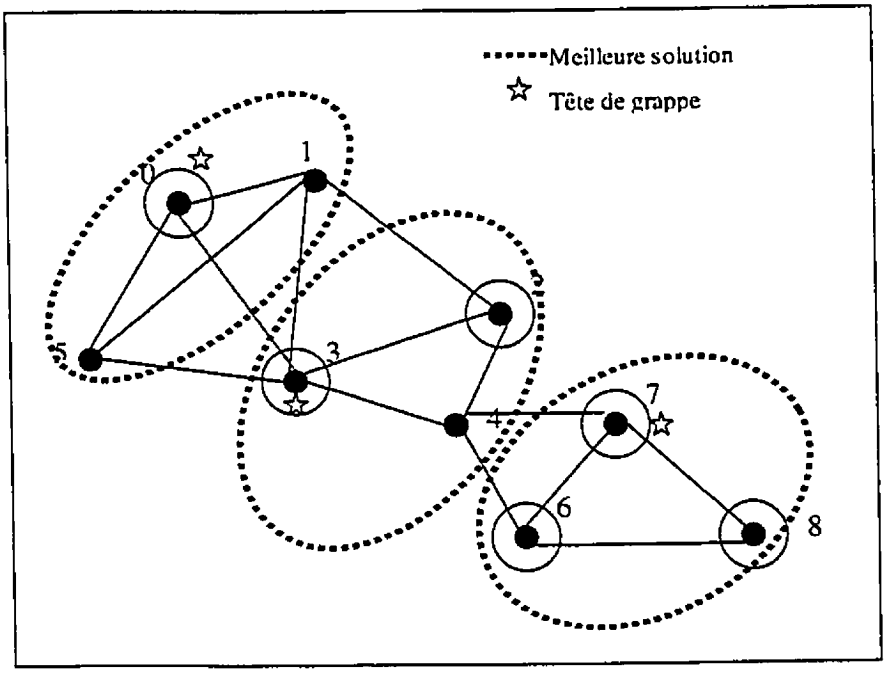

Figure 5.10 La meilleure solution du réseau de validation

En fixant la valeur du nombre d'itérations maximum à 20 et la valeur du nombre d'itérations maximum qui n'améliore pas la meilleure solution, nous obtiendrons une meilleure solution illustrée à la Figure 5.10 dont le coût est -48. La solution est formée de trois grappes {0,1,5},{3,4,2} et {7,8,6} dont les têtes sont 0, 3 et 7 respectivement. La solution est obtenue après 7 itérations.

Nous remarquons que la solution initiale a été améliorée d'une manière significative, un gain de presque 50% du coût, et que les têtes de grappes sont déterminées de telle manière à minimiser le coût des grappes. Les têtes sont les nœuds de la grappe les plus proches du nœud collecteur. Ceci implique une minimisation de l'énergie de transmission des mesures collectives. Lorsque nous avons analysé les solutions intermédiaires, nous avons remarqué que les itérations ont permis d'analyser différents nombres de grappes et différentes élections de têtes de grappes.

Afin de valider l'implémentation des algorithmes de la deuxième méthode de résolution basée sur CPLEX, nous avons fait des tests à l'aide du même réseau introduit au début de cette section. L'algorithme de la détermination des grappes réalisables trouve 11 cliques. La phase d'optimisation avec CPLEX retourne le recouvrement formé des grappes suivantes {0,1,3,5}, {3,2,4} et {7,6,8} dont les têtes sont les nœuds 0,3 et 7 respectivement. Ce recouvrement ne constitue pas un partitionnement. Par conséquent, la phase post optimisation donne la solution finale formée des grappes {0,1,5}, {3,2,4} et {7,6,8} dont les têtes sont les nœuds 0, 3 et 7 respectivement. Cette dernière phase n'a fait qu'éliminer le nœud 3 de la première grappe. Pour ce réseau de validation, les solutions obtenues par CPLEX et par la recherche taboue sont identiques.

Consommation d'énergie totale pour la méthode basée sur la recherche taboue

Dans une première analyse, nous avons étudié l'impact des deux paramètres les plus importants sur la qualité de la solution trouvée par les itérations de la recherche taboue. Les deux paramètres considérés sont : la taille de la liste taboue et le nombre maximum d'itérations. Pour mener cette étude, nous avons considéré un réseau de capteurs qui a une topologie en carré et dont tous les nœuds sont actifs.

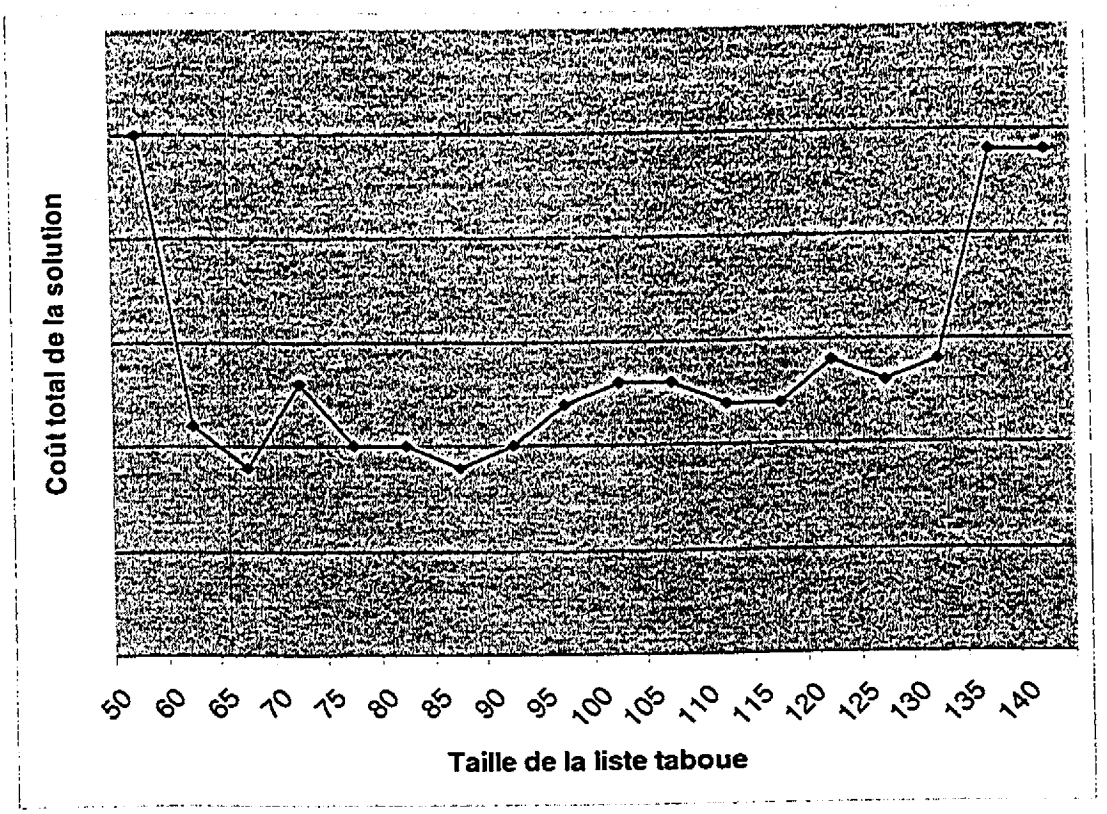

Figure 5.11 Impact de la taille de la liste taboue

Impact de la taille de la liste taboue sur la qualité de la solution trouvée

Comme il a été déjà mentionné, la taille de la liste taboue influence la qualité de la solution trouvée par la recherche taboue. Des résultats de recherche [GLO93] ont montré que le fait de changer cette taille d'une manière dynamique est plus efficace qu'une valeur fixe le long des itérations. De ce fait, nous avons étudié l'impact de la valeur de cette taille sur les solutions trouvées pour un réseau constitué de 100 nœuds répartis sur les sommets des carrés d'une zone. Pour faciliter l'analyse, nous avons considéré que tous les nœuds sont actifs (i.e. peuvent assurer la couverture de leur zone). Nous avons aussi fixé le nombre d'itérations à 1000. Nous avons varié la taille de la liste taboue tout en analysant le coût total de la solution trouvée. La Figure 5.11 illustre le résultat des exécutions.

Ces résultats sont concordants avec l'étude d'impact de la taille de la liste taboue que nous avons présentée. En effet, les meilleurs résultats sont obtenus pour des valeurs voisines du nombre de nœuds du réseau. Des valeurs élevées dégradent la qualité de la solution vu

qu'elles limitent le nombre de voisinage des solutions analysées en interdisant un grand nombre de mouvements. Des petites valeurs génèrent des cycles et dégradent ainsi la qualité de la solution. De ce fait, la valeur de la taille de la liste taboue implémentée est dynamique, elle oscille dans l'intervalle [0.75N,1.1N] (N est le nombre de nœuds du réseau). Après un certain nombre d'itérations, nous augmentons la taille de la liste taboue.

Impact du nombre d'itérations sur la qualité de la solution trouvée

Dans cette étude, nous voulons quantifier l'impact du nombre d'itérations sur la qualité de la solution trouvée et connaître les limites des améliorations que notre algorithme permet d'atteindre. Nous considérons un réseau de capteurs de 1000 nœuds et nous faisons varier le nombre d'itérations maximal pour étudier l'impact de ce facteur sur la qualité de la solution. Les valeurs des autres paramètres ne changent pas, à l'exception du paramètre 'nombre maximum d'itérations qui n'améliorent pas la meilleure solution' qui prend une valeur permettant à l'algorithme d'atteindre le nombre maximum d'itérations. La Figure 5.12 illustre l'impact du nombre d'itérations sur la qualité de la solution trouvée.

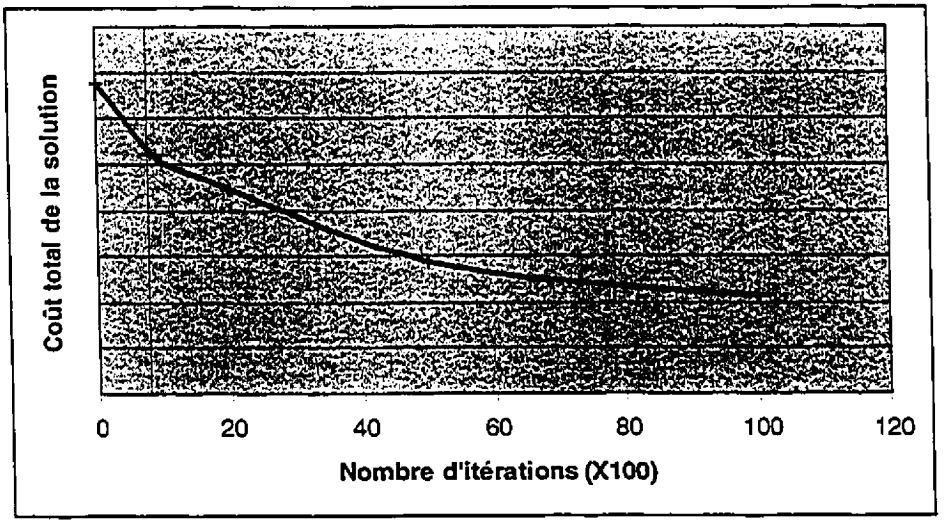

Figure 5.12 Impact du nombre d'itérations sur la qualité de la solution

Les résultats montrent qu'en augmentant le nombre d'itérations, le coût de la solution trouvée diminue et ainsi nous avons une meilleure qualité de la solution. Ceci est vrai jusqu'à une certaine valeur où nous obtiendrons des solutions avec presque le même coût, même si le nombre d'itérations augmente. En effet, de 0 à 5000 itérations, nous avons

amélioré la solution de 2,33 fois (i.e. le coût est diminué de 2,33 fois). Par contre, de 5000 à 10000 itérations, l'amélioration de la solution n'était que de 1,35 fois. Cette stagnation est due au fait que l'algorithme ne peut améliorer la solution après une certaine limite liée au choix des autres paramètres, comme par exemple la valeur du coefficient de la pénalité, la valeur des coefficients dans la formule du coût des grappes, la valeur de la taille de la liste taboue. Il faut noter aussi qu'une autre raison justifiant ces résultats est que l'algorithme qui détermine la solution initiale donne une bonne solution et ainsi les itérations de la recherche taboue n'ont pas un grand intervalle d'amélioration de la solution initiale.

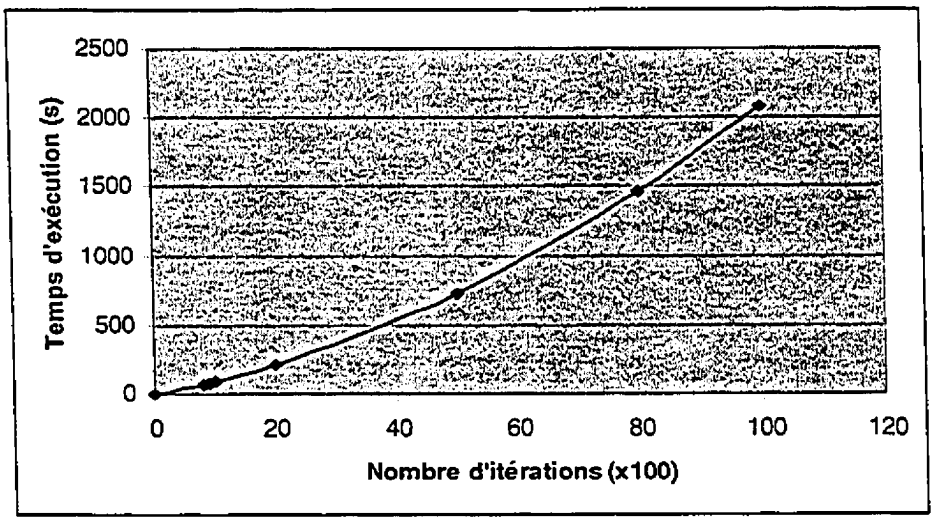

Figure 5.13 Temps de réponse et nombre d'itérations

Afin de pousser les analyses de ces derniers résultats, nous avons calculé le temps de réponse et le nombre total des mouvements traités pour chaque nombre maximum d'itérations. Nous rappelons que, dans une itération, l'algorithme analyse un voisinage d'une solution formée par les mouvements appliqués à celle-ci. La Figure 5.13 représente le temps d'exécution. Les résultats montrent que le temps d'exécution augmente d'une manière significative en augmentant le nombre d'itérations. En effet, de 800 à 5000 itérations, le temps de réponse augmente de 10 fois, et c'est dans cet intervalle d'itérations que l'algorithme améliore la qualité de la solution d'une manière efficace, i.e. le coût est diminué de 40%. Il faut noter aussi que le nombre de solutions analysées augmente de 6,3 fois, i.e. il passe de 16 millions de solutions à 101 millions.

Par contre, de 8000 à 10000 itérations, le temps d'exécution augmente de 1,4 fois mais le coût de la solution n'est amélioré que de 9%. Dans cet intervalle d'itérations, le nombre de solutions analysées augmente de 1,2 fois, i.e. il passe de 159 millions de solutions à 196 millions. Le nombre de solutions analysées augmente d'une manière linéaire par rapport au nombre d'itérations.

La conclusion que nous avons retenue de cette analyse est que la qualité de la solution trouvée s'améliore en augmentant le nombre d'itérations. L'amélioration se fait jusqu'à une valeur limite après laquelle le temps d'exécution augmente et le nombre de solutions analysées augmente sans améliorer efficacement la qualité de la solution trouvée. Cette valeur limite est évaluée à 8 fois le nombre de nœuds du réseau. C'est cette valeur que nous allons utiliser par la suite.

Qualité des solutions de la recherche taboue : Comparaison avec CPLEX

L'objectif principal de la conception de la deuxième méthode de résolution du problème de la formation des grappes basée sur CPLEX est d'étudier la qualité des solutions trouvées par la recherche taboue. En effet, l'heuristique ne donne pas la solution optimale mais elle fournit la meilleure solution trouvée. De ce fait, afin d'analyser la qualité de ces solutions, il faut comparer leurs coûts avec ceux trouvés par une autre approche utilisant des méthodes exactes. Pour mener cette étude, nous avons considéré des réseaux qui ont une topologie en carré et dont les nœuds sont tous actifs (i.e. peuvent assurer la couverture). Nous avons fixé les valeurs des paramètres fournies par l'application (i.e. les paramètres de la qualité de service : *pas* et *sem*).

À chaque expérience, nous varions le nombre de nœuds N du réseau et nous affectons la valeur ($\delta x N$) au nombre d'itérations maximal de la recherche taboue (résultat des précédentes expériences). Nous calculons les coûts de la solution trouvée par la recherche taboue et ceux de la solution trouvée par CPLEX, les temps d'exécution des deux approches et le nombre des grappes analysées par les deux méthodes. La Figure 5.14 illustre l'évolution des coûts des solutions en fonction du nombre de nœuds pour les deux approches.

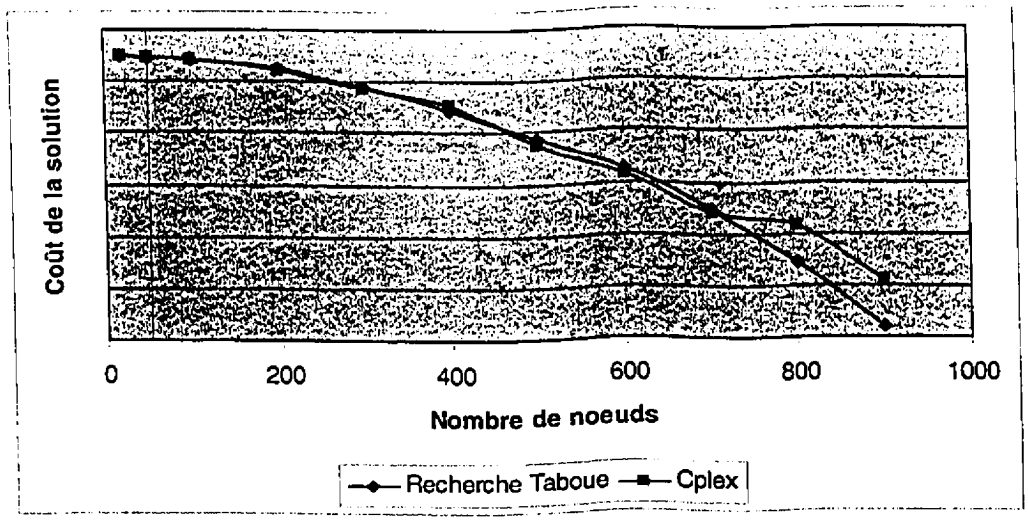

Figure 5.14 Comparaison des coûts des solutions recherche taboue et CPLEX : Topologie en carré

Ce résultat révèle que les solutions trouvées par la recherche taboue et CPLEX ont presque les mêmes coûts pour des réseaux dont le nombre est inférieur à 700 nœuds. À partir de cette taille de réseau, les solutions trouvées par la recherche taboue sont meilleures que celle trouvée par l'approche basée sur CPLEX. Ceci est dû, d'une part, à la phase post optimisation de cette approche où le calcul d'un partitionnement du graphe à partir du recouvrement trouvé par CPLEX peut donner des solutions qui ne sont pas optimales, et d'autre part, la détermination des cliques réalisables limite l'ensemble des grappes considérées dans la phase d'optimisation.

Nous avons analysé les temps d'exécution des deux approches afin d'évaluer la qualité de la solution trouvée par la recherche taboue en terme de temps d'exécution. La Figure 5.15 illustre les temps d'exécution en fonction du nombre de nœuds pour les deux approches. Ces courbes révèlent que les temps d'exécution des approches sont presque identiques pour les réseaux qui ont un nombre de nœuds entre 20 et 700 nœuds. Cependant, à partir de cette taille de réseau, le temps de l'approche basée sur CPLEX augmente d'une manière fulgurante. Sa valeur est multipliée par 46 pour une augmentation de la taille du réseau de l'ordre de 1,28 fois. Ceci est expliqué par le fait que l'algorithme simplexe est exponentiel. Par contre, le temps d'exécution de la recherche taboue augmente d'une manière

raisonnable. Sa valeur est multipliée par 2 pour une augmentation de la taille du réseau de l'ordre de 1,28 fois.

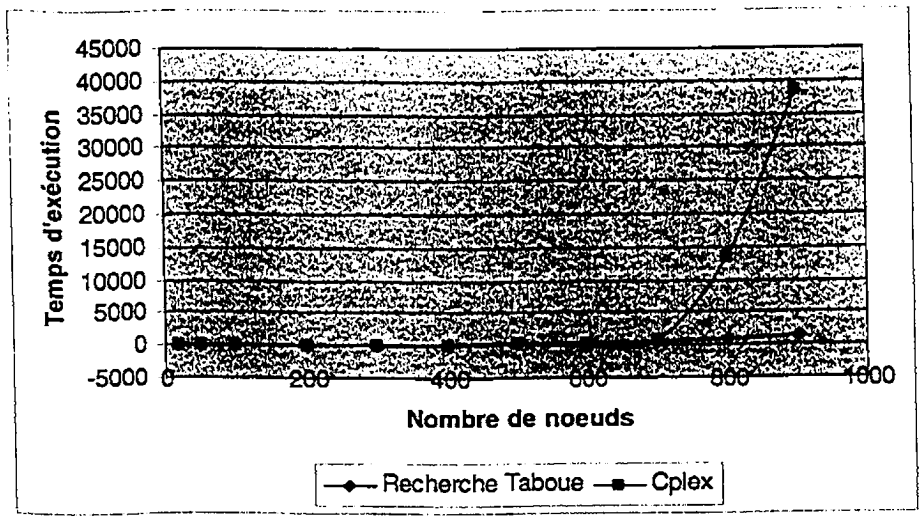

Figure 5.15 Comparaison des temps d'exécution de la recherche taboue et CPLEX : Topologie en carré

Ceci confirme les résultats trouvés précédemment lors de l'analyse de ce facteur dans les sections précédentes. En ce qui concerne l'ensemble des grappes analysées dans les deux approches, le nombre est compté par milliers pour l'approche basée sur CPLEX. Il varie entre 1 et 37 mille grappes pour l'intervalle du nombre de nœuds variant entre 20 et 900 nœuds. Le même nombre est compté en millions pour la recherche taboue. Il varie entre 0,01 et 128 millions de grappes analysées. Même si l'ensemble des grappes analysées dans la recherche taboue est nettement supérieur à celui considéré par l'approche basée sur CPLEX, le temps d'exécution de la première approche reste inférieur à celui de la deuxième approche.

Les résultats de cette étude nous ont permis de conclure que, d'une part, les solutions trouvées par la recherche taboue sont d'une très bonne qualité en terme de coût des grappes et en terme de temps d'exécution, et d'autre part, l'approche basée sur la recherche taboue se comporte bien avec l'extensibilité du réseau. En effet, en augmentant le nombre de nœuds du réseau, le temps d'exécution augmente d'une manière raisonnable (i.e. la courbe est presque linéaire).

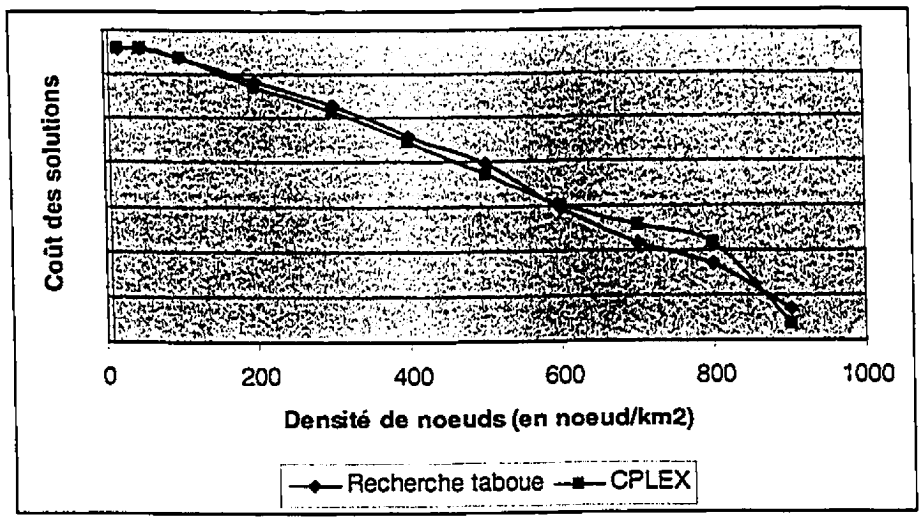

Figure 5.16 Comparaison des coûts des solutions recherche taboue et CPLEX : Topologie aléatoire

Comparaison de la recherche taboue et la méthode basée sur CPLEX dans des réseaux avec une topologie aléatoire

L'analyse précédente est menée en considérant des réseaux dont la topologie est en carré. Afin de valider ces résultats dans d'autres topologies, nous avons mené les mêmes expériences en prenant en considération des réseaux avec des topologies aléatoires où les nœuds sont mis aléatoirement dans une superficie de 1 km^2. Nous avons gardé les mêmes paramètres utilisés dans la topologie en carré, i.e. la valeur de *pas* égale à 3, la valeur de *sem* égale à 10, le nombre maximum d'itérations égale à *8*N*, avec *N* désigne le nombre de nœuds du réseau. Le nœud collecteur est mis au centre avec des coordonnées égalent à (0,0). La distribution des nœuds est uniforme. À chaque construction de graphe, nous générons des nombres réels aléatoires à l'aide de la librairie Boost [BOOST]. Nous avons fait varier la densité des nœuds de 20 à 900 nœuds par km^2.

La Figure 5.16 illustre les coûts des solutions par rapport à la densité des nœuds pour les deux méthodes de résolution. L'allure des courbes ne diffère pas de celle obtenue avec la topologie en carré. Les coûts des solutions trouvées par la recherche taboue et ceux obtenus par la méthode basée sur CPLEX sont proches. En analysant les courbes illustrées à la Figure 5.17 et qui représentent les temps d'exécution par rapport à la densité des nœuds, nous constatons que les résultats obtenus avec la topologie aléatoire sont identiques à ceux obtenus avec la topologie en carré. Les temps d'exécution de la méthode basée sur la

recherche taboue augmentent d'une manière raisonnable avec le nombre de nœuds, alors que dans la méthode basée sur CPLEX, les temps d'exécution augmentent d'une manière fulgurante. Les mêmes raisons évoquées dans le cas des réseaux avec la topologie en carré justifient ces derniers résultats.

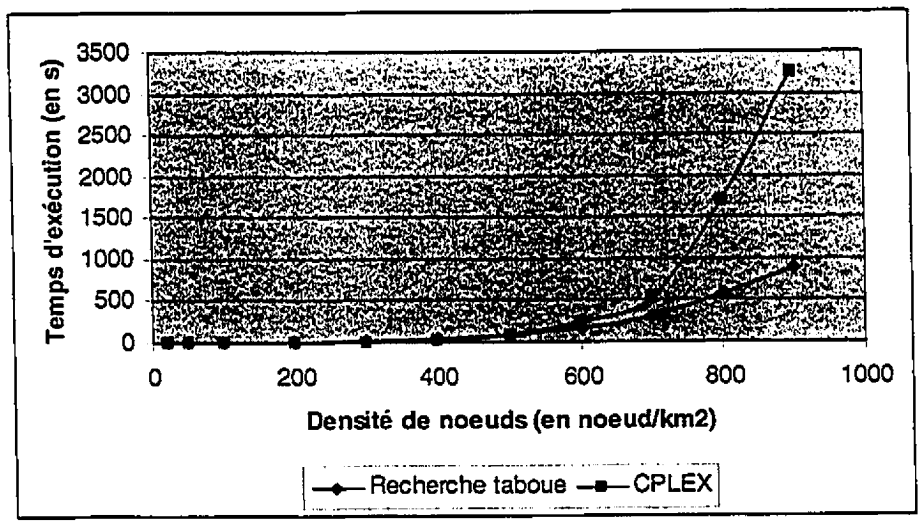

Figure 5.17 Comparaison des temps d'exécution de la recherche taboue et CPLEX : Topologie aléatoire

Cette dernière étude nous a permis de faire les mêmes conclusions obtenues lors de l'analyse des réseaux avec une topologie en carré. D'une part, les solutions trouvées par la recherche taboue sont d'une très bonne qualité en terme de coût des grappes et en terme de temps d'exécution, et d'autre part, l'approche basée sur la recherche taboue se comporte bien avec l'extensibilité du réseau, et ceci dans des réseaux pouvant avoir une topologie en carré ou une topologie aléatoire.

Comparaison des approches répartie et centralisée

La raison qui nous a poussée à proposer une approche centralisée de la formation des grappes est notre constatation que l'approche répartie ne donne pas une solution optimale. L'objectif est de comparer les deux approches afin de valider cette hypothèse. Nous avons considéré un réseau dont la taille varie entre 9 et 100 nœuds. La topologie du réseau est en carré. C'est la même topologie utilisée à la section 5.1.2 où nous avons analysé les résultats de l'approche répartie. Tous les nœuds du réseau sont considérés actifs et se situent d'une manière uniforme aux sommets des carrés d'une grille couvrant la zone étudiée. Les deux

approches utilisent les mêmes valeurs des paramètres des requis des applications, i.e. les valeurs des paramètres *pas* et *sem.* Nous avons modifié la classe qui implémente le comportement des nœuds dans le simulateur OMNET++ pour qu'elle puisse calculer le coût de chaque grappe suivant la même relation de coût utilisée dans l'approche centralisée. Nous avons utilisé les résultats de la recherche taboue que nous avons paramétrée de la manière expliquée aux sections précédentes. Par exemple, le nombre d'itérations maximum prend la valeur du nombre de nœuds du réseau multiplié par 8.

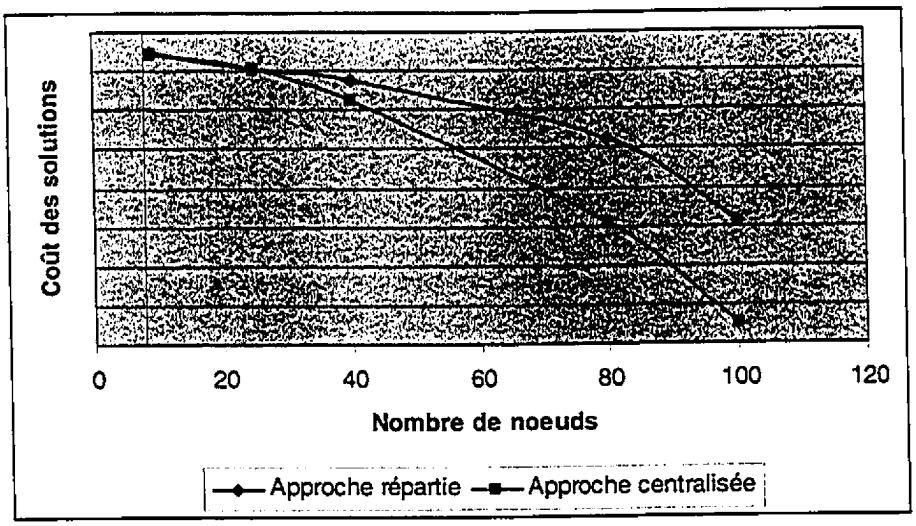

Figure 5.18 Coût des solutions pour les deux approches répartie et centralisée

La Figure 5.18 illustre les variations de coûts des solutions des deux approches répartie et centralisée en faisant varier le nombre de nœuds. Les résultats révèlent qu'effectivement les grappes formées par l'approche centralisée sont nettement meilleures que celles formées par l'approche répartie, et ceci spécialement pour des tailles de réseaux plus grandes. Plus la taille est grande, plus l'écart entre les deux approches est grand. Par exemple, le coût total des grappes dans un réseau de 100 nœuds a été diminué de presque 6 fois, donc, l'approche centralisée a fait un gain de 500% du coût des grappes par rapport à l'approche répartie. Ce gain est de plus en plus faible pour les réseaux de petite taille. Par exemple, la différence entre les deux approches n'est que de 0,004 % pour un réseau avec 9 nœuds.

Cette analyse nous a permis de valider notre hypothèse sur la qualité des grappes formées par les deux approches. Nous concluons que l'approche centralisée conduit à la

formation de grappes de très bonne qualité comparée à celle de l'approche répartie. Ceci est justifié par le fait que, dans l'approche répartie, les nœuds ont une visibilité limitée à leurs nœuds voisins alors que l'algorithme centralisé optimise la formation des grappes globalement. Néanmoins, nous avons remarqué aussi que le temps d'exécution de l'algorithme centralisé est supérieur à celui de l'algorithme réparti. Il faut noter que, tout au long des expériences, le coût de la solution initiale trouvée avant de procéder aux itérations de la recherche taboue était plus petit que le coût total des grappes formées par l'approche répartie. Ceci dit, un système de capteurs peut réduire la consommation d'énergie en n'implémentant que l'algorithme de la solution initiale qui donne une meilleure formation de grappes en un temps comparable à celui de l'approche répartie.

Comparaison des énergies consommées pour former les grappes dans les deux approches (centralisée et répartie)

Les deux approches répartie et centralisée ont besoin d'échanger des messages dans le réseau afin d'être capable de former les grappes. En effet, d'une part, l'approche répartie procède à l'envoi d'un message de la formation des grappes et d'un message de la procédure de négociation au besoin. La consommation de l'énergie due à cet effet est analysée dans les sections précédentes. D'autre part, l'approche centralisée implique l'envoi des nœuds du réseau des données pertinentes permettant au nœud central de procéder au calcul des grappes. Ces deux énergies consommées dans les deux approches ne doivent pas être négligeables. D'où l'importance de mener des études permettant de comparer la consommation d'énergie due à la collecte des informations sur les nœuds pour résoudre le problème d'optimisation dans l'approche centralisée, et la consommation d'énergie due aux messages échangés dans la phase de formation des grappes dans l'approche répartie.

Dans l'approche centralisée, les informations dont le nœud central a besoin pour calculer le problème d'optimisation sont :

- les coordonnées de chaque nœud pour pouvoir construire le graphe G^r et calculer les coûts des grappes. Les informations sur les coordonnées des nœuds ne sont envoyées qu'une seule fois lors de la première requête émise par l'application. Vu que ces informations ne changent que si un nœud tombe en panne, la communication de ces données au nœud central ne devra consommer qu'une faible quantité d'énergie.

- les mesures de chaque nœud pour pouvoir construire le graphe G^r. Ces informations devront être communiquées au nœud central à chaque fois qu'il reçoit une requête d'une application spécifiant la valeur du paramètre *sem*.
- le fait que le nœud est actif ou non. Chaque nœud capteur peut calculer ce flag en sachant son énergie résiduelle et l'énergie prédite. Cependant, pour calculer celle-ci, le nœud a besoin de connaître le temps total d'exécution de la requête émise par l'application. Initialement, le nœud central devra envoyer un message à tous les nœuds afin qu'il obtienne la valeur de ce flag pour chaque nœud.

Afin de mener cette analyse, nous avons utilisé le simulateur OMNET++ pour pouvoir calculer les énergies consommées pour les deux approches. La classe modélisant le comportement des nœuds a été modifiée pour qu'elle puisse prendre en compte les échanges de messages entre le nœud central et les différents nœuds du réseau avant que le nœud central puisse entamer les calculs des grappes optimales. Nous avons utilisé le même modèle de consommation d'énergie implémenté lors de l'analyse de l'approche répartie.

Afin d'effectuer l'analyse pour l'approche centralisée, nous avons procédé à des simulations d'un protocole qui permet au nœud central d'envoyer des messages à chaque nœud du réseau et vice versa, i.e. chaque nœud du réseau doit être capable d'envoyer un message au nœud central. Dans une première phase, le nœud central envoie à chaque nœud du réseau un message contenant le temps global d'exécution de la requête, i.e. le seul paramètre dont chaque nœud a besoin pour calculer sa prédiction de la consommation d'énergie. Dans une deuxième phase, chaque nœud va envoyer au nœud central un message contenant ses coordonnées, sa mesure et son flag de couverture. Le modèle du réseau est multihop, i.e. les nœuds utilisent leurs voisins afin de communiquer avec le nœud central. À chaque simulation, nous faisons varier le nombre de nœuds du réseau et nous mesurons l'énergie consommée pendant les deux phases du protocole.

La Figure 5.19 illustre les variations d'énergie consommée pour former les grappes dans les approches répartie et centralisée. Les résultats montrent que l'approche répartie a besoin de moins d'énergie pour former les grappes que l'approche centralisée. L'écart entre les deux courbes devient de plus en plus grand quand le nombre de nœuds du réseau augmente. En effet, l'écart est multiplié par 133 fois en faisant augmenter la taille du réseau de 10 fois. Ceci est justifié par le fait que l'approche centralisée a besoin de faire échanger

beaucoup de messages afin de centraliser les données en un seul nœud capable de calculer les grappes optimales.

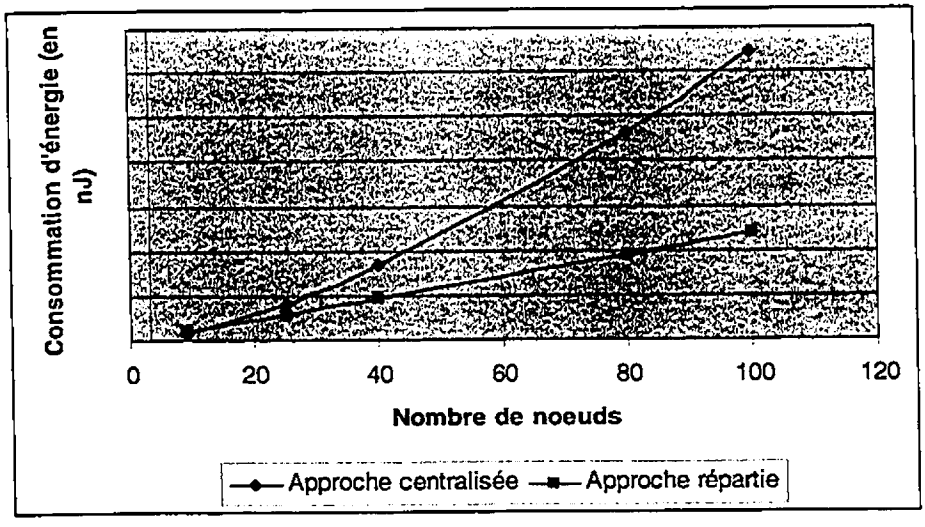

Figure 5.19 Comparaison des énergies consommées pour former les grappes dans les deux approches

Nous pouvons conclure que l'approche centralisée est moins efficace que l'approche répartie en terme de consommation d'énergie pour former les grappes. Cependant, l'approche centralisée pourra être plus efficace dans la phase de collecte des données. C'est ce que nous voudrions mesurer dans les expériences de la section suivante.

Comparaison de l'énergie totale consommée dans les deux approches (centralisée et répartie)

Nous avons mené d'autres simulations afin de comparer l'énergie totale consommée (i.e. l'énergie consommée par les nœuds dans la phase de la formation des grappes et l'énergie consommée dans la phase de collecte des données) dans les deus approches centralisée et répartie. Ces énergies sont comparées à celle consommée par la méthode TAG référée dans les sections précédentes. Nous avons utilisé la même topologie et la même requête décrites dans les sections précédentes. La Figure 5.20 montre l'énergie totale consommée en variant le nombre de nœuds du réseau. Pour les trois méthodes, l'énergie totale consommée croit proportionnellement au nombre de nœuds. Ceci est trivial vu que plus un réseau contient de nœuds, plus les méthodes vont générer des messages, et plus

l'énergie totale consommée sera grande. Cependant, les pentes des deux courbes de nos approches centralisées et réparties croient d'une manière raisonnable avec le nombre de nœuds comparées à TAG. La deuxième remarque est que, pour ce genre de requête, les nœuds dans l'approche centralisée consomment moins d'énergie totale comparée à la consommation d'énergie des nœuds dans l'approche répartie. Ceci est justifié par le fait que les grappes sont formées plus efficacement dans l'approche centralisée. Par conséquent, les nœuds génèrent moins de messages et exécute le filtrage d'une manière plus efficace.

Nous pouvons conclure que l'approche centralisée est moins efficace que l'approche répartie en terme de consommation d'énergie pour former les grappes. Cependant, une fois celles-ci construites, la phase de collecte des données est plus efficace dans l'approche centralisée comparée à l'approche répartie, vu que les grappes seront formées d'une manière optimale. Ceci dit, l'approche centralisée est plus efficace quand la phase de collecte des données est longue. Pour des requêtes dont le temps global d'exécution est court, l'approche répartie sera plus efficace.

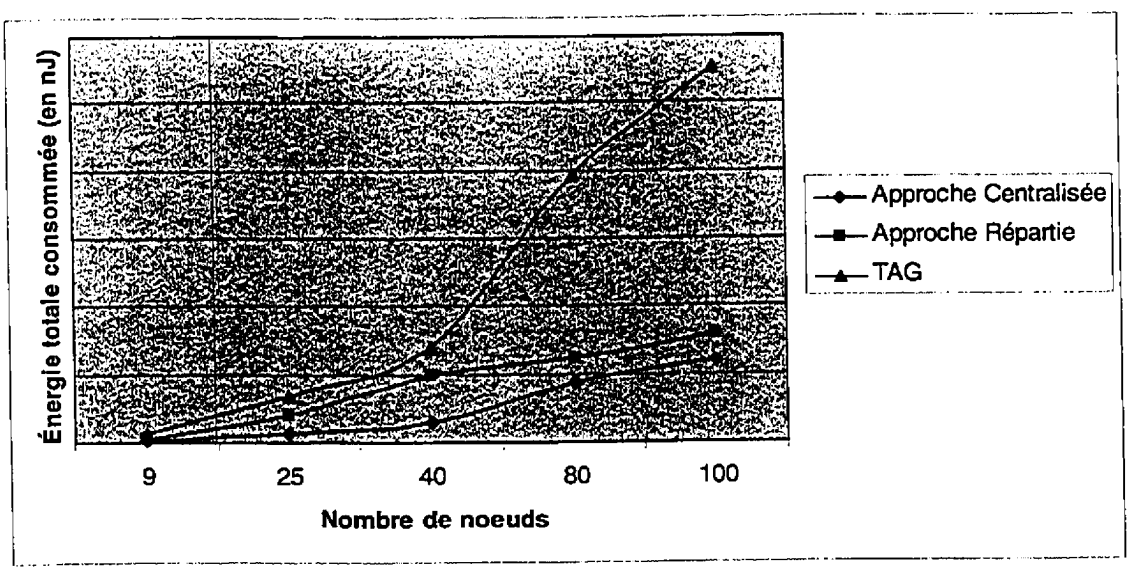

Figure 5.20 Comparaison des énergies totales consommées dans les deux approches

Comparaison de l'approche basée sur la recherche taboue (RT) avec une autre approche basée sur la méta-heuristique 'Recuit Simulé'

La méta-heuristique *Recuit Simulé (RS)* est une approche probabiliste. La transition d'une solution x vers une solution y de son voisinage est acceptée avec une probabilité calculée de la manière suivante:

$$P(x \rightarrow y) = \min\left(1, \exp^{\frac{f(x)-f(y)}{c_i}}\right) \qquad (5.1)$$

Avec $f()$ est la fonction de coût et ci est un paramètre de contrôle, appelé la température. La valeur de ce paramètre est initialement grande et diminue graduellement à chaque itération jusqu'à ce qu'elle atteigne une petite valeur finale. Si N est le nombre d'itérations et si la séquence de la valeur du paramètre c est $c_1, c_2 \ldots c_N$, alors l'algorithme s'écrit de la manière présentée à la Figure 5.21.

Généralement, la séquence de la valeur du paramètre c est déterminée de la manière suivante :

$$c_{i+1} = \frac{c_i}{1+\beta \times c_i} \qquad i = 1,2 \ldots N-1 \qquad (5.2)$$

avec β est une constante positive, c_1 et c_N sont la valeur initiale et la valeur finale de c.

Nous nous sommes inspirés des formules présentées par Osman et al. [OSM89] pour déterminer les valeurs de ces paramètres de la manière suivante:

$$\beta = \frac{c_1 - c_N}{c_1 \times c_N \times (N-1)} \qquad (5.3)$$

$$c_1 = \frac{\sum_{i=1}^{m} d_i}{5m} \qquad \text{et} \qquad c_N = 1 \qquad (5.4)$$

où m désigne le nombre de noeuds du réseau et d_i la distance du noeud i au noeud collecteur.

La solution initiale est choisie en utilisant le même algorithme que celui de la recherche taboue (l'algorithme est présenté à la Figure 4.4). La solution suivante y est choisie d'une manière aléatoire parmi les solutions dans le voisinage de la solution courante x. Ce voisinage est déterminé de la même façon que la détermination du voisinage des solutions dans la recherche taboue (la détermination des mouvements est présentée à la

section 4.2.3 du chapitre 4). Les algorithmes de RS ont été implémentés dans le même environnement et avec les mêmes outils que les algorithmes de la recherche taboue. Pour des fins de comparaison, nous avons considéré des réseaux qui ont une topologie en carré et dont les nœuds sont tous actifs (i.e. peuvent assurer la couverture). Nous avons fixé les valeurs des paramètres fournis par l'application (i.e. les paramètres de la qualité de service : *pas* et *sem*).

Étape 1: Initialisation
i:=O
Choisir une solution initiale aleatoire x
Étape 2: (Recherche au voisinage)
i: = i + 1
Choisir aléatoirement une solution *y* du voisinage de la solution courante *x*
Remplacer la solution *x* par *y* avec une probabilité définie en 5.1
Étape 3 : (Terminaison)
Si i >= N alors termine l'algorithme
Sinon retourne à l'étape 2

Figure 5.21 Algorithme général de la méta-heuristique Recuit Simulé

La Figure 5.22 montre les variations de coûts des solutions en faisant varier le nombre de nœuds du réseau pour les deux approches RT et RS. Nous constatons que les solutions trouvées par RT sont toujours meilleures que celles trouvées par RS et que, plus la taille du réseau est grande, plus la qualité des solutions de RT est meilleure. Ceci est justifié principalement par le fait que, dans RT, l'algorithme analyse tout le voisinage de la solution courante au lieu de choisir une solution d'une manière aléatoire de ce voisinage. Le résultat de ces expériences a montré l'efficacité de notre approche basée sur la recherche taboue.

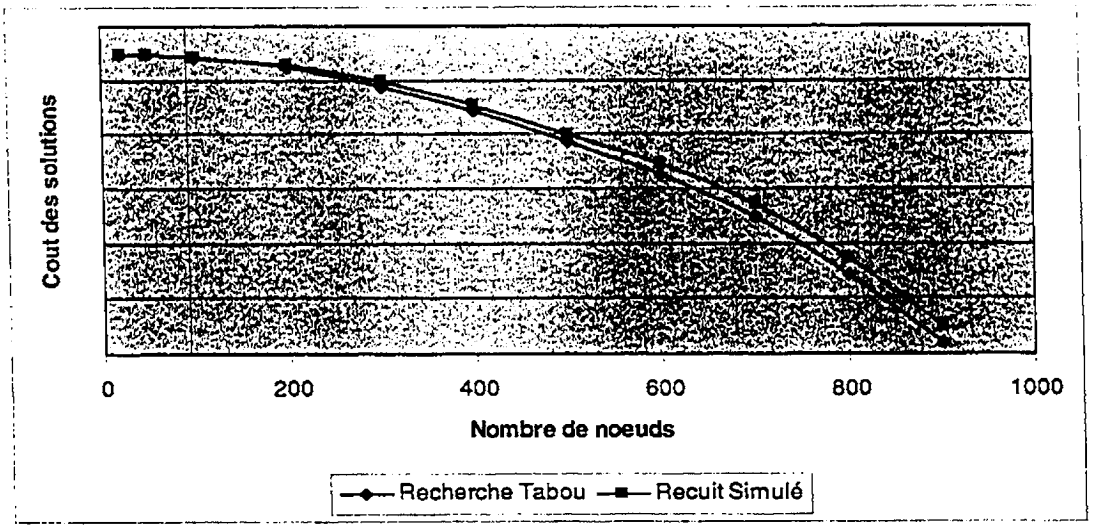

Figure 5.22 Comparaison des coûts des solutions (Recherche Taboue et Recuit Simulé)

CHAPITRE VI
CONCLUSION

Dans ce chapitre final, nous allons tout d'abord présenter nos plus importantes contributions dans le domaine de l'optimisation de la consommation d'énergie par les mécanismes de la collecte des données dans les réseaux de capteurs sans fil. Ensuite, nous allons montrer les limitations des travaux. Enfin, nous présenterons les indications de recherche future.

6.1 Synthèse des travaux

Le sujet que nous avons choisi pour notre thèse a une très grande importance pour le déploiement des réseaux de capteurs. Il est parmi les sujets les plus abordés dans la littérature. Ceci est dû à plusieurs raisons. Premièrement, vu que les systèmes de capteurs sont déployés pour prendre des mesures et les envoyer aux applications, le mécanisme de la collecte des données constitue la partie la plus importante de ces systèmes. Deuxièmement, les capacités limitées des capteurs en énergie rendent l'optimisation de la consommation d'énergie cruciale. Enfin, la complexité des modèles mathématiques et le grand nombre de paramètres de ces problèmes rendent leur résolution non triviale.

Dans un premier temps, nous avons étudié les modèles existants de la consommation d'énergie dans les réseaux de capteurs. Ces modèles se sont avérés inadéquats à la conception d'un mécanisme de collecte de données. En effet, la majorité de ces modèles ne considèrent pas le changement des états d'un capteur qui est toujours actif en train de capter de l'information ou de transmettre de l'information. L'analyse de ces modèles nous a permis de trouver une formulation mathématique générale de la consommation d'énergie. Cette formulation a été utilisée afin de définir la carte des énergies qui combine la prédiction de l'énergie consommée dans un intervalle de temps futur avec l'énergie résiduelle.

Nous avons aussi étudié et analysé les mécanismes de collecte des données existants. Il s'est avéré que la majorité de ces mécanismes ne permettent pas de réduire la consommation d'énergie d'une manière efficace du fait que ces derniers ne considèrent qu'un seul aspect à la fois des réseaux de capteurs. À partir de ces faiblesses, nous avons conçu une approche

novatrice qui combine les exigences des applications avec la carte des énergies des capteurs. Cette approche constitue une première contribution très importante dans la littérature vu qu'elle permet de réduire la consommation d'énergie d'une manière significative et permet d'alléger le traitement des applications en ne leur fournissant que les données pertinentes. À notre connaissance, c'est la première fois qu'une telle approche voit le jour dans la littérature.

Cette nouvelle approche utilise la formation des grappes avec des contraintes liées aux exigences des applications et à la carte des énergies. Afin de résoudre le problème de la formation des grappes, nous avons conçu deux nouveaux algorithmes : Un algorithme réparti et un autre centralisé. L'algorithme réparti utilise les capacités de traitement de tous les nœuds du réseau afin de former les grappes de proche en proche à partir du nœud collecteur. L'algorithme centralisé a été conçu pour pallier les faiblesses de l'algorithme réparti qui ne permet pas d'optimiser la consommation d'énergie vu que les nœuds n'ont qu'une vision restreinte limitée aux nœuds voisins. Ainsi, nous avons étudié les approches centralisées existantes de la formation des grappes. L'analyse des lacunes de ces algorithmes nous a permis de concevoir un algorithme novateur de formation de grappes. La particularité de cet algorithme est le fait que ni les têtes de grappes ni le nombre de grappes ne sont prédéterminés, ce qui permet d'exploiter tous les scénarios possibles, contrairement aux approches existantes. Le problème a été modélisé comme celui du partitionnement d'un hypergraphe qui est NP-Complet. Sa résolution utilise une adaptation intelligente de la méta-heuristique Recherche Taboue.

Afin de mesurer ses performances, l'approche centralisée a été simulée à l'aide de l'outil OMNET++. Les résultats, comparés à ceux de l'approche existante TAG, montrent que notre approche réduit d'une manière significative la consommation d'énergie. L'approche centralisée a été implémentée à l'aide du langage de programmation C++ avec des librairies spécialisées dans les traitements des graphes. Les résultats, comparés à ceux d'une deuxième résolution basée sur CPLEX et d'une troisième résolution basée sur la méta-heuristique Recuit Simulé, montrent que l'adaptation de la recherche taboue donne des grappes avec une meilleure qualité et que notre approche centralisée se comporte d'une manière efficace avec l'extensibilité des réseaux de capteurs sans fil. La comparaison des deux approches centralisée et répartie a révélé que l'approche centralisée est plus efficace

dans le cas où le temps d'exécution des requêtes est long. Dans le cas contraire, l'approche répartie est plus efficace.

6.2 Limitations des travaux

L'analyse des performances de l'approche répartie de la formation des grappes est basée sur les résultats obtenus en procédant à des simulations des algorithmes avec le simulateur OMNET++ reconnu par la communauté scientifique pour ce type de réseaux. Cependant, il s'est avéré que l'environnement ne fournit pas une interface usager facilitant la création des réseaux. Nous étions limités par cet outil et, par conséquent, seuls les réseaux avec des topologies simples (en carré et en ligne) et avec un nombre de nœuds ne dépassant pas une centaine de nœuds ont été considérés. Une analyse approfondie nécessiterait une simulation des réseaux avec plus de nœuds et avec des topologies proches de la réalité (par exemple la topologie aléatoire). Une autre limitation liée aussi à l'environnement de simulation, est le fait que nous avons supposé que tous les nœuds des réseaux simulés sont actifs (i.e. peuvent assurer la couverture de leurs zones pendant toute la durée de l'exécution de la requête). C'est une supposition valide pour un réseau nouvellement déployé dont les batteries des nœuds fonctionnent à pleine capacité. Cependant, l'analyse approfondie de nos algorithmes doit prendre en considération des réseaux dont certains nœuds ne peuvent pas assurer la couverture de leurs zones due à leur carte d'énergie.

Notre travail de recherche traite des réseaux de capteurs sans fil fixes dont les localisations des nœuds demeurent constantes le long de l'exécution des requêtes. Les défis de ce genre de réseaux de capteurs sont nombreux, complexes et très importants pour le déploiement de ces réseaux à grande échelle. Cependant, il existe une catégorie étendue d'applications qui utilisent des réseaux de capteurs sans fil dont les nœuds sont mobiles. Le fait de considérer les nœuds fixes dans la modélisation et la conception de nos algorithmes limite leurs utilisations à ce genre de réseaux. En effet, le caractère fixe des nœuds constitue la base de la formulation du problème de la formation des grappes. Dans un milieu où les nœuds sont mobiles, les grappes ne satisferont plus les critères de la formation des grappes dans le cas où les distances entre les nœuds changent.

Dans la fonction objectif du modèle mathématique de la formation des grappes, nous avons considéré une formule simplifiée du coût des grappes. C'est une bonne approximation

du coût dans un réseau dense où le protocole de routage peut facilement trouver un chemin, qui peut être considéré comme une ligne droite, entre un nœud et le nœud collecteur. Cette approximation n'est plus valide dans le cas où la densité des nœuds du réseau est faible. Dans ces cas, la fonction de coût des grappes doit considérer les distances entre les nœuds constituant le chemin de routage.

6.3 Indications de recherche future

Les limitations évoquées dans la section précédente nous amènent à proposer les travaux futurs suivants. Premièrement, les algorithmes de l'approche répartie doivent être simulés dans un autre environnement de simulation. Le principal aspect à considérer dans le choix de simulateur est sa convivialité par rapport à la création des réseaux avec une variété de tailles et de topologies. Ce premier travail aura comme objectif de valider les résultats obtenus. Deuxièmement, nous proposons d'étudier et d'analyser les modèles de la mobilité des nœuds dans les réseaux de capteurs. Ces modèles aideront à adapter les algorithmes conçus pour les réseaux de capteurs sans fil mobiles. Le résultat escompté de ce travail sera une conception et une analyse de performance d'un mécanisme de collecte de données qui combine les exigences des applications et la carte des énergies des nœuds dans un réseau de capteurs mobiles. Enfin, le troisième travail de recherche que nous proposons est lié à la considération des réseaux de capteurs avec des densités faibles. Cette caractéristique du réseau rende les approximations faites lors de la formulation du problème mathématique de formation des grappes invalide. Ce dernier travail consistera en la reformulation de la fonction objectif en considérant les distances entre les nœuds constituant les chemins entre les nœuds. Le résultat escompté de ce travail de recherche est d'évaluer les performances de notre approche centrale dans des réseaux de capteurs sans fil dont la densité est faible.

BIBLIOGRAPHIE

[ABO05] C. Abondo, *Gestion de la Qualité de Service dans les système mobiles de prochaines générations*, Thèse de doctorat, École Polytechnique de Montréal, Juin 2005.

[AGA02] P.K. Agarwal, C.M. Procopiuc, "Exact and Approximation Algorithms for Clustering", *Algorithmica, New York*, USA, vol. 33, no. 2, June 2002, pp. 201-226.

[ALE05] *The Application Level Events (ALE) Specification*, Version 1.0, EPCglobal Ratified Specification, Version of September 15, 2005.

[BAR02] R. Barr, J. Bicket, D. Dantas, B. Du, T. Kim, B. Zhou and E. Sirer, "On the Need for System-Level Support for Ad Hoc and Sensor Networks", *Operating System Review*, vol. 36, no. 2, April 2002, pp. 1-5.

[BAS04] P. Basu, J. Redi, "Effect of overhearing transmissions on energy efficiency in dense sensor networks", *Third International Symposium on Information Processing in Sensor Networks*, Berkeley, California, USA, Avril 2004, pp. 196 – 204.

[BEA03] J. Beaver, M.A. Sharaf, A. Labrinidis, P.K. Chrysanthis, "TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation", *Proceedings of the third ACM MobiDE Workshop*, San Diego, California, USA, September 2003, pp. 69-76.

[BET06] B. Betts, "Smart Sensors", *IEEE Spectrum Journal*, Avril 2006, vol. 43, no. 3, pp. 50-53.

[BOL85] B. Bollbas, *Random Graphs*, Academic Press, 1985.

[BOOST] http://boost.org/libs/random/index.html, September 2007.

[CAI05] C.H. Caicedo, M. Zefran, "Energy cost of coordination algorithms for wireless mobile sensor networks", *Proceedings of IEEE International Conference of Networking, Sensing and Control*, Tucson, Arizona, USA, March 2005, pp. 419 – 424.

[CAL04] E.H. Callaway, *Wireless Sensor Networks: Architectures and Protocols*, Auerbach Publications, 2004.

[CHI00] C. F. Chiasserini and R. R. Rao, "Routing protocols to maximize battery efficiency", *Proceedings of MILCOM* 2000, October 2000, Los Angeles, California, USA, vol. 1, pp. 496–500.

[CHO01] D.N. Chorafas, *Integrating ERP, CRM, supply chain management, and smart materials*, Auerbach Publications, 2001.

[FEE01] L. M. Feeney and M. Nilsson, "Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment", *Proceedings of IEEE INFOCOM*, Anchorage, Alaska, Avril 2001, vol. 3, pp. 1548-1557.

[GHI02] S. Ghiasi , A. Srivastava, X. Yang, M. Sarrafzadeh, "Optimal Energy Aware Clustering in Sensor Networks", *Sensors, 2002*, vol. 2, no. 7, pp. 258-269.

[GIR05] A. Giridhar and P.R. Kumar, "Towards a Theory of In-Network Computation in Wireless Sensor Networks", *IEEE Communications Magazine*, April 2006, vol. 44, pp. 98-107.

[GLO93] F. Glover, E. Taillard, D. de Werra, "A user's guide to tabu search ", *Annals of Operations Research*, May 1993, vol. 41, pp. 3-28.

[GON85] M. Gondran, M. Minoux, *Graphes et algorithmes*, 2ième edition, Editions Eyrolles, Paris, 1985.

[HEI02] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application specific protocol architecture for wireless microsensor networks", *IEEE Transactions on wireless Communications*, October 2002, vol. 1, no. 4, pp. 660-670.

[JER02] S. Jeremy, *The boost graph library: user guide and reference manual*, Addison-Wesley, Toronto, 2002.

[KAN02] Kanugo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D.,Silverman, R., Wu, A. Y., "A Local Search Approximation Algorithm for k-Means Clustering", *Proceedings of the 18th annual ACM Symposium on Computational Geometry*, Barcelona, Spain, 2002, pp. 10-18.

[LEE04] J.J. Lee, B. Krishnamachari, C.-C.J. Kuo, "Impact of heterogeneous deployment on lifetime sensing coverage in sensor networks", *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, California, USA, *Oct. 2004*, pp. 367 – 376.

[LIA07] W. Liang, Y. Liu, "Online data gathering for maximizing network lifetime in sensor networks", *IEEE Transactions on Mobile Computing*, Jan. 2007, vol. 6, no. 1, pp. 2-11.

[LIU03] T. Liu and M. Martonosi, "Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems", *Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, June 2003, San Diego, California, USA, pp. 107-118.

[ILOG] http://www.ilog.com/products/cplex/, February 2007.

[MAD02] S.R. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, "TAG: Tiny Aggregation service for ad-hoc sensor networks", *Proceedings of the 5th symposium on Operating systems design and implementation*, Boston, Massachusetts, USA, December 2002, pp. 131-146.

[MIN04] R.A.F Mini. A.A.F Loureiro and B. Nath, "The distinctive design characteristic of a wireless sensor network: The energy map", *Computer Communications*, June 2004, vol. 27, no. 10, p 935-945.

[MOU06] O. Moussaoui , A. Ksentini, M. Naimi, M. Gueroui, "A novel clustering algorithm for efficient energy saving in wireless sensor networks", *Proceedings of International Symposium on Computer Networks* , Istanbul, Turkey, June 2006, pp. 66-72.

[POO01] R.D. Poor, *Embedded networks: Pervasive, low-power, wireless connectivity*, Ph.D. dissertation, 2001, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

[OSM89] H.Osman and C.N.Potts, "Simulated annealing for permutation flow-shop scheduling", *OMEGA*, Elsevier, Royaume-Uni, 1989, vol. 17, no. 6, pp. 551-557.

[RAB00] J.M. Rabaey, M.J. Ammer, J.L. da Silva, D. Patel, S. Roundy, "PicoRadio supports ad hoc ultra-low power wireless networking", *IEEE Computer*, July 2000, vol. 33, no. 7, pp. 42–48.

[RAG03] S. Raghuwanshi, A. Mishra, "A self-adaptive clustering based algorithm for increased energy-efficiency and scalability in wireless sensor networks", *IEEE 58th Vehicular Technology Conference*, Orlando, Florida, USA, Oct. 2003, vol. 5, pp. 2921-2925.

[SIN98] S. Singh and C. S. Raghavendra, "PAMAS Power aware multi-access protocol with signalling for ad hoc networks", *ACM SIGCOMM Computer Communication Review*, New York, USA, July 1998, vol. 28, no. 3, pp. 5–26.

[WEI03] S.A. Weis, S.E Sarma, R.I. Rivest and D.W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems", *Security in Pervasive computing* 2003, Springer-Verlag , 2004, pp. 201-212.

[YAU04] Y. Yau, B. Krishnamachari, and V.K. Prasanna, "Issues in Designing Middleware for Wireless Sensor Networks", *IEEE Network Journal*, January/February 2004, vol. 18, no. 1, pp. 15-21.

[YOO04] S. Yoon and C. Shahabi, "An Energy Conserving Clustered Aggregation Technique Leveraging Spatial Correlation", The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, Poster, Santa Clara, California, USA, October 2004.

[YOU04] O. Younis, S. Fahmy, "Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach", *Proceedings of Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, Hong Kong, China, March 2004, vol. 1, pp. 640.

[ZHO01] L.C. Zhong, R. Shah, C. Guo, J. Rabaey, "An ultra-low power and distributed access protocol for broadband wireless sensor networks", *IEEE Broadband Wireless Summit*, Las Vegas, N.V., USA, May 2001.