Université de Franche-Comté Besançon
U.F.R Sciences et Techniques
École Doctorale SPIM

Thèse de Doctorat

Spécialité Informatique

présentée par

LEHSAINI Mohamed

Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique

Soutenue le 1^{er} Juillet 2009 devant le jury :

Président du Jury :	M.A. CHIKH	Maître de Conférences, Université A.B.B Tlemcen
Directeurs de Thèse :	H. GUYENNET	Professeur, Université de Franche-Comté
	M. FEHAM	Professeur, Université A.B.B Tlemcen
Rapporteurs :	P. LORENZ	Professeur, Université de Haute Alsace
	A. BENYETTOU	Professeur, Université des Sciences et de la Technologie d'Oran
Examinateurs :	N. ZERHOUNI	Professeur, Ecole Nationale de Mécanique et des Microtechniques de Besançon
	A. HAMMAD	Maître de Conférences, Université de Franche-Comté

Remerciements

Je tiens à remercier en premier lieu Mr. Hervé GUYENNET et Mr. Mohammed FEHAM, mes directeurs de thèse pour leur sympathie, leur disponibilité, leurs idées, leurs conseils et leurs encouragements qui m'ont permis de mener à bien cette thèse.

J'adresse aussi mes très sincères remerciements à Mr. Mohamed Amine CHIKH de me faire l'honneur de s'intéresser à ce travail et d'avoir présidé le jury.

J'exprime ensuite ma plus profonde gratitude à Mr. Pascal LORENZ et Mr. Abdelkader BENYETTOU qui ont accepté de rapporter cette thèse malgré leur emploi du temps surchargé.

Je tiens à remercier vivement Mr. Noureddine ZERHOUNI et Mr. Ahmed HAMMAD pour l'intérêt qu'ils ont bien voulu porter à ce travail, j'ai beaucoup apprécié leur participation au jury de cette thèse malgré le long voyage.

Ce travail a été réalisé au sein du laboratoire d'Informatique de l'université de Franche-Comté (LIFC) et au laboratoire Systèmes et Technologie de l'Information et de la Communication (STIC) de l'université A.B de Tlemcen. Je tiens donc à remercier les responsables de ces deux laboratoires de m'avoir accueilli et donné l'opportunité de réaliser ce travail de thèse.

Un grand merci à tous les membres du laboratoire d'informatique de l'université de Franche-Comté qui m'ont offert un excellent cadre de travail ainsi qu'un séjour extrêmement agréable.

Merci à tous mes collègues de l'équipe Réseaux de Capteurs : Hung-Cuong LE, Pamba CAPO-CHICHI, David MARTINS, et Kamal BEYDOUN pour leur sympathie durant le temps où on a travaillé ensemble et pour les discussions de recherche dans le domaine des réseaux de capteurs. Merci également à Nabil ELMARZOUQI pour ses discussions et ses encouragements.

Merci à Mohamed Amine mon collègue de l’université de Tlemcen pour sa disponibilité et son soutien indéfectible dans les moments difficiles.

Un merci pudique à toute ma famille pour son soutien, à la mémoire de mes parents, et à mon petit Imed qui m’a manqué durant mon séjour à Besançon.

Table des matières

Acronymes

3hBAC	3-hop Between Adjacent Cluster-heads
ACOS	Area-based Collaborative Sleeping
ADCs	Analog-to-Digital Converter
ADS	Area Dominating Set Protocol
APTEEN	Adaptive Threshold-sensitive Energy Efficient Network protocol
B-MAC	Berkeley Medium Access Control
BCDCP	Base-Station Controlled Dynamic Clustering Protocol
CBRP	Cluster Based Routing Protocol
CCP	Coverage Configuration Protocol
CDS	Connected Dominating Set
CH	Clusterhead
CKA	Cluster-based Algorithm
CSMA/ CD	Carrier Sens Multiple Access with Collision Detection
CSMA/ CA	Carrier Sens Multiple Access with Collision Avoidance
CTS	Clear To Send
DARPA	Defense Advanced Research Projects Agency
DCA	Distributed Clustering Algorithm
DMAC	Distributed Mobility Adaptive Clustering
DSMAC	Dynamic Sensor Medium Access Control
DSN	Distributed Sensor Network
EAR	Eavesdrop And Register
FDMA	Frequency Division Multiple Access
FND	First Node Dies

GAF	Geographic Adaptive Fidelity
HEED	Hybrid Energy-Efficient Distributed Clustering
HNA	Half of the Nodes Alive
JBREWS	Joint Biological Remote Early Warning System
Idle	Ecoute non actif
IP	Integer Programming
k-CS	k-Coverage Set
k-CCS	k-Coverage Connected Set
LANs	Local Area Network
LCC	Least Cluster Change
LEACH	Low Energy Adaptive Clustering Hierarchy
LEACH-C	Low Energy Adaptive Clustering Hierarchy Centralized
LND	Last Node Dies
LPA	Linear Programming Algorithm
LRWPAN	Low Rate Wireless Personal Area Networks
M-LEACH	Multi-hop Low Energy Adaptive Clustering Hierarchy
MAC	Medium Access Control
MCDS	Minimum Connected Dominating Set
MIS	Maximal Independant Set
MOBIC	Lowest Relative Mobility Clustering Algorithm
MPR	Multi-Point Relais
NES	Neighbor Elimination Scheme
NS	Network Simulator
OLSR	Optimized Link State Routing
OGDC	Optimal Geographic Density Control
PDA	Personal Digital Assistant
PEAS	Probing Environment and Adaptive Sleeping
PECAS	Probing Environment and Collaborating Adaptive Sleeping
PEGASIS	Power-Efficient Gathering in Sensor Information Systems
PKA	Pruning-based Algorithm

QoS	Quality of Service
RdC	Réseaux de capteurs sans fil
RSSI	Receiver Signal Strength Indication
RTS	Ready To Send ou Request To Send
S-MAC	Sensor Medium Access Control
SMACS	Self-organizing Medium Access Control for Sensor Networks
SRB	Saved ReBroadcast
T-MAC	Timeout Medium Access Control
TDMA	Time Division Multiple Access
TEEN	Threshold-sensitive Energy Efficient sensor Network protocol
VGA	Virtual Grid Architecture routing
WATS	Wide Area Tracking System
WCA	Weighted Clustering Algorithm
WSNs	Wireless Sensor Networks
μAMPS	Micro-Adaptive Multidomain Power Aware Sensors

Table des figures

Liste des tableaux

Introduction générale

Au cours de ces dernières années, la technologie des réseaux sans fil n'a cessé de croître grâce aux développements technologiques dans divers domaines liés à la micro-électronique. En plus, avec l'émergence des Réseaux de Capteurs sans fil (RdC ou WSN : Wireless Sensor Networks), de nouvelles thématiques ont été ouvertes et de nouveaux défis ont vu le jour pour répondre aux besoins des personnes et aux exigences de plusieurs domaines d'application (industriel, culturel, environnemental) : observation de la vie des espèces rares, surveillance de la structure des infrastructures, optimisation de traitement pour les patients, etc. Ces problématiques motivent de nombreux chercheurs. En effet, malgré les avancées remarquables dans ce domaine, il reste encore beaucoup de problèmes à résoudre. Ainsi, de nouveaux protocoles ont été proposés pour traiter le contrôle de l'accès au médium, le routage, etc. dans les réseaux de capteurs. Cependant, la maîtrise de la consommation d'énergie par les réseaux capteurs et la maximisation de leur durée de vie restent les problématiques les plus fondamentales car les capteurs sont de petits composants avec une faible capacité de stockage, de calcul et sont alimentés par des batteries dont la capacité est très limitée et qui sont généralement non rechargeables. Par exemple, dans certaines applications où les capteurs sont déployés dans des zones hostiles, il est difficile voire impossible de changer les batteries. C'est le cas pour les applications conçues pour le réchauffement climatique et la surveillance de la variation de la température au pôle Nord où il n'est pas pratique de mettre en place une équipe ou de l'envoyer chaque fois pour changer les batteries des capteurs. Donc, pour qu'un réseau de capteurs reste autonome pendant une longue durée (quelques mois ou quelques années) et ait par la suite une longévité maximale, il faut que la consommation d'énergie soit prise en compte à tous les niveaux de l'architecture réseaux (de la couche physique à la couche application).

Les réseaux de capteurs sont classés selon l'application et les caractéristiques des capteurs composant le réseau. Ils se distinguent par le modèle de communication, le modèle de transmission de données à la station de base, le modèle de mobilité, etc. Dans ce travail, nous nous

intéressons à deux modèles d'interaction entre les capteurs et la station de base : le modèle de transmission périodique de données et le modèle de détection d'événements pertinents. Dans le premier modèle, les capteurs en mode actifs envoient périodiquement les données collectées à leur cluster-head, qui à son tour les transmettent à la station de base directement ou par un cheminement de cluster-head à cluster-head (CH-à-CH). Le but de ce modèle est d'assurer un monitoring permanent de la zone d'intérêt pour visualiser et analyser l'ensemble des données. Dans le deuxième modèle, un capteur envoie des données à son cluster-head uniquement s'il y a l'occurrence d'un événement pertinent. Ce type de modèle est recommandé pour les applications de surveillance d'événements critiques. Il vise à remonter l'information au centre de contrôle le plus vite possible.

Les caractéristiques particulières des RdC modifient les critères de performances par rapport aux réseaux sans fil traditionnels. Dans les réseaux locaux filaires ou les réseaux cellulaires, les critères les plus pertinents sont le débit, la latence et la qualité de service vu que les nouvelles activités telles que le transfert d'images, transfert de vidéos, et la navigation sur Internet requièrent un débit important, une faible latence, et une bonne qualité de service. En revanche, dans les réseaux de capteurs conçus pour un monitoring permanent de la zone d'intérêt, nous visons à maximiser la longévité des réseaux. De ce fait, la conservation de l'énergie est devenue un critère de performance prépondérant et se pose en premier lieu tandis que les autres critères comme le débit ou l'utilisation de la bande passante sont devenus secondaires. En outre, dans les réseaux de capteurs orientés événements où l'information devra être remontée au centre de contrôle dans un meilleur délai, il est nécessaire d'optimiser conjointement les critères latence et conservation d'énergie car la faible latence peut avoir un impact négatif sur le nombre de communications et par suite sur la consommation d'énergie dans les réseaux.

Organisation de la thèse

L'objectif de cette thèse est de traiter les problèmes de diffusion et de couverture de zone dans les réseaux de capteurs sans fil. Au niveau de la diffusion, nous avons proposé un algorithme pour la diffusion dans un environnement réaliste basé sur l'approche MPR (Multi Points Relais) puis un algorithme d'auto-organisation appelé CSOS (Cluster-based Self-Organization Scheme) basé sur le concept de clustering. Ce dernier permet d'organiser les capteurs en 2-clusters homogènes en taille et de confier aux cluster-heads la responsabilité de communiquer et de transmettre les données collectées par les capteurs à la station de base. L'élection des cluster-heads se fait pé-

riodiquement selon leur poids qui est fonction de leur capacité pour supporter cette tâche. Le poids d'un nœud est calculé grâce aux métriques suivantes : k-densité, énergie restante et mobilité. Nous avons impliqué ces facteurs dans le calcul des poids des capteurs pour générer des clusters stables et éviter par suite les réactions en chaîne qui se déclenchent quand la structure des clusters formés change suite à l'immigration d'un capteur vers un autre cluster, au crash d'un cluster-head ou à l'épuisement de l'énergie.

Au niveau de la couverture, nous avons proposé un algorithme pour assurer la couverture totale de la zone d'intérêt à des degrés différents. Cet algorithme se base sur le même concept que CSOS mais il implique plus de capteurs actifs pour la couverture de zone. En effet, le nombre de capteurs actifs augmente en fonction du degré de couverture de zone appelé k-couverture ou couverture multiple.

Pour tester les performances de nos contributions, nous avons réalisé plusieurs simulations et nous avons comparé les résultats obtenus à ceux déduits d'autres protocoles. En outre, pour tester notre protocole de couverture de zone, nous avons développé une application générique au sein du laboratoire. Cette application consiste à assurer la couverture de zone dans un parc d'une maison de retraite ou clinique pour les personnes dépendantes. Pour mettre en place cette application, nous avons utilisé les capteurs MicaZ de Crossbow. Le choix de la plateforme MicaZ se justifie par la popularité de ce type de capteurs ainsi que la facilité de leur programmation.

Cette thèse s'articule autour de six chapitres.

Le premier chapitre de ce manuscrit présente les contraintes et les caractéristiques liées aux réseaux de capteurs, leurs domaines d'applications, leurs catégories de communication et leurs défis.

Le deuxième chapitre est une étude bibliographique sur les protocoles de diffusion dans les réseaux ad hoc et de capteurs. Dans cette étude, nous avons présenté les différentes approches proposées dans la littérature pour organiser un réseau en clusters et les techniques qui se basent sur le concept des ensembles dominants connectés (CDS).

Le troisième chapitre est notre première contribution. Il analyse l'impact de l'utilisation d'un environnement non idéal sur les performances du protocole de diffusion MPR et présente une version gloutonne distribuée permettant de pallier les limites du protocole MPR dans ce type d'environnement. Pour cela, nous avons utilisé le modèle lognormal au lieu du modèle du

disque unitaire pour modéliser la probabilité de réception d'un message sans erreur par un nœud récepteur.

Le quatrième chapitre, constituant notre deuxième contribution, présente un algorithme d'auto-organisation appelé CSOS. Cet algorithme est basé sur l'approche de clustering pour une diffusion efficace dans les réseaux de capteurs. Au travers des simulations, nous montrons les effets positifs de cet algorithme sur les performances d'un réseau de capteurs par étude comparative à d'autres protocoles de clustering existants. Les simulations ont été réalisées respectivement dans un environnement moins mobile et un autre fortement mobile ; ceci dans le but d'illustrer la robustesse de notre algorithme de clustering face à la mobilité.

Le cinquième chapitre réalise une évaluation des performances du protocole d'auto-organisation CSOS dans un environnement mobile en termes de quantité de données envoyées à la station de base, et le comportement des capteurs durant la durée de vie du réseau. En outre, pour montrer les gains substantiels du protocole proposé, nous avons comparé ses performances à d'autres protocoles existants.

Le sixième chapitre constitue notre troisième contribution dans cette thèse. Après un état de l'art, il propose un protocole de couverture de zone appelé CSA_VS (Cluster-based Scheduling Algorithm- Virtual Sensor). Ce protocole est basé sur l'ordonnancement d'activité des capteurs pour assurer la couverture totale de la zone d'intérêt. Dans ce protocole, nous avons essayé d'impliquer un nombre minimal de capteurs actifs pour garantir les couvertures simple ou multiple de zone. Une mise en œuvre de cette approche est réalisée dans l'annexe A.

Finalement, une conclusion termine ce manuscrit et rappelle les principales contributions élaborées tout au long de ce travail de thèse. Elle présente également les perspectives et implications relatives aux résultats obtenus. Les travaux contenus dans ce document ont été publiés dans plusieurs conférences et journaux dont la liste est disponible dans la bibliographie personnelle.

En annexe, nous présentons le déploiement d'un réseau de capteurs au sein d'un parc d'une maison de retraite ou une clinique pour surveiller les personnes dépendantes et remonter les événements critiques au centre de contrôle lorsqu'ils surviennent.

Chapitre 1

Les réseaux de capteurs : concepts et applications

1.1 Introduction

Les progrès récents dans la technologie des systèmes micro-électromécaniques (Micro Electro-Mechanical Systems MEMS), les communications sans fil, et l'électronique numérique ont permis le développement de petits dispositifs peu coûteux, de faible puissance, et qui peuvent communiquer entre eux, appelés capteurs. Ces dispositifs intègrent une unité d'acquisition de données environnementales (température, humidité, vibrations, luminosité, ...) pouvant être transformés en grandeurs numériques, une unité de traitement permettant d'agréger les données collectées, une unité de stockage, un module de transmission radio, et une source d'alimentation (batterie). Ils coopèrent entre eux pour former une infrastructure de communication appelée réseau de capteurs.

Les réseaux de capteurs se composent généralement d'un grand nombre de petits dispositifs, qui communiquent entre eux via des liens radio pour le partage d'information et le traitement coopératif. Ces dispositifs sont déployés aléatoirement dans une zone d'intérêt pour superviser ou surveiller des phénomènes divers. Après le déploiement initial, les capteurs peuvent s'auto-organiser en une infrastructure réseau appropriée, souvent en mode multi-sauts. Les données collectées par ces capteurs sont acheminées directement ou via un routage multi-sauts à un nœud considéré comme "point de collecte", appelé station de base. Cette dernière peut être connectée à une machine puissante via internet ou par satellite. En outre, l'usager peut adresser ses requêtes aux capteurs en précisant l'information d'intérêt.

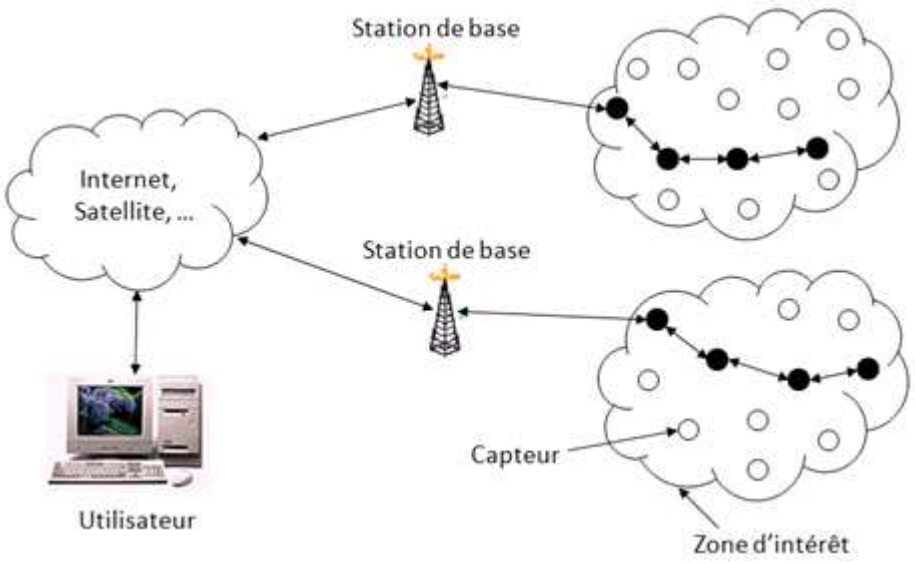


FIG. 1.1 – Exemple de réseaux de capteurs

Un exemple de réseaux de capteurs est fourni en figure 1.1 : les capteurs sont déployés de manière aléatoire dans une zone d'intérêt, et une station de base située à l'extrémité de cette zone, est chargée de récupérer les données collectées par les capteurs. Lorsqu'un capteur détecte un événement pertinent, un message d'alerte est envoyé à la station de base par le biais d'une communication multi-sauts. Les données collectées sont traitées et analysées par des machines puissantes.

Les réseaux de capteurs sont étroitement contraints en termes d'énergie, mémoire, capacité de traitement, et débit réalisable. Ils sont aussi contraints par une bande passante réduite et une latence élevée due à la nature du canal radio partagé. Le canal de communication radio est moins fiable qu'un médium filaire. Les capteurs peuvent aussi être mobiles, ce qui nécessite des algorithmes adaptatifs au changement de la topologie réseau. Néanmoins, le vrai défi critique dans ce type de réseaux est l'énergie car les capteurs sont dotés souvent de batteries non rechargeables. Ainsi, l'objectif principal dans ces réseaux est de minimiser la consommation d'énergie tout en assurant que le réseau effectue sa tâche dans des meilleures conditions. Par conséquent, une gestion de ressource rigoureuse en terme d'énergie sera exigée.

En outre, le déploiement d'un réseau de capteurs exige la fidélité d'acquisition (sensing fidelity) c'est-à-dire que l'occurrence d'un événement pertinent doit être détectée par au moins un capteur et la fidélité de routage (routing fidelity) c'est-à-dire qu'il doit exister au moins un

chemin entre le capteur qui a détecté l'événement et la station de base.

1.2 Qu'est-ce qu'un capteur ?

Un capteur sans fil est un petit dispositif electronique capable de mesurer une valeur physique environnementale (température, lumière, pression, etc.), et de la communiquer à un centre de contrôle via une station de base. Il est composé de quatre unités de base [3] (figure 1.2) :

- L'unité d'acquisition : est généralement composée de deux sous-unités : les capteurs et les convertisseurs analogique-numérique (ADCs). Les capteurs obtiennent des mesures numériques sur les paramètres environnementaux et les transforment en signaux analogiques. Les ADCs convertissent ces signaux analogiques en signaux numériques.
- L'unité de traitement : est composée de deux interfaces : une interface avec l'unité d'acquisition et une autre avec le module de transmission. Elle contrôle les procédures permettant au nœud de collaborer avec les autres nœuds pour réaliser les tâches d'acquisition, et stocke les données collectées.
- Un module de communication (Transceiver) : il est responsable de toutes les communications via un support de communication radio qui relie le nœud au réseau.
- Batterie : alimente les unités citées précédemment.
- Il existe des capteurs qui sont dotés d'autres composants additionnels : les systèmes de localisation tels que GPS (Global Position System) et un mobilisateur lui permettant le déplacement.

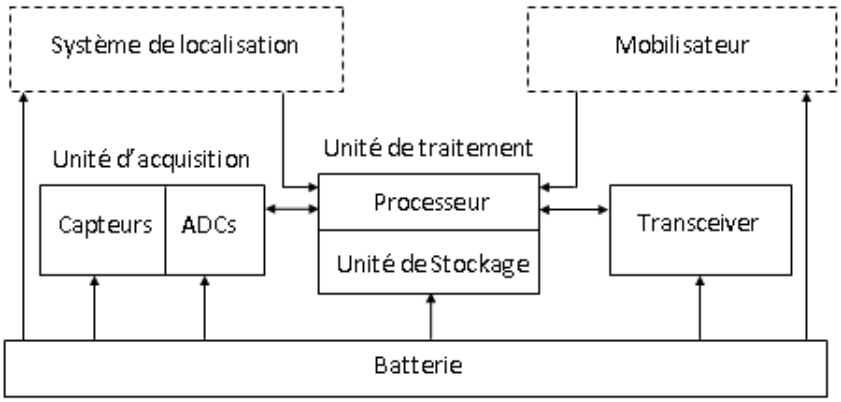


FIG. 1.2 – Les composants d'un nœud capteur

1.3 Domaines d'applications des réseaux de capteurs

La miniaturisation, l'adaptabilité, le faible coût et la communication sans fil permettent aux réseaux de capteurs d'envahir plusieurs domaines d'applications. Ils permettent aussi d'étendre le domaine des applications existantes. Parmi ces domaines où ces réseaux se révélent très utiles et peuvent offrir de meilleures contributions, on peut noter le militaire, la santé, l'environnemental, et les maisons intelligentes....

1.3.1 Applications militaires

Le faible coût, le déploiement rapide, l'auto-organisation et la tolérance aux pannes sont des caractéristiques qui ont rendu les réseaux de capteurs efficaces pour les applications militaires. Plusieurs projets ont été lancés pour aider les unités militaires dans un champ de bataille et protéger les villes contre des attaques, telles que les menaces terroristes. Le projet DSN (Distributed Sensor Network) [32] au DARPA (Defense Advanced Research Projects Agency) était l'un des premiers projets dans les années 80 ayant utilisés les réseaux de capteurs pour rassembler des données distribuées. Les chercheurs du laboratoire national Lawrence Livermore ont mis en place le réseau WATS (Wide Area Tracking System) [44]. Ce réseau est composé de détecteurs des rayons gamma et des neutrons pour détecter et dépister les dispositifs nucléaires. Il est capable d'effectuer la surveillance constante d'une zone d'intérêt. Il utilise des techniques d'agrégation de données pour les rapporter à un centre intelligent. Ces chercheurs ont mis en place ensuite un autre réseau appelé JBREWS (Joint Biological Remote Early Warning System) [17] pour avertir les troupes dans le champ de bataille des attaques biologiques possibles. Un réseau de capteurs peut être déployé dans un endroit stratégique ou hostile, afin de surveiller les mouvements des forces ennemies, ou analyser le terrain avant d'y envoyer des troupes (détection des armes chimiques, biologiques ou radiations). L'armée américaine a réalisé des tests dans le désert de Californie.

1.3.2 Applications à la surveillance

L'application des réseaux de capteurs dans le domaine de la sécurité peut diminuer considérablement les dépenses financières consacrées à la sécurisation des lieux et des êtres humains. Ainsi, l'intégration des capteurs dans de grandes structures telles que les ponts ou les bâtiments aidera à détecter les fissures et les altérations dans la structure suite à un séisme ou au vieillissement de la structure. Le déploiement d'un réseau de capteurs de mouvement peut constituer un

système d'alarme qui servira à détecter les intrusions dans une zone de surveillance.

1.3.3 Applications environnementales

Le contrôle des paramètres environnementaux par les réseaux de capteurs peut donner naissance à plusieurs applications. Par exemple, le déploiement des thermo-capteurs dans une forêt peut aider à détecter un éventuel début de feu et par suite faciliter la lutte contre les feux de forêt avant leur propagation. Le déploiement des capteurs chimiques dans les milieux urbains peut aider à détecter la pollution et analyser la qualité d'air. De même leur déploiement dans les sites industriels empêche les risques industriels tels que la fuite de produits toxiques (gaz, produits chimiques, éléments radioactifs, pétrole, etc.).

Dans le domaine de l'agriculture, les capteurs peuvent être utilisés pour réagir convenablement aux changements climatiques par exemple le processus d'irrigation lors de la détection de zones sèches dans un champ agricole. Cette expérimentation a été réalisée par Intel Research Laboratory and Agriculture and Agri-Food Canada sur une vigne à British Columbia.

1.3.4 Applications médicales

Dans le domaine de la médecine, les réseaux de capteurs peuvent être utilisés pour assurer une surveillance permanente des organes vitaux de l'être humain grâce à des micro-capteurs qui pourront être avalés ou implantés sous la peau (surveillance de la glycémie, détection de cancers, ..). Ils peuvent aussi faciliter le diagnostic de quelques maladies en effectuant des mesures physiologiques telles que : la tension artérielle, battements du cœur, ... à l'aide des capteurs ayant chacun une tâche bien particulière. Les données physiologiques collectées par les capteurs peuvent être stockées pendant une longue durée pour le suivi d'un patient [57]. D'autre part, ces réseaux peuvent détecter des comportements anormaux (chute d'un lit, choc, cri, ...) chez les personnes dépendantes (handicapées ou âgées).

1.3.5 La domotique

Avec le développement technologique, les capteurs peuvent être embarqués dans des appareils, tels que les aspirateurs, les fours à micro-ondes, les réfrigérateurs, les magnétoscopes, ... [85]. Ces capteurs embarqués peuvent interagir entre eux et avec un réseau externe via Internet pour permettre à un utilisateur de contrôler les appareils domestiques localement ou à distance.

Le déploiement des capteurs de mouvement et de température dans les futures maisons dites intelligentes permet d'automatiser plusieurs opérations domestiques telles que : la lumière s'éteint et la musique se met en état d'arrêt quand la chambre est vide, la climatisation et le chauffage s'ajustent selon les points multiples de mesure, le déclenchement d'une alarme par le capteur anti-intrusion quand un intrus veut accéder à la maison.

1.4 Contraintes et caractéristiques liées aux réseaux de capteurs

La réalisation des réseaux de capteurs dédiés aux applications citées dans le paragraphe précédent, exigent les techniques et les protocoles qui prennent en compte les spécificités et les exigences de ces réseaux, vu que les techniques conçues pour les réseaux ad hoc traditionnels, ne sont pas bien adaptées aux réseaux de capteurs. Pour illustrer ce point, les différences entre les réseaux de capteurs et les réseaux ad hoc sont décrits ci-dessous [3] :

- Dans les réseaux de capteurs, les nœuds sont déployés en grand nombre,
- Les réseaux de capteurs sont non fiables : en tout moment, les capteurs peuvent être défaillants ou inhibés.
- La topologie des réseaux de capteurs change très fréquemment.
- Les réseaux de capteurs utilisent principalement le paradigme de communication broadcast tandis que les réseaux ad hoc sont basés sur le paradigme de communication point-à-point.
- Les capteurs sont limités en énergie, capacités de calcul, et mémoire.

Le contexte de développement des réseaux de capteurs prend en compte leurs caractéristiques et leurs spécificités.

1.4.1 Modèle énergétique

Dans les réseaux ad hoc, la consommation de l'énergie a été considérée comme un facteur déterminant mais pas primordial parce que les ressources d'énergie peuvent être remplacées par l'utilisateur. Ces réseaux se focalisent plus sur la QoS que sur la consommation de l'énergie. Par contre, dans les réseaux de capteurs, la consommation d'énergie est une métrique de performance très importante puisque généralement les capteurs sont déployés dans des zones inaccessibles.

Ainsi, il est difficile voire impossible de remplacer les batteries après leur épuisement. De ce fait, la consommation d'énergie au niveau des capteurs a une grande influence sur la durée de vie du réseau.

La tâche principale d'un capteur dans une zone d'intérêt est de détecter des événements, calculer et agréger les données collectées, puis les transmettre à la station de base. La consommation de l'énergie peut concerner trois opérations : acquisition, communication, calcul et agrégation des données. La consommation de l'énergie par l'opération d'acquisition dépend de la nature de l'application. Par exemple, une détection sporadique pourrait consommer moins d'énergie qu'une surveillance permanente d'un événement. La communication de données, en particulier dans la transmission de données, consomme plus d'énergie que les autres opérations. Par conséquent, la consommation d'énergie dans un réseau de capteurs est en fonction d'émissions : plus le nombre de nœuds réémettant le message est faible plus la technique de diffusion est efficace. En outre, elle dépend aussi de la distance séparant l'émetteur du récepteur. En effet, un faible nombre d'émetteurs à grande portée peut fournir de moins bons résultats qu'un grand nombre d'émetteurs à faible portée.

Puisque la consommation d'énergie d'un nœud *u* est fonction de la portée de communication $r(u)$ alors sa valeur $C_{Energie}(u)$ d'après le modèle le plus largement utilisé est :

$$C_{Energie}(u) = k.r(u)^{\alpha} \qquad \text{avec} \quad \alpha \geq 2 \tag{1.1}$$

où *k* est la taille du message et α le coefficient d'atténuation du signal.

Si on suppose que la taille des messages et la portée de communication sont fixes, alors plus la valeur de α est élevée plus l'énergie nécessaire pour couvrir une distance donnée est élevée. Si on ajoute à cette consommation les dépenses énergétiques des diverses opérations impliquées dans une émission telles que : l'opération de préparation du message, de traitement par la couche MAC, ...[41], alors le modèle énergétique réel devient :

$$C_{Energie}(u) = \begin{cases} k \times r(u)^{\alpha} + C_e & \text{si } r(u) \neq 0 \\ 0 & \text{sinon} \end{cases} \tag{1.2}$$

où les valeurs α et C_e sont exprimées en unités arbitraires [91] : $\alpha = 4$ et $C_e = 10^8$.

Une solution intuitive pour économiser l'énergie est de maintenir des capteurs en état de sommeil jusqu'à ce qu'ils commencent une tâche spécifique. Le réveil des capteurs est fait à une

certaine heure définie ou déclenchée par un événement externe. Ainsi, les capteurs peuvent alors communiquer les informations recueillies à leurs voisins pour les envoyer à la station de base. Néanmoins, cette alternative peut s'avérer inefficace si nous prenons en considération le temps et la consommation d'énergie liés au changement d'état "sommeil/actif" des capteurs.

1.4.2 Lien radio

En plus de la technologie des systèmes micro-électro-mécanique, le développement de WSNs se base également sur les technologies sans fil. Le protocole 802.11 et ses variants, bien qu'ils soient conçus pour les LANs sans fil qui se composent habituellement d'ordinateurs portables et de PDAs, sont également supposés bénéfiques aux réseaux de capteurs. Cependant, la consommation d'énergie élevée et le débit excessif rendent les protocoles 802.11 non appropriés aux réseaux de capteurs. Ce fait a motivé les chercheurs à concevoir des protocoles MAC (Medium Access Control) efficaces en énergie [38, 78, 98, 104, 116]. Récemment, le protocole 802.15.4 [127] basé sur ZigBee a été réalisé. Il a été conçu spécifiquement pour les réseaux locaux personnels sans fil (LR_WPAN). Ses performances en termes de consommation d'énergie ont poussé plusieurs industriels de capteurs, y compris MicaZ [124], Telos [87], ... à l'intégrer dans les capteurs.

Au dessus de la couche physique et la couche MAC, les techniques de routage dans les réseaux sans fil sont une autre direction de recherche pour les réseaux de capteurs. Les premiers protocoles de routage dans les réseaux de capteurs sont réellement des protocoles de routage pour les réseaux ad hoc. Ces protocoles, y compris [35, 54], s'appliquent rarement aux réseaux de capteurs à cause de leur consommation d'énergie élevée. Ils sont aussi conçus pour supporter les demandes d'établissement de routes dans les réseaux sans fil, sans considérer les modèles de communication spécifiques dans les réseaux de capteurs. Ainsi, l'adaptation et la personnalisation de ces protocoles et le développement de nouvelles techniques de routage sont devenus des domaines de recherche très attractifs. L'objectif principal de ces recherches est de permettre le routage à moindre coût en énergie tout en étant robuste en exploitant plusieurs liens et routes.

1.4.3 La couche MAC (Medium Access Control)

Dans la plupart des plates-formes matérielles, la transmission radio est la source principale de consommation d'énergie et l'activité radio est en grande partie commandée par la couche MAC. En ce qui concerne le protocole MAC, les sources principales de perte d'énergie sont les

collisions, l'écoute (idle listening) pour recevoir des données possibles, overhearing i.e. recevant des données destinées à d'autres nœuds, le contrôle de l'overhead et over-emitting i.e. la transmission d'un message quand le destinataire n'est pas prêt. Donc, pour économiser au mieux l'énergie de la batterie d'un capteur, il faudrait que les transmetteurs radio soient éteints le plus possible. Cependant, ceci pourrait poser le problème de la synchronisation des capteurs et la répartition des périodes de réveil. Ainsi, il faudrait que la couche MAC permette aux capteurs d'avoir des périodes de sommeil assez longues, mais sans perturber les communications.

Le protocole MAC dédié aux réseaux de capteurs devrait être efficace en terme d'énergie, stable lorsque la taille du réseau augmente, et adaptatif aux changements de la topologie et de la connectivité du réseau lorsque les capteurs cessent de fonctionner, ou de se déplacer. Nous allons distinguer deux catégories de protocoles au niveau de la couche MAC.

Les protocoles ordonnancés (Scheduling protocols)

Ces protocoles sont efficaces en consommation d'énergie parce qu'ils évitent les collisions et l'overhearing. Ils ne permettent pas les communications pair-à-pair et exigent généralement aux nœuds de former des clusters. La communication inter-cluster est réalisée par les approches : TDMA, FDMA, et CDMA. L'utilisation de FDMA pour les communications intra-cluster évite les collisions, mais elle augmente la consommation d'énergie des capteurs à cause du besoin de circuits additionnels pour communiquer dynamiquement avec les différents canaux radio. De même, CDMA évite les collisions mais les calculs consomment suffisamment de l'énergie [36], alors que TDMA essaie d'optimiser le procédé d'attribution des slots dans le but de minimiser la consommation de l'énergie. Cependant, ces approches manquent la scalabilité et l'adaptabilité aux changements de la topologie ainsi que l'ordonnancement d'activité des nœuds relais. En outre, ils dépendent d'une synchronisation distribuée et leur débit est limité à cause des slots d'écoute non utilisés.

Pour remédier aux limitations présentées dans ces approches, d'autres techniques ont été proposées dans cette direction. Sohrabi et Pottie [99] ont proposé un protocole MAC appelé SMACS (Self-organizing Medium Access Control for Sensor Networks) pour organiser le réseau. SMACS implique une méthode d'accès au médium combinant TDMA et FDMA, dans laquelle les nœuds voisins choisissent aléatoirement un slot et une fréquence qui définit un lien. Sohrabi et al. [98] ont proposé un autre protocole appelé EAR (Eavesdrop And Register). Ce protocole intègre les nœuds mobiles dans ce mécanisme en introduisant une table des voisins stockée au niveau de chaque nœud et un autre message d'échange pour mettre à jour la topo-

logie. Ces deux protocoles sont très utiles quand la transmission périodique de l'information exige la maintenance continuelle du réseau. Cependant, ils exigent un overhead important pour construire le backbone, et la bande passante n'est pas bien exploitée. En outre, Arisha et al. [11] ont présenté un protocole MAC basé sur TDMA, qui se sert des passerelles comme cluster-heads pour assigner des slots aux capteurs dans un cluster. Néanmoins, le passage de l'état "ON/OFF" est très coûteux en consommation d'énergie.

Protocoles basés sur la contention

Les protocoles d'accès au médium basés sur la contention, sont plus flexibles aux changements de la topologie. Ils permettent la communication pair-à-pair, et n'exigent aucune synchronisation. Néanmoins, ils n'utilisent pas souvent efficacement les ressources à cause des collisions et de l'écoute inutile.

S-MAC (Sensor MAC) [115] est un protocole similaire au protocole 802.11. Il utilise la méthode d'accès au médium CSMA/CA RTS/CTS (Request-To-Send, Clear-To-Send) qui permet d'éviter les collisions et le problème du nœud caché. Ce protocole instaure un mécanisme de mise en veille distribué à chaque nœud dans le but de réduire sa consommation d'énergie et prolonger sa durée de vie. Chaque nœud devrait coordonner et échanger des informations avec ses voisins pour choisir son propre cycle "Sommeil/Réveil". Néanmoins, l'utilisation intensive du cycle sommeil augmente considérablement la latence, car il est difficile de synchroniser les nœuds de telle sorte que la communication soit toujours possible. Pour réaliser cette synchronisation, S-MAC permet à chacun des nœuds d'envoyer un paquet SYNC générant un ordonnancement à tous ses voisins. S-MAC est particulièrement conçu pour les systèmes d'alerte dans lesquels les applications ont de longues périodes d'inactivité et peuvent tolérer la latence. Donc, son but est de maximiser la durée de vie du réseau en dépit des autres critères de performances. Néanmoins, les concepteurs de S-MAC ont négligé le coût énergétique du cycle "Réveil".

Le protocole T-MAC (Timeout MAC) [104] a été conçu pour améliorer les performances de S-MAC avec une charge de trafic variable. Dans T-MAC, chaque nœud se réveille périodiquement pour communiquer avec ses voisins en utilisant le mécanisme RTS/CTS. Ainsi, le nœud écoute le canal radio et il ne peut transmettre que s'il est dans la période active. Cette période se termine si au bout d'un certain temps TA aucun événement d'activation ne se produit tel que le déclenchement d'un temporisateur de frame, la réception des données sur le canal, la détection d'une communication sur le canal,... En outre, la synchronisation d'un nœud avec ses voisins est réalisée comme suit : quand un nœud se réveille, il écoute le canal. S'il n'entend rien au bout

d'un certain temps, il choisit un ordonnancement et le transmet via un paquet SYNC, alors que s'il reçoit un paquet SYNC une fois qu'il se réveille, il suivra l'ordonnancement défini dans ce paquet et transmet par la suite son propre paquet SYNC. Un nœud pourrait recevoir deux ordonnancements à la fois de deux voisins qui ne sont pas voisins entre eux, et dans ce cas, il alertera le premier nœud auprès duquel il a reçu un ordonnancement. La présence de deux ordonnancements permettra au nœud de s'activer à la fin de chacune des deux frames. Un nœud ne devra commencer une transmission de données qu'au début de sa période active. Par suite, tous ses voisins se réveilleront suite à cet ordonnancement et seront prêts à recevoir le message. Cependant, les performances de T-MAC se dégradent quand le trafic est unidirectionnel, par exemple quand seulement les capteurs remontent leurs données collectées à la station de base.

DSMAC (S-MAC dynamique) [76] est une autre variante de S-MAC. Ce protocole ajoute un cycle dynamique d'activité à S-MAC, qui réduit la latence dans les applications critiques. Par exemple, un nœud peut doubler son cycle d'activité si le niveau de sa batterie est au-dessus d'un certain seuil. DSMAC génère une latence inférieure à celle de S-MAC avec une meilleure moyenne de consommation d'énergie par paquet.

B-MAC (Berkeley MAC) [86] a été développé par l'université de Berkeley pour les capteurs compatibles ZigBee. Il se base sur deux principes : l'analyse du bruit sur le canal et l'écoute basse consommation d'énergie. Ainsi, un nœud voulant transmettre des données devrait écouter le canal. Si le canal est libre, il devra émettre au début un préambule. Les nœuds sont souvent dans l'état "sommeil" et ils se réveilleront à des intervalles réguliers. A leur réveil, ils écouteront le canal s'il y a du bruit, cela signifie qu'il y a des données qui vont arriver sinon ils retourneront à l'état sommeil. Le principal avantage du B-MAC est qu'il ne nécessite pas de synchronisation entre les nœuds. Cependant, l'utilisation de préambule avant tout envoi pourrait avoir des conséquences sur la consommation d'énergie.

Le standard LRWPAN (Low Rate Wireless Personal Area Networks) utilise deux méthodes d'accès au médium CSMA/CA et slotted CSMA/CA. CSMA/CA est similaire à celle utilisée par les protocoles 802.11 mais avec un petit backoff, alors que slotted CSMA/CA (accès multiple durant un intervalle de temps) utilise des supertrames qui sont définies par le coordinateur, et qui ont pour objectif de permettre la synchronisation de l'ensemble des nœuds entre eux et économise la consommation d'énergie de chaque nœud. La synchronisation entre les nœuds est réalisée grâce à l'envoi d'un beacon. Cet envoi est suivi par une zone appelée Contention Access Period (CAP), qui permet à chaque capteur d'envoyer ou recevoir des trames de commandes

et/ou de données suivant le mécanisme CSMA/CA. Cependant, CSMA/CA induit une consommation d'énergie importante surtout durant les périodes de fort trafic, ce qui a conduit à réduire la valeur du backoff et par conséquent réduire la période de contention pendant laquelle le nœud émetteur devrait écouter le canal.

1.4.4 La mobilité

La mobilité est une question clé pour les réseaux de capteurs. Par exemple, quand les capteurs sont embarqués sur des dispositifs mobiles tels que les véhicules, ou sur des animaux. Lorsque la mobilité est trop fréquente, elle ne peut être considérée comme un problème secondaire. Ainsi, la détection des voisins et la reconfiguration du réseau exige habituellement un nombre important de messages de contrôle de la topologie, donc une dépense importante d'énergie. En outre, un autre type de mobilité pourrait être pris en compte, qui est la mobilité de la station de base ou les deux types de dispositifs peuvent être mobiles simultanément.

Dans notre contexte, nous considérons un type particulier de réseaux de capteurs, les réseaux de capteurs mobiles dans lesquels seulement la mobilité des capteurs est prise en considération.

1.5 Catégories de communication

Le comportement des réseaux de capteurs est vraiment différent que celui des réseaux IP. Il n'y a pas généralement de communication capteur-à-capteur (pair-à-pair). Au lieu de cela, les capteurs envoient les mesures directement ou en mode multi-sauts à une ou plusieurs stations de base.

Dans les réseaux de capteurs, les contraintes imposées aux protocoles de communication dépendent de la nature des capteurs ainsi que de l'application qui en est faite. Toutefois, la contrainte énergétique est considérée comme une contrainte plus forte, puisque l'on considère que les capteurs sont généralement déployés dans des zones d'intérêt hostiles et par suite le rechargement de leurs batteries est presque impossible.

La manière d'acheminer les messages dans les réseaux de capteurs peut changer selon l'application qui en est faite. Ainsi, dans les scénarios impliquant la collecte de données périodiques, les capteurs transmettent régulièrement leurs mesures à la station de base. Ce processus peut être continu ou pourrait suivre une certaine distribution (géométrique, gaussienne, ...), déterministe

ou probabiliste. La connaissance du processus d'envoi peut également influencer les périodes de sommeil des capteurs non impliqués dans l'envoi de données. Ce comportement est typique pour effectuer des mesures statistiques d'une certaine métrique, par exemple l'étude du climat. En outre, dans les scénarios orientés événement (event-driven), les capteurs doivent seulement transmettre une alerte quand un événement pertinent survient ou quand un changement brusque se produit. Dans ce type de scénarios, la réception de message doit être assurée et le délai de transmission doit être limité. Néanmoins, certains mécanismes d'alerte peuvent être désactivés à cause de la défaillance d'un lien radio. Par conséquent, il est recommandé d'envoyer l'alerte plusieurs fois.

1.6 Architectures adoptées pour les réseaux de capteurs

Les architectures dans les réseaux de capteurs dépendent des applications et des techniques utilisées pour faire acheminer l'information des capteurs à la station de base. Une taxonomie des applications peut être dérivée et l'adaptabilité d'algorithmes à ce genre de scénario peut être évaluée.

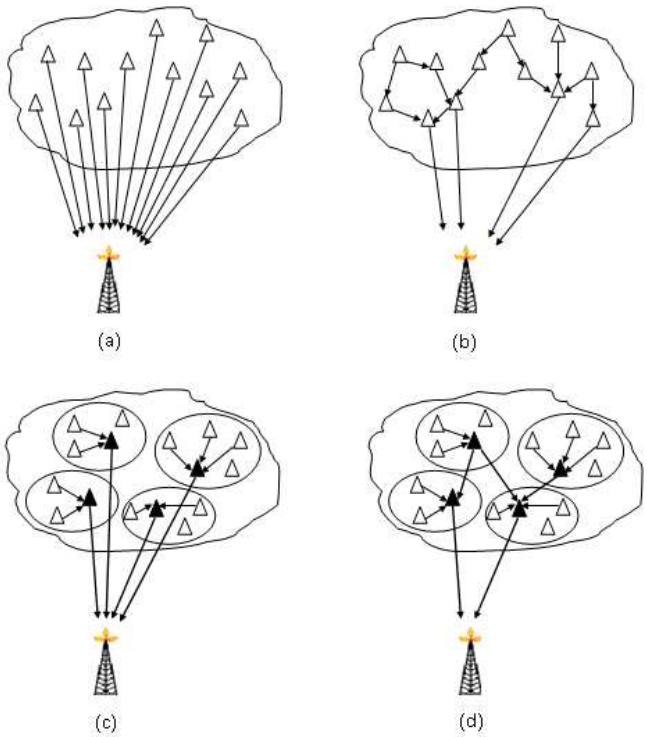


FIG. 1.3 – Architecture de communication de données dans un réseau de capteurs

Le processus d'acheminement de l'information des capteurs à la station de base peut prendre quatre formes. Dans les architectures à plat, les capteurs peuvent communiquer directement avec la station de base en utilisant une forte puissance (figure 1.3 (a)), ou via un mode multi-sauts avec des puissances très faibles (figure 1.3 (b)), alors que dans les architectures hiérarchisées, le nœud représentant le cluster, appelé cluster-head, transmet directement les données à la station de base (figure 1.3 (c)), ou via un mode multi-saut entre les cluster-heads (figure 1.3 (d)).

1.7 Les défis des réseaux de capteurs

La conception de protocoles et de techniques pour les réseaux de capteurs est influencée par plusieurs contraintes :

• Lien radio : les capteurs possèdent le matériel nécessaire pour effectuer des communications par ondes radio. Toutefois, la diffusion de l'information via ces liens est peu aisée à cause de l'instabilité et du manque de fiabilité qu'ils présentent. En plus, l'utilisation d'un médium de communication partagé pour faire face aux interférences radio, réduit considérablement la capacité d'exploitation du canal.

• La dynamicité : la topologie des réseaux de capteurs sans fil mobiles est en constante évolution à cause de l'état d'activité des capteurs (extinction, mise en veille et actif) ainsi que leurs déplacements indépendamment les uns des autres. La conception d'un protocole d'auto-organisation qui s'adapte continuellement et rapidement aux changements, s'avère nécessaire pour assurer le bon fonctionnement du réseau.

• La limitation des ressources : les capteurs se caractérisent par une limitation de ressources en termes de calcul, de stockage et d'autonomie d'énergie. La contrainte énergie dans ce type de réseaux étant très forte, puisque l'on considère généralement qu'il est difficile voire impossible de recharger les batteries des capteurs (capteurs déployés dans des zones hostiles et inaccessibles). L'ordonnancement d'activité et l'auto-organisation sont considérés comme des solutions de premier choix, dans le but d'éteindre l'équipement radio de certains capteurs et permettre seulement aux capteurs disposant de plus d'énergie de transmettre les données captées à la station de base.

• Localisation des décisions : un réseau de capteurs est potentiellement constitué d'un grand nombre de capteurs qui ont une portée de transmission très limitée. D'où, le processus d'acheminement d'une information de la source à la destination se fait généralement en multi-sauts par le biais de plusieurs capteurs. En outre, il est inconcevable de faire impliquer un grand nombre de

capteurs pour une prise de décision d'un capteur. Ainsi, tout processus de prise de décision doit être localisé.

• Passage à l'échelle : la plupart des protocoles sont conçus pour des réseaux de capteurs d'une grande taille. Cependant, ces protocoles sont dits efficaces si les performances des réseaux ne doivent pas chuter d'une manière drastique quand le nombre de capteurs augmente dans le réseau.

• Auto-configuration : les réseaux de capteurs sont généralement déployés aléatoirement dans des zones d'intérêt hostiles (capteurs largués par un avion). Par conséquent, aucune intervention humaine ne peut être requise pour assurer leur organisation. L'auto-configuration de ces réseaux s'avère nécessaire pour leur bon fonctionnement.

• Tolérance aux fautes : certains capteurs pourraient être défaillants, bloqués à cause de l'épuisement de leurs batteries, ou subissent des dommages physiques (écrasés par des animaux ou lorsqu'ils sont jetés par un avion). La défaillance de ces capteurs ne devrait pas avoir une influence sur le fonctionnement du réseau. La tolérance aux fautes est la capacité de soutenir les fonctionnalités d'un réseau de capteurs sans causer d'interruption lorsqu'un capteur cesse de fonctionner [49, 58]. Les protocoles et les techniques peuvent être conçus pour évaluer le niveau de la tolérance aux fautes exigée par les réseaux de capteurs. Si les capteurs sont déployés dans un habitat, la tolérance aux fautes exigée peut être basse puisque ce type de réseaux n'est pas facilement endommagé, alors que si les capteurs sont déployés dans un champ de bataille pour la surveillance et la détection, la tolérance aux fautes devrait être élevée puisque les données surveillées sont critiques et les capteurs peuvent être détruits facilement par des actions hostiles.

1.8 Différentes problématiques dans les réseaux de capteurs

Les recherches dans le domaine des réseaux de capteurs ont révélé plusieurs problématiques, parmi ces problématiques, nous citons :

• Routage : concevoir un protocole de routage performant en termes de minimisation de la consommation de l'énergie, du choix des routes optimales pour l'acheminement de l'information d'un capteur à la station de base et vice versa, de réduction du délai de délivrance des paquets...Ainsi le réseau doit passer à l'échelle sans que ses performances se dégradent.

• Couche MAC : la spécificité des réseaux de capteurs sans fil mobiles nécessite le développement de nouveaux protocoles MAC qui s'adaptent aux contraintes imposées par ces réseaux.

Ceci dans le but d'améliorer le débit, minimiser la consommation d'énergie, optimiser le partage du médium ainsi que minimiser le délai de délivrance des paquets.

• Qualité de service : des protocoles au niveau de la couche MAC devraient être capables d'établir des priorités entre les flux, limiter les pertes de paquets vitaux pour la gestion du réseau, ou du moins en restreindre l'impact.

• Cross-layer : dans les modèles classiques, les concepteurs essaient d'optimiser les performances au niveau d'une couche indépendamment des autres couches. Par exemple, le routage et les fonctions de la couche MAC sont optimisés indépendamment les uns des autres. Cependant, une telle indépendance est communément considérée comme trop onéreuse pour les réseaux de capteurs. Par conséquent, une coopération permettant un compromis entre performances, dépendance et flexibilité doit être proposée pour optimiser les capacités du réseau en général.

• Diffusion de l'information : les protocoles de diffusion conçus pour les réseaux de capteurs doivent tenir compte de leurs spécificités ainsi que de leurs contraintes intrinsèques imposées. Ainsi, pour concevoir un protocole efficace, il faudrait assurer une couverture maximale des capteurs composant le réseau (taux d'accessibilité supérieur 90%), minimiser le nombre des réémetteurs et des réceptions redondantes ainsi que la consommation d'énergie.

• Sécurité : pour les applications qui exigent un niveau de sécurité assez élevé telles que les applications militaires, des mécanismes d'authentification, de confidentialité, et d'intégrité doivent être mis en place au sein de leur communauté. Les algorithmes de cryptographie conçus pour les réseaux de capteurs doivent tenir compte des ressources limitées que présentent ces réseaux.

1.9 Motivations et objectifs

La plupart des solutions proposées pour la diffusion et la couverture de zone dans les réseaux de capteurs s'appuient sur une vue à plat où tous les nœuds sont considérés comme égaux et doivent contribuer ensemble à la gestion du réseau pour qu'ils puissent accomplir ses tâches. Par ailleurs, les capteurs peuvent apparaître, disparaître (par extinction de leur radio, épuisement de leur énergie ou leur crash), ou se déplacer indépendamment les uns des autres. Ainsi, la topologie du réseau est en constante évolution. De ce fait, l'acheminement de l'information d'un capteur à une station de base distante et vice versa doit se faire par un protocole efficace en énergie. Afin que ce protocole soit efficace, il devra prendre en considération les contraintes intrinsèques du réseau (topologie en constante évolution), les ressources limitées des capteurs en termes de

calcul, de stockage et d'énergie, ainsi que le mode de gestion du médium de communication partagé (bande passante limitée, collisions, ...).

L'une des solutions communément proposées pour la diffusion et la couverture de zone dans les réseaux de capteurs sans fil est d'utiliser une architecture hiérarchisée appelée encore clustering. Cette architecture permet de regrouper les capteurs proches géographiquement en clusters et d'utiliser des schémas de routage différents au sein des clusters et entre les clusters. Ainsi un tel capteur au sein d'un cluster stocke la totalité des informations des capteurs qui font partie de son cluster et seulement une partie des informations qui concernent les autres clusters. Dans cette architecture, seulement les cluster-heads et les nœuds passerelles sont responsables du cheminement de l'information captée par un capteur à la station de base.

Dans cette thèse, nous proposons deux protocoles de diffusion et un protocole de couverture de zone. Le premier est un protocole pour la diffusion dans un environnement réaliste (modèle Lognormal) basé sur l'approche MPR, et le deuxième est un protocole d'auto-organisation basé sur le clustering pour la diffusion dans les réseaux de capteurs. Ce dernier organise les capteurs en 2-clusters homogènes en taille et confie aux cluster-heads et aux nœuds passerelles la responsabilité de transmettre les informations captées par les capteurs à la station de base. Le troisième est un protocole de couverture de zone basé sur l'ordonnancement d'activité de capteurs. Il sélectionne les capteurs actifs en fonction de leurs capacités en terme d'énergie et de structure topologique pour assurer la couverture totale de la zone d'intérêt pour une longue durée.

Chapitre 2

Algorithmes de diffusion dans les réseaux ad hoc et de capteurs

2.1 Introduction

Dans les réseaux de capteurs, les problèmes sont principalement liés à l'autonomie résultant du coût des communications. Ces coûts sont importants lors de l'envoi des données collectées par un capteur à la station de base ou par l'envoi d'une requête à tous les capteurs par la station de base. Ainsi, le choix d'une technique de diffusion est alors primordial pour garantir une grande autonomie pour ces réseaux qui sont généralement déployés dans des zones hostiles. Cette technique doit être capable de diffuser et acheminer une information sans perdre trop d'énergie. De ce fait, pour le traitement du problème de la diffusion dans les réseaux ad hoc et de capteurs, plusieurs approches ont été décrites dans la littérature. Ces approches se basent essentiellement sur le clustering, l'ensemble dominant connecté (CDS) et la diffusion dépendant de la source. Cependant, les techniques d'acheminement de l'information dédiées particulièrement aux réseaux de capteurs doivent tenir compte de la spécificité de ces réseaux et du type de communications induit par l'application.

Dans les architectures à plat, la majorité des protocoles de routage et de diffusion conçus pour les réseaux de capteurs ou les réseaux ad hoc de petite ou moyenne taille avec une faible mobilité de nœuds fournissent de bonnes performances. Cependant, lorsque le nombre de nœuds augmente [93] ou que les nœuds sont mobiles [61], le trafic de contrôle prédomine les communications réelles. Ce qui conduit à une augmentation de la latence et à une explosion des tables de routage. Afin de pallier à ces limites, l'architecture hiérarchique s'est considérée comme l'une

des solutions communément efficaces pour la diffusion et le routage dans les réseaux ad hoc et de capteurs. Elle consiste à regrouper les nœuds proches géographiquement en groupes aussi appelés "clusters" et d'établir des schémas de routage différents à l'intérieur des clusters (intra-clusters) et entre les clusters (inter-clusters). Chaque cluster est représenté par un nœud particulier appelé cluster-head. Ce nœud est élu comme cluster-head selon une métrique spécifique ou une combinaison de métriques. Il est responsable de la coordination entre les différents membres de son cluster, pour agréger leurs données collectées et les transmettre vers la station de base. Dans un cluster, chaque nœud stocke la totalité des informations de son cluster et une partie des informations des autres clusters ; ce qui minimise considérablement la taille des tables de routage et le nombre de messages échangés dans le réseau.

En outre, l'approche de l'ensemble dominant connexe (CDS) a été utilisée pour optimiser les coûts des communications. Dans [110], la réduction de l'overhead de la diffusion consiste à réduire la taille et le coût de maintenance de tables de routage en enlevant tous les liens entre les nœuds qui ne font pas partie de backbone formé par l'ensemble dominant connecté, et en impliquant seulement les nœuds formant le backbone dans la diffusion. Ainsi, la redondance excessive d'émission pourrait être évitée. Dans [26], la performance de la diffusion est optimisée grâce à l'implication seulement des nœuds formant le backbone dans le maintien de la connectivité du réseau et la mise en mode sommeil des autres nœuds. En outre, dans [22], les auteurs ont impliqué la technique de l'ensemble dominant connexe pour assurer à la fois la couverture de zone et l'acheminement des informations des capteurs à la station de base dans les réseaux de capteurs déployés en grand nombre. Bien que les réseaux ad hoc et de capteurs ne disposent pas d'une infrastructure dorsale physique, un backbone virtuel pourrait être construit par les nœuds dans un ensemble dominant connexe (CDS). Un tel ensemble pouvant être utilisé pour la diffusion d'information. Ainsi plus il sera petit, plus il sera efficace car la redondance des transmissions et des réceptions de messages consomment plus d'énergie.

Dans ce chapitre, nous présentons et nous analysons ces approches ainsi que les protocoles sous-jacents à ces approches pour tirer profit de leurs avantages et d'éviter leurs limitations.

2.2 Critères de performances d'un protocole de diffusion

L'efficacité d'une technique de diffusion permet à un réseau de capteurs de mener sa mission jusqu'à la fin et pour une longue durée. Ainsi, on dit qu'un protocole de diffusion est efficace s'il répond aux propriétés suivantes :

- Extensibilité : il supporte le passage à l'échelle sans présenter de goulots d'étranglement.
- Accessibilité (Reachability ou RE) : cette propriété reflète le taux de nœuds recevant le message de diffusion par rapport au nombre total de nœuds accessibles à partir du nœud source. Un protocole est dit parfait si tous les nœuds sont accessibles à partir de la source (RE=100%). Si le taux d'accessibilité d'un protocole est inférieur à 90%, le protocole est considéré comme invalide. Notons qu'il existe des protocoles qui fonctionnent en meilleur effort (best effort), et qui ont un taux d'accessibilité de l'ordre de 90% et plus.
- Rediffusion économisée (Saved ReBroadcast ou SRB) : SRB représente le pourcentage de nœuds recevant le message de diffusion et ne le réémettant pas. Soit N_r le nombre de nœuds recevant le message de diffusion et N_t le nombre de nœuds qui le transmet, alors le taux de la rediffusion économisée est : $SRB = (N_r - N_t)/N_r$.
- Consommation énergétique : l'énergie est considérée comme une ressource précieuse dans les réseaux de capteurs, sa conservation est indispensable pour garantir une longue durée de vie aux réseaux puisqu'il est généralement impossible de recharger les batteries des capteurs. Cette consommation est optimisée quand le nombre de messages retransmis et le nombre de réceptions redondantes sont réduits.
- Consommation de la bande passante : la minimisation du nombre de messages redondants optimise la consommation de la bande passante.
- Latence : délai entre l'émission par la source et la dernière réception du message de diffusion. C'est un critère déterminant dans les réseaux de capteurs car il permet à l'usager d'intervenir rapidement à l'occurrence d'événement.

En particulier, dans les réseaux de capteurs, il faudrait garantir la fidélité du routage (routing fidelity) c'est-à-dire il devrait exister au moins un chemin entre tout capteur du réseau déployé et la station de base. En outre, il faudrait assurer la fidélité de surveillance (sensing fidelity) c'est-à-dire il existe au moins un capteur qui détecte un événement sur la zone d'intérêt.

2.3 Techniques de clustering

L'approche de clustering consiste à partitionner le réseau en un certain nombre de clusters, plus homogènes selon une métrique spécifique ou une combinaison de métriques, et former une topologie virtuelle. Les clusters sont généralement identifiés par un nœud particulier appelé cluster-head. Ce dernier permet de coordonner entre les membres de son cluster, d'agréger leurs

données collectées et de les transmettre à la station de base. Il est sélectionné pour jouer ce rôle selon une métrique bien particulière ou une combinaison de métriques.

Avant de présenter les différents algorithmes basés sur le clustering, il est nécessaire de présenter les concepts liés à cette approche.

2.3.1 Définition

Soit un réseau sans fil modélisé par un graphe connexe non orienté $G = (V, E)$ où les sommets V sont les nœuds et les arêtes E représentent les liens de communication. Le processus de clustering consiste à un découpage virtuel de V en un ensemble de groupes proches géographiquement {V_1,V_2,...,V_k} tel que :

$$V = \bigcup_{i=1}^{k} V_i \tag{2.1}$$

où chaque sous-ensemble V_i induit un sous-graphe connexe de G ou une composante connexe du graphe G. L'ensemble de ces composantes peuvent former un graphe réduit ou (graphe de composantes) $G^{'} = (V^{'}, E^{'})$ de G, où :

- les sommets $v_i^{'} \in V^{'}$ correspondent aux composantes connexes V_i de G ;
- $E^{'}$ contient l'arête $(v_i^{'}, v_j^{'})$ si et seulement si il existe dans le graphe G une arête d'un sommet $u_i \in V_i$ vers un sommet $u_j \in V_j$.

Ces groupes sont appelés "Clusters" et ils ne sont pas nécessairement disjoints. Chaque cluster est identifié par un nœud particulier appelé cluster-head "CH". Le choix du cluster-head se fait sur la base d'une métrique spécifique ou d'une combinaison de métriques telles que : l'identifiant, le degré, l'énergie, la k-densité, la mobilité,....

L'efficacité d'un algorithme de clustering est évaluée en termes du nombre de clusters formés et de la stabilité des clusters en fonction de la mobilité des nœuds. Le processus de clustering vise principalement à optimiser la maintenance des informations de la topologie du réseau et de réduire l'overhead de la diffusion pour la découverte des chemins.

2.3.2 Formation de clusters

Il existe plusieurs méthodes de formation de clusters. La plus répandue [16, 24, 25, 75, 92] s'exécute comme suit :

1. Chaque nœud devra connaître son voisinage par le biais des messages Hello,
2. Chaque nœud prend la décision selon sa connaissance locale de la topologie pour être cluster-head ou non,
3. Le nœud choisi comme cluster-head diffuse son statut dans son voisinage et invite ses voisins qui ne sont pas encore affiliés à d'autres clusters de le rejoindre.

Election de cluster-head

La phase d'élection de cluster-heads appelée aussi la phase Set-up utilise une métrique spécifique ou une combinaison de métriques pour chaque nœud telle que le plus grand/petit ID dans son voisinage, le degré de connectivité, la puissance de transmission, l'énergie restante ou la mobilité,.... ou bien un poids qui représente une combinaison de quelques métriques.

Communication intra-cluster et inter-cluster

Chaque cluster-head se charge des communications à l'intérieur de son cluster et maintient les informations de routage lui permettant de joindre les autres cluster-heads. De plus, comme les cluster-heads ne sont pas directement reliés, des nœuds passerelles sont aussi élus et utilisés pour les communications entre cluster-heads.

Maintenance des clusters

Dans le but de s'adapter aux changements de la topologie du réseau, une mise à jour des clusters est dynamiquement réalisée dans le cas où un cluster-head ou un membre migre d'un cluster C_i à un autre C_j. D'autre part, si le cluster-head garde son statut le plus longtemps possible, même s'il ne possède pas le poids maximum dans son propre cluster alors il perdra son rôle une fois qu'il s'éteindra i.e. sa batterie sera épuisée.

2.3.3 Quelques approches de clustering

Dans cette section, nous présentons et nous analysons les principaux algorithmes de construction de clusters dans les réseaux de capteurs et les réseaux ad hoc. Ces algorithmes de clustering regroupent la méthode de formation des clusters, les critéres utilisés pour élire les cluster-heads et la maintenance des clusters. L'analyse détaillée de ces algorithmes nous permet de tirer profits de leurs avantages et améliorer ses limitations.

Algorithmes de plus faible ID et de plus grand Degré

Baker et Ephremides [12] ont proposé l'un des premiers algorithmes de clustering pour les réseaux ad hoc. Cet algorithme est exécuté en mode synchrone tel qu'à chaque nœud est attribué un intervalle fini de temps appelé slot (TDMA) pour éviter les collisions. Un nœud v est choisi comme cluster-head par son voisin u si v a le plus grand identifiant dans le voisinage du nœud u i.e. parmi les nœuds $(N_1(u))$. Les nœuds choisis comme cluster-heads dans le réseau forment un ensemble dominant indépendant du graphe G.

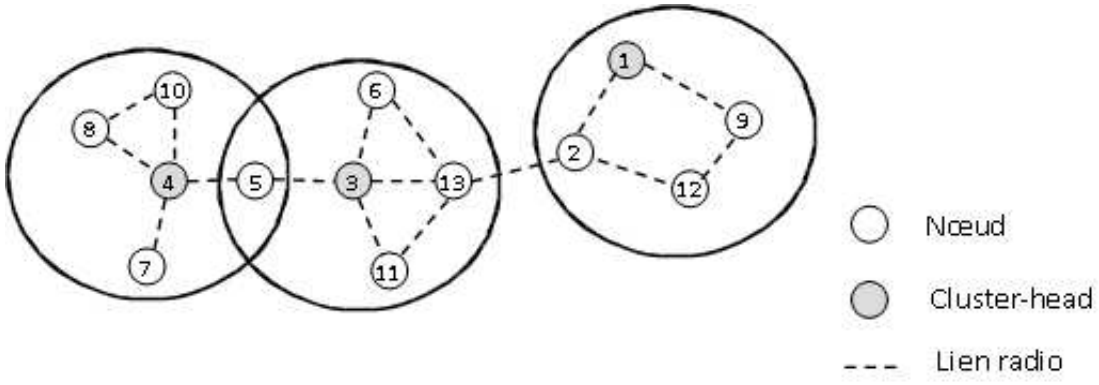


FIG. 2.1 – Formation de clusters basée sur ID

Dans [39], les auteurs ont proposé un autre algorithme "Plus Petit ID" (Lowest-Identifier ou Lowest-ID) pour la construction des clusters où chaque nœud se déclare cluster-head ou non en se basant sur son identifiant et ceux de ses voisins, ainsi que sur le statut de ses voisins comme il est illustré sur la figure 2.1. Dans l'algorithme "Plus Petit ID", un nœud peut avoir quatre statuts : ordinaire, cluster-head, membre, ou passerelle. Au début, tous les nœuds ont un statut de nœud ordinaire. Si un nœud u possède le plus petit identifiant dans son 1-voisinage, il se déclarera comme cluster-head et ses 1-voisins dont les identifiants sont supérieurs à celui du cluster-head le joignent et deviennent des nœuds membres. Sinon, il attendra que tous ses 1-voisins déclareront leurs statuts. Ainsi si un parmi eux se déclare cluster-head alors le nœud *u* déclare à son 1-voisinage son statut de nœud membre. Si tous les voisins du nœud *u* ayant un identifiant plus petit que celui de *u* qui a le statut de nœud membre, alors le nœud *u* se déclarera cluster-head. Une fois que tous les nœuds ont soit le statut de membre ou de cluster-head, alors si un nœud a parmi ses 1-voisins plus d'un cluster-head, il se déclarera passerelle. Le protocole de routage CBRP (Cluster Based Routing Protocol) [56] utilise l'algorithme "Plus Petit ID" pour la formation des clusters.

Par ailleurs, l'algorithme "plus petit ID" n'est pas commode pour le clustering dans les réseaux de capteurs parce que le nœud ayant le plus petit identifiant dans son voisinage peut devenir cluster-head même si sa réserve d'énergie est faible. De plus, les cluster-heads consomment en moyenne plus d'énergie que les autres membres des clusters, puisqu'ils sont responsables de la coordination entre les membres des clusters, l'agrégation de leurs données collectées et la transmission vers la station de base. Ainsi, si ces cluster-heads gardent leur statut de cluster-head pour une longue durée, il en résulte qu'ils consomment rapidement leur énergie et par conséquent ils causent des goulets d'étranglement dans leurs clusters. Dans les réseaux de capteurs mobiles, ces nœuds sont les premiers qui épuisent leur énergie. D'autre part, dans un environnement mobile, si le cluster-head est très mobile alors il détruira constamment la hiérarchie et déclenchera l'opération de re-clustering de nouveau.

Gerla et Tsai [43] ont opté pour générer des clusters en utilisant le degré des nœuds "Plus Grand Degré" (Highest-Degree) plutôt que leur identifiant. Le nœud de plus fort degré dans son voisinage se déclare cluster-head. Si deux nœuds voisins ont le même degré, alors le nœud ayant le plus petit identifiant ID deviendra cluster-head. L'algorithme "Highest-Degree" génère un nombre réduit de clusters puisqu'il favorise les nœuds ayant le plus fort degré pour être cluster-heads i.e. les nœuds qui couvrent plus de nœuds voisins. Dans un environnement mobile, cet algorithme produit des cluster-heads qui ne sont pas susceptibles de jouer leur rôle comme cluster-heads pour très longtemps puisque leurs degrés changent très fréquemment contrairement à l'algorithme du plus petit ID où les nœuds de faible identifiant ont tendance à garder le statut de cluster-head plus longtemps.

Les deux algorithmes "Plus Petit ID" et "Plus Grand Degré" génèrent une structure dans laquelle les clusters se recouvrent. Ainsi, les nœuds appartenant à plusieurs clusters jouent le rôle de passerelles. Cette structure implique seulement les cluster-heads et les passerelles dans l'opération d'acheminement des messages de contrôle et de diffusion.

Algorithmes CONID

Dans le but de tirer profit des avantages de ces deux algorithmes "plus petit ID" et "plus grand Degré", Chen et Stojmenović [27] ont proposé de combiner les deux algorithmes en un seul algorithme appelé "CONID". Cet algorithme considère le degré (connectivité) des nœuds comme clé primaire et l'identifiant ID des nœuds comme clé secondaire pour choisir les cluster-heads. Dans "CONID", à chaque nœud *u* du réseau est associée une paire $CONID_u = (CON_u, ID_u)$.

Cette paire indique la connectivité CON_i et l'identifiant ID_i d'un nœud. Si deux nœuds *u* et *v* ont respectivement les paires $CONID_u = (CON_u, ID_u)$ et $CONID_v = (CON_v, ID_v)$ alors le choix du cluster-head entre ces deux nœuds est illustré par le schéma algorithmique suivant :

si $(CON_u > CON_v) \vee (CON_u = CON_v \wedge ID_u < ID_v)$
alors "*u* est cluster-head"
sinon "*v* est cluster-head"

Cet algorithme génère aussi des clusters recouvrants et les nœuds qui appartiennent à plus d'un cluster sont considérés comme des nœuds passerelles.

Dans les algorithmes que nous avons présentés, l'opération de maintenance est très onéreuse parce que le mouvement d'un nœud peut détruire la structure et déclencher des réactions en chaîne pour la restructuration du réseau en clusters. C'est la raison pour laquelle Chang et al. [23] ont proposé une version améliorée des algorithmes "Plus Petit ID" et "Plus Grand Degré" appelée "Least Cluster Change" (LCC). Cette version améliorée contient une étape de maintenance qui permet de minimiser le coût de la restructuration du réseau en clusters lors d'une migration d'un nœud *u* d'un cluster C_i vers un autre C_j. En effet, cette opération de maintenance est exécutée comme suit : quand un nœud non-cluster-head *u* migre d'un cluster C_i vers un cluster C_j alors la réélection du cluster-head dans le cluster C_j n'aura pas lieu même si le nœud *u* possède les capacités d'être cluster-head de C_j. Cependant, quand un nœud non-cluster-head se déplace en dehors des clusters formés et ne migre pas vers un cluster existant alors il deviendra cluster-head et formera un nouveau cluster. En outre, quand deux cluster-heads CH_i et CH_j se trouvent voisins, alors l'un des deux (CH_i ou CH_j) devra abandonner le rôle de cluster-head selon le critère "Plus Grand Degré" et/ou " Plus Petit ID". De cette façon, l'algorithme LCC améliore considérablement la stabilité de la structure mais le coût de la restructuration du réseau reste toujours un peu onéreux vu qu'un seul nœud peut relancer le processus de clustering s'il n'existe pas de cluster-head dans son voisinage.

Algorithmes basés sur la mobilité

Dans le but d'assurer une certaine stabilité des clusters générés, Basu et al. [16] ont proposé un algorithme de clustering distribué appelé MOBIC (Lowest Relative Mobility Clustering Algorithm). Cet algorithme implique la mobilité relative des nœuds pour structurer le réseau en clusters. La mobilité relative d'un nœud représente le rapport des niveaux de puissance des transmissions successives reçues par un nœud de ses voisins. Ainsi, le nœud ayant la plus faible

mobilité dans son voisinage sera un bon candidat pour être cluster-head puisqu'il gardera un voisinage plus stable au cours du temps et favorisera la stabilité des clusters. D'autre part, MOBIC implique la même procédure de maintenance que LCC mais il ajoute une règle supplémentaire pour minimiser le coût de l'opération de maintenance : si deux cluster-heads CH_i et CH_j se trouvent voisins, alors l'un des deux n'abandonnera son statut de cluster-head qu'après une certaine période Δt, sinon ils garderont tous les deux leurs statuts. Cela permet de ne pas faire appel à l'opération de maintenance une fois que deux cluster-heads se trouvent dans le même voisinage. En effet, ce procédé a contribué de réduire à plus de 33% le nombre de changements des cluster-heads relativement au protocole LCC. Cependant, les limitations de l'algorithme LCC ne sont pas totalement éliminées. De plus, MOBIC exige que les nœuds doivent être capables d'estimer le niveau de signal avec leurs voisins pour calculer la mobilité relative.

Dans [10], les auteurs ont proposé un autre algorithme de clustering basé sur la mobilité relative pour créer des clusters dans les environnements mobiles. Les nœuds possédant une mobilité relative faible peuvent faire partie d'un même cluster. Chaque nœud mesure donc au cours du temps le niveau de signal qui l'unit avec chacun de ses voisins pour calculer sa mobilité relative à ses voisins. Le quotient de la puissance actuelle sur la puissance mesurée à l'intervalle de temps précédent est représentatif de la mobilité. En outre, deux nœuds ayant une trajectoire différente auront tendance à former des clusters disjoints. Bien que cet algorithme prenne en considération la métrique mobilité pour créer les clusters, il ne surpasse pas considérablement les algorithmes de clustering que nous avons présentés précédemment.

Algorithmes basés sur les poids des nœuds

Les protocoles que nous avons présentés dans les sections précédentes, impliquent une seule métrique pour élire les cluster-heads. Ce choix n'était pas judicieux pour engendrer la stabilité des clusters formés. D'autres algorithmes de clustering proposés dans la littérature, combinent plusieurs métriques pour élire les cluster-heads. Ces algorithmes associent un poids à chaque nœud. Ce poids est représenté par une somme pondérée des différentes métriques impliquées dans son calcul comme montré dans l'équation (2.2). Le coefficient de pondération de chaque métrique dépend de l'application et reflète son degré d'implication dans le calcul du poids. Par exemple, dans les réseaux de capteurs où l'énergie est une ressource précieuse, il est nécessaire de faire associer à la métrique énergie résidentielle un coefficient de pondération très élevé.

$$Weight(u) = \sum_{i=1}^{k} \alpha_i * P_i \quad \text{avec} \quad \sum_{i=1}^{k} \alpha_i = 1 \tag{2.2}$$

où α_i représente le coefficient de la métrique (le degré d'implication de la métrique) et P_i la valeur de la métrique.

L'algorithme WCA [24] (Weighted Clustering Algorithm) implique quatre métriques dans le calcul du poids d'un nœud : la différence de degré D_u, la somme des distances avec ses voisins P_u, la mobilité relative moyenne M_u et le temps de service en tant que cluster-head T_u tel que :

$$Weight(u) = \alpha * D_u + \beta * P_u + \gamma * M_u + \delta * T_u \tag{2.3}$$

$$\text{avec} \quad \alpha + \beta + \gamma + \delta = 1$$

La différence de degré D_u est la différence entre le degré du nœud *v* et une valeur M qui représente la taille seuil d'un cluster. La mobilité relative M_u est calculée de la même manière que dans MOBIC. Les distances P_u entre les nœuds sont calculées à l'aide d'un GPS. Le nœud ayant le plus petit poids dans son voisinage devient cluster-head. Bien que WCA ait de meilleures performances que les autres algorithmes que nous avons présentés en termes du nombre de clusters produits et du nombre de changements de cluster-heads, il a un inconvénient pour connaître le poids de tous les nœuds avant de commencer le processus de clustering et il épuise rapidement les batteries des cluster-heads. En conséquence, l'overhead induit par WCA est très élevé.

Les algorithmes DCA [14] (Distributed Clustering Algorithm) et DMAC [15] (Distributed Mobility Adaptive Clustering) sont des versions améliorées de l'algorithme "Plus Petit ID". Ces deux algorithmes considèrent que chaque nœud a un poids unique et les cluster-heads sont choisis à la base des poids des nœuds. Un nœud *u* est choisi pour être cluster-head s'il possède le plus grand poids dans son voisinage $N_1[u]$, autrement, il joint le cluster-head qui se trouve dans son voisinage. DCA suppose que la topologie du réseau ne change pas pendant l'exécution de l'algorithme du clustering alors que DMAC s'adapte aux changements de la topologie de réseau. Ainsi, DCA montre qu'il est bien adapté pour les réseaux dans lesquels les nœuds sont immobiles ou se déplacent avec une petite vitesse alors que DMAC sera plutôt utilisé pour les réseaux mobiles. Cependant, l'attribution des poids aux nœuds n'a pas été discutée dans les deux algorithmes et il n'y a aucune optimisation sur les paramètres du système tels que le nombre de paquets échangés et le contrôle de la puissance.

Tous les algorithmes que nous avons présentés, produisent des clusters recouvrants. Leur inconvénient est que la ré-affiliation d'un nœud mobile d'un cluster à un autre provoque la restructuration des clusters concernés et dans certains cas la restructuration de tout le réseau. Pour remédier à ce problème, des algorithmes de clustering ont été proposés, permettant la création de clusters disjoints c'est-à-dire un nœud ne peut appartenir qu'à un seul cluster. Ceci pour limiter l'opération de maintenance au niveau d'un seul cluster lors de la ré-affiliation d'un nœud d'un cluster à un autre.

Algorithmes générant des clusters disjoints

Yu et Chong [120] ont proposé un algorithme de clustering appelé 3hBAC (3-hop Between Adjacent Cluster-heads). 3hBAC génère des clusters disjoints tels que les cluster-heads de deux clusters adjacents se trouvent à trois sauts les uns des autres. Le nœud ayant le plus grand degré dans son voisinage se déclare cluster-head et ses 1-voisins le rejoignent et se déclarent "nœuds membres". Les nœuds voisins des nœuds membres et non voisins de cluster-heads se déclarent "non spécifié". Ces nœuds ne peuvent pas être des cluster-heads. Pour la maintenance des clusters, les auteurs ont combiné l'algorithme 3hBAC et l'algorithme LCC pour minimiser les changements de la structure. Si deux cluster-heads CH_i et CH_j se trouvent voisins pour une durée de temps seuil alors le cluster-head ayant le plus grand identifiant abandonne son rôle de cluster-head et deviendra nœud membre et ses voisins deviennent soit nœuds membres s'ils se trouvent dans le voisinage d'un cluster-head soit des nœuds non spécifiés. Cette technique permet de réduire le nombre de clusters produits tout en gardant l'information de connexion et évite la restructuration entière de tout le réseau lors de la ré-affiliation d'un nœud d'un cluster à un autre. 3hBAC fournit aussi de bonnes performances que Highest-Degree en termes du nombre moyen de clusters formés et du temps moyen de garde du statut cluster-head ou membre [120].

Nous avons présenté des algorithmes de clustering qui produisent des 1-clusters i.e. des clusters où les membres sont à un saut de leur cluster-head correspondant. Parmi ces algorithmes, nous avons vu ceux qui produisent des 1-clusters recouvrants (les 1-clusters à passerelles). Dans ce type de structure, seuls les cluster-heads et les passerelles sont responsables pour relayer les paquets de contrôle et les paquets de découverte de chemins pour la fonction de routage. Ceci permet de réduire considérablement l'overhead de la diffusion puisque la redondance des retransmissions et les collisions sont minimisées dans le processus de diffusion. D'autre part, nous avons présenté les techniques de clustering qui génèrent des clusters disjoints. Ces clusters sont stables dans un environnement mobile comparativement à ceux créés par les autres techniques.

Ils permettent la réutilisation spatiale de fréquences ou les codes CDMA i.e. les nœuds de deux clusters non adjacents peuvent utiliser le même code CDMA. Cependant, l'inconvénient principal des techniques générant des 1-clusters est la génération d'un grand nombre de clusters dans le réseau ce qui cause le problème de congestion.

Des approches plus récentes permettent la formation de clusters dont les membres sont à *k* sauts de leur cluster-head correspondant, appelés aussi des k-clusters.

Algorithmes générant des k-clusters

La technique permettant la génération des k-clusters où tout membre dans un cluster est au plus à k sauts de son cluster-head correspondant, est une version améliorée de celle des 1-clusters. Les k-clusters formés couvrent une zone géographique plus large et ils sont stables relativement aux 1-clusters. Ils permettent d'implémenter d'autres fonctions telles la découverte de services et le routage hiérarchique.

Chen et al. [28] ont proposé un algorithme semblable à celui présenté par Lin et Gerla [75]. Cet algorithme s'exécute sur le k-voisinage d'un nœud : tout nœud non affilié possédant le plus fort poids (Plus Petit ID ou Plus Grand Degré) parmi ses k-voisins non encore affiliés devient cluster-head. Chen et al. ont adapté l'opération de maintenance de [75] en prenant en compte le rayon du cluster. Cependant, nous retrouvons toujours les mêmes inconvénients que présentent les algorithmes générant des 1-clusters, à savoir un petit changement dans la topologie du réseau peut provoquer une restructuration de tout le réseau. En outre, dans les réseaux denses tels que les réseaux de capteurs, la taille des clusters peut être importante, ce qui peut épuiser rapidement les batteries des cluster-heads et par la suite causer des goulets d'étranglement dans les clusters.

Dans [74], Lin et Chu ont proposé un autre algorithme pour générer des k-clusters. Cet algorithme n'utilise aucune métrique pour le choix des cluster-heads et s'exécute comme suit : lorsqu'un nœud *u* joint le réseau, il passe à l'état "Initialisation". Puis, il consulte l'état de ses voisins, s'ils sont à l'état "Initialisation" ou ils sont affiliés à des clusters, et dans ce cas à quelle distance se situe leur cluster-head correspondant ? Si tous les voisins de *u* sont à l'état "Initialisation" alors le nœud *u* se déclarera comme cluster-head et diffusera sa décision dans son voisinage. Tous les k-voisins de *u* qui ne sont pas affiliés à aucun autre cluster-head plus proche que *u* joindront le cluster formé par le nœud *u*. Sinon le nœud *u* joint le cluster de son voisin dont le cluster-head est le plus proche et qui se trouve à au plus k sauts de lui. Si tous les cluster-heads de ses voisins sont à k sauts du nœud *u*, alors *u* se déclarera cluster-head et invitera ses k-voisins

qui ont des cluster-heads plus éloignés que *u* à le rejoindre.

Dans cette technique, l'opération de maintenance se déclenchera quand deux cluster-heads se trouvent à une distance D qui est inférieure à *k* sauts $(D < k)$. Le cluster-head ayant le plus petit identifiant cèdera son rôle et ses membres doivent affilier à d'autres cluster-heads. Cette technique produit des nombres réduits de k-clusters disjoints où les cluster-heads sont éloignés d'au moins $(k + 1)$ sauts. Cependant, l'opération de maintenance s'avère un peu coûteuse quand un cluster-head cède son rôle. D'autre part, l'algorithme proposé ne prend pas en compte la contrainte taille dans la formation des clusters, ce qui permet de générer des clusters ayant une taille importante dans le cas des réseaux denses tels que les réseaux de capteurs.

Fernandess et Malkhi [42] ont proposée une approche originale pour générer des k-clusters. Cette approche s'exécute en deux phases. La première phase consiste à construire un arbre couvrant T (Spanning Tree) en utilisant l'algorithme présenté dans [115] pour la construction d'un ensemble dominant connecté de cardinalité minimale (MCDS). La seconde phase consiste à partitionner l'arbre couvrant T en des ensembles disjoints S_i où S_i est un 2k-sous-arbre de T i.e. un sous-arbre dont le diamètre est au plus 2*k* sauts. Le diamètre d'un sous-arbre S_i est représenté comme suit :

$$Diam(S_i) = max\{d(u, v) : u, v \in S_i\} \tag{2.4}$$

Chaque sous-arbre S_i consiste en un k-cluster. Cependant, la construction de l'arbre couvrant T peut créer un trafic de contrôle important qui se répercute sur le temps de convergence et la complexité en messages de l'algorithme.

Dans [9], Amis et al. ont proposé une heuristique appelée Max-Min D-cluster pour construire des d-clusters non recouvrants (disjoints) où D est un paramètre de l'heuristique. Le nombre de clusters formés dépend du paramètre D. Max-Min D-cluster utilise la métrique ID pour élire les cluster-heads et elle s'exécute en trois phases. Lors de la première phase appelée aussi 1^{er} d-round, chaque nœud diffuse la valeur WINNER qui représente son identifiant à ses d-voisins (voisins se trouvant à d sauts de lui) et assigne à cette valeur le plus grand ID entendu parmi ses d-voisins. Puis, il diffuse cette nouvelle valeur WINNER à ses d-voisins (procédure Floodmax). Au cours de la deuxième phase ($2^{\text{ième}}$ d-round), chaque nœud garde le plus petit $ID(WINNER)$ parmi ceux qu'il a reçus durant cette phase (le plus petit ID parmi les plus grands ID diffusés). Le choix du cluster-head est basé sur les identifiants enregistrés lors des deux phases précédentes (1^{er} d-round et $2^{\text{ième}}$ d-round). Un nœud *u* se déclare cluster-head s'il reçoit son identifiant ID

$(WINNER = my\_ID)$ lors de la deuxième phase de la diffusion ($2^{\text{ième}}$ d-round). Sinon, si le nœud *u* a reçu WINNER durant chacune des deux phases 1 et 2, alors il choisira le nœud dont l'identifiant est attribué à la valeur WINNER comme cluster-head (WINNER correspond à la plus petite valeur reçue par le nœud *u*). Sinon, *u* choisit comme cluster-head le nœud ayant le plus grand ID dans son d-voisinage. Cependant, cette technique augmente considérablement l'overhead du réseau puisqu'elle utilise un nombre important de messages pour élire les cluster-heads. La complexité en messages est $\theta(n * 2d)$. D'autre part, cette technique ne tient pas compte de la mobilité des nœuds et du changement de la topologie.

Dans [8], Amis et Prakash ont proposé une version améliorée de l'algorithme Max-Min D-cluster appelée Cluster-head load-balancing. Ceci dans le but est d'assurer un équilibrage de charge entre les nœuds et d'éviter qu'un nœud reste cluster-head pour une longue durée car ceci épuise sa batterie rapidement et cause par la suite des goulets d'étranglement dans son cluster. Ainsi, les nœuds sont choisis cluster-heads pour une période de temps $Time_{CH}$. La technique proposée implique une métrique absolue appelée Identifiant Virtuel (VIP) pour élire les cluster-heads. Le nœud ayant le plus grand VID dans son k-voisinage devient cluster-head. En cas d'égalité, le nœud ayant opéré le moins de fois comme cluster-head sera élu cluster-head et en cas d'égalité toujours, c'est le nœud qui a le plus grand ID qui deviendra cluster-head. Au début, l'identifiant virtuel de chaque nœud est initialisé par son propre identifiant i.e. $VID(u) = ID(u)$. Après chaque processus d'élection de cluster-head, chaque nœuds non cluster-head incrémente de 1 son identifiant virtuel i.e. $VID(u) = VID(u) + 1$ jusqu'à ce que son VID atteigne une valeur maximale de seuil MAX_{COUNT}. Un nœud joue le rôle de cluster-head pour une période de temps $Time_{CH}$ et après cette période, il initialisera son VID à zéro et abandonnera le statut de cluster-head. Lorsque deux cluster-heads se trouvent à moins de *k* sauts l'un de l'autre, le cluster-head ayant le plus faible VID abandonnera son statut. La technique "cluster-head load-balancing" quoiqu'elle assure une certaine stabilité de la structure, nécessitera durant l'élection des cluster-heads une synchronisation des nœuds (non cluster-heads) pour l'incrémentation de leurs VID et des cluster-heads pour comptabiliser la période pour jouer ce rôle. Cette synchronisation est coûteuse en nombre de messages échangés entre les différents nœuds.

Algorithmes basés sur des critères absolus

Il est intéressant de borner la distance entre les membres d'un cluster afin de limiter le trafic de contrôle induit par les fonctions à implémenter telles le routage et la découverte de services. Il est également intéressant de générer des clusters homogènes en taille. Dans cette section, nous

présentons quelques approches de clustering se basant sur des critères absolus pour construire les clusters.

Dans [59], Krishna et al. ont proposé une approche de clustering pour le routage dans les réseaux dynamiques. Les auteurs ont utilisé la distance entre les nœuds comme critère absolu pour la formation des clusters. Ainsi, dans un même cluster, la distance séparant deux membres est au plus *k* sauts. Les clusters générés sont sans cluster-heads et ils sont recouvrants. Quoique cette approche réduise considérablement le coût des opérations de maintenance, elle implique beaucoup de messages de contrôle pour coordonner l'ensemble des nœuds d'un même cluster. D'autre part, cette approche n'est pas commode pour les réseaux de capteurs car elle implique seulement le critère distance entre les nœuds pour former les clusters. Or, les réseaux de capteurs sont composés généralement d'un grand nombre de nœuds. D'où, les clusters générés par cette approche peuvent être de grandes tailles en termes de nombre de nœuds.

Bannerjee et Khuller [13] ont proposé une approche qui permet de construire des k-clusters limités par leur taille plutôt que par leur rayon. Cette approche commence par créer un arbre couvrant (spanning tree) tout le réseau. Ensuite, les clusters sont formés par branche à partir des feuilles (du bat en haut), où les sous-arbres sont fusionnés récursivement en clusters jusqu'à ce qu'une branche comporte entre *k* et $(2 * k - 1)$ nœuds selon le schéma d'exécution suivant : soit B_i et B_j deux branches de l'arbre couvrant construit,

- Si $|B_i| + |B_j| < 2 * k$ alors B_i et B_j sont fusionnés dans un même cluster.
- Si $|B_i| + |B_j| < 2 * k$, $|B_i| > k$ et $|B_j| > k$ alors B_i et B_j forment des clusters.
- Si $|B_i| + |B_j| < 2 * k$ et $|B_i| < k$ alors B_i constitue un cluster partiel.

Ainsi, à la fin de l'exécution du processus de clustering, des clusters de cardinalité comprise entre k et $(2 * k - 1)$ sont construits mais peu de clusters partiels. Les auteurs ont proposé une maintenance, permettant de fusionner les clusters de petite taille ou au contraire de les scinder. Cependant, la construction d'un arbre recouvrant semble coûteuse en temps de convergence et en trafic de contrôle.

2.3.4 Algorithmes de clustering conçus pour les réseaux de capteurs

Les réseaux de capteurs sans fil sont apparentés aux réseaux ad hoc de point de vu infrastructure, architecture, autonomie d'énergie et utilisation des ondes radio pour communiquer.

Cependant, leurs spécificités, leurs objectifs et leurs besoins sont différents. Le clustering dans les réseaux de capteurs est une manière efficace de minimiser la consommation d'énergie dans un cluster en exécutant les fonctions d'agrégation et de fusion de données dans le but de diminuer le nombre de messages transmis à la station de base.

Récemment, plusieurs techniques de clustering ont été proposées pour traiter les principaux défis dans les réseaux de capteurs. Ces techniques visent à maintenir les informations de la topologie du réseau, réduire l'overhead généré par la découverte de routes, et minimiser la consommation d'énergie en tenant compte de la spécificité de ces réseaux. Elles sont dans la plupart des cas orientées énergie i.e. elles visent à prolonger la durée de vie du réseau, et dans certains cas elles sont orientées QoS (qualité de service).

Dans cette section, nous présentons les principales techniques de clustering proposées pour les réseaux de capteurs dans la littérature.

LEACH (Low Energy Adaptive Clustering Hierarchy)

Dans [47], Heinzelman et al. ont proposé un algorithme de clustering distribué appelé LEACH pour le routage dans les réseaux de capteurs homogènes. LEACH choisit aléatoirement les nœuds cluster-heads et attribue ce rôle aux différents nœuds selon la politique de gestion Round-Robin i.e. tour à tour pour garantir une dissipation équité d'énergie entre les nœuds. Dans le but de réduire la quantité d'informations transmise à la station de base, les cluster-heads agrègent les données capturées par les nœuds membres qui appartiennent à leur propre cluster, et envoient un paquet agrégé à la station de base.

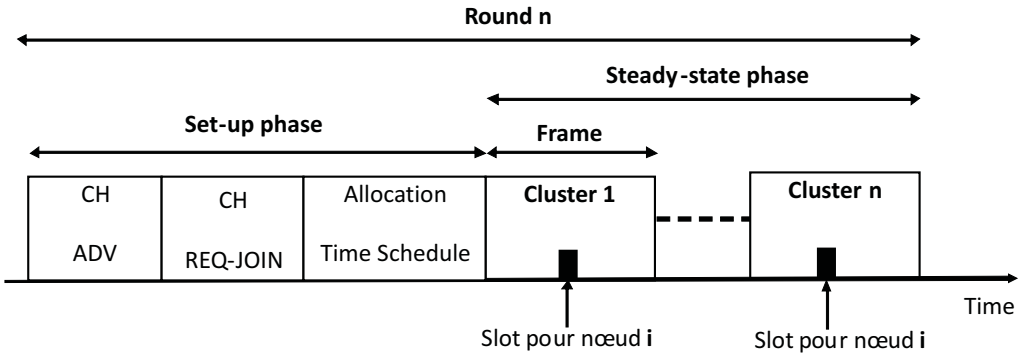


FIG. 2.2 – Formation de clusters par LEACH

LEACH est exécuté en deux phases : la phase set-up et la phase d'état stationnaire (steady-state phase) suivant la figure 2.2. Dans la première phase, les cluster-heads sont sélectionnés et les clusters sont formés, et dans la seconde phase, le transfert de données vers la station de base aura lieu. Durant la première phase, le processus d'élection des cluster-heads est déclenché pour choisir les futurs cluster-heads. Ainsi, une fraction prédéterminée de nœuds "p" s'élisent comme cluster-heads selon le schéma d'exécution suivant : durant une période r, un nœud u choisit un nombre aléatoire r_u dont la valeur est comprise entre 0 et 1 $(0 < r_u < 1)$. Si r_u est inférieur à une valeur seuil $T(u)$ alors le nœud *u* deviendra cluster-head durant la période courante soit r cette période sinon le nœud *u* devrait rejoindre le cluster-head le plus proche dans son voisinage. La valeur seuil $T(u)$ est calculée comme suit :

$$T(u) = \begin{cases} \frac{p}{1-p\times(r\times \bmod (\frac{1}{p}))} & \text{si } u \in G \\ 0 & \text{sinon} \end{cases} \tag{2.5}$$

où p : le pourcentage de cluster-heads dans la plupart des cas $p = 5\%$, r : la période courante et G : l'ensemble des nœuds qui n'étaient pas cluster-heads durant la période précédente. Cependant, bien que LEACH puisse augmenter la durée de vie du réseau, il présente certaines limitations. LEACH suppose que tous les nœuds puissent transmettre des données avec une grande puissance pour atteindre la station de base et que chaque nœud a une puissance de calcul lui permettant de supporter différentes couches MAC. Par conséquent, LEACH n'est pas souhaitable pour les réseaux déployés dans de vastes régions. En outre, LEACH choisit aléatoirement la liste des cluster-heads et il ne pose aucune contrainte sur leur distribution ainsi que sur leur niveau d'énergie. D'où, les cluster-heads peuvent se concentrer dans un même endroit et par conséquent, il pourrait exister des nœuds isolés (sans cluster-head). D'autre part, dans LEACH, l'agrégation des données est centralisée ainsi qu'elle est exécutée périodiquement. Or, dans certains cas, la transmission périodique des données pourrait ne pas être nécessaire et par suite elle pourrait épuiser rapidement l'énergie limitée des capteurs.

Les variantes de LEACH

Dans [81], les auteurs ont comparé les réseaux homogènes et hétérogènes en termes de dissipation d'énergie dans tout le réseau et ils ont analysé les performances des réseaux à un saut et ceux à sauts multiples. Ils ont choisi pour cela LEACH comme représentant des réseaux homogènes et ils l'ont comparé avec un réseau hétérogène à un saut. Les auteurs ont constaté que l'utilisation des communications à un saut entre les membres d'un cluster et leur cluster-head

correspondant pourrait ne pas être le bon choix quand l'index k de perte de propagation $(k > 2)$ pour les communications intra-clusters est plus grand. D'autre part, LEACH pourrait produire des clusters possédant une taille importante dans les réseaux denses et des clusters dont la taille est limitée dans les réseaux de petites tailles. Dans ces deux cas, les cluster-heads pourraient rapidement épuiser leur puissance de batterie. Dans les réseaux denses, les cluster-heads coordonnent entre plusieurs membres des clusters alors que dans les réseaux de petites tailles, les cluster-heads sont placés loin de la station de base ce qui nécessite des transmissions de forte puissance. Dans le même article, les auteurs ont proposé une version améliorée de LEACH appelée M-LEACH [81] (Multi-hop LEACH), dans laquelle les membres d'un cluster peuvent être à plus d'un saut de leur cluster-head correspondant et communiquent avec lui en mode multi-saut. Ainsi, ils ont illustré les cas dans lesquels M-LEACH surpasse LEACH. Cependant, cette version proposée exige que chaque capteur doive être capable d'agréger les données, ce qui augmente l'overhead pour tous les capteurs. Pour améliorer cette stratégie, dans [117], les auteurs se sont focalisés sur les réseaux de capteurs hétérogènes, dans lesquels deux types de capteurs sont déployés : capteurs de grandes capacités (Super Sensor) et capteurs simples. Les capteurs de grandes capacités ont des capacités de traitement et de communication si élevées et agissent comme cluster-heads, alors que les autres sont des capteurs simples avec une puissance limitée, affiliés au cluster-head le plus proche dans leur voisinage et communiquent avec lui directement ou en mode multi-saut.

En outre, une autre variante de LEACH appelée LEACH-C [48] a été conçue pour améliorer les performances de LEACH. Cette variante utilise une architecture centralisée pour choisir les cluster-heads tout en impliquant la station de base et l'information de localisation des capteurs. Cependant, elle augmente considérablement l'overhead du réseau puisque tous les capteurs devront envoyer leurs informations de localisation à la station de base en même temps pendant chaque phase d'élection de cluster-heads. Plusieurs travaux présentés dans la littérature ont prouvé qu'une telle architecture centralisée ne supporte pas le passage à l'échelle et est plus particulièrement appropriée à des réseaux de petite taille.

D'une manière similaire à LEACH-C, BCDCP [63] (Base-Station Controlled Dynamic Clustering Protocol) implique le niveau d'énergie des capteurs envoyé à la station de base pour construire des clusters homogènes durant la phase d'installation ($1^{\text{ière}}$ phase). La station de base choisit aléatoirement les cluster-heads tout en garantissant une distribution uniforme de leurs emplacements dans la zone d'intérêt dans laquelle ils sont déployés, et exécute un algorithme itératif de fusion pour trouver le nombre optimal de clusters. Puis, elle établit les routes inter-clusters

(CH-to-CH) pour l'acheminement des données d'un cluster-head à un autre, et crée un schedule pour chaque cluster qui le diffuse dans le réseau. Durant la deuxième phase, les cluster-heads transmettent les données collectées à la station de base par des chemins CH-to-CH [82]. Néanmoins, BCDCP présente les mêmes limitations que LEACH-C puisqu'il utilise une architecture centralisée pour élire les cluster-heads.

Les techniques de clustering que nous avons présentées dans cette section, quoiqu'elles préconisent une solution garantissant l'équilibre des charges dans l'élection des cluster-heads, ont un impact négatif sur les cluster-heads, puisque leur choix se fait aléatoirement. Or, ces derniers consomment leur énergie plus rapidement qu'un nœud ordinaire puisqu'ils supportent des fonctions additionnelles comme l'agrégation des données et le routage. Le choix d'un cluster-head qui a un niveau d'énergie plus faible, pourrait vite devenir un goulet d'étranglement de son cluster. D'autre part, dans la phase de reconstruction des clusters, un overhead de communications et de calculs est généré puisque tous les capteurs envoient simultanément leurs niveaux d'énergie à la station de base et la connaissance appropriée de la topologie du réseau est exigée.

HEED (Hybrid Energy-Efficient Distributed Clustering)

Younes et Fahmy [119] ont proposé un algorithme de clustering distribué appelé HEED pour les réseaux de capteurs ad hoc. Contrairement aux techniques précédentes, HEED ne fait aucune restriction sur la distribution et la densité des nœuds ainsi qu'il ne dépend pas de la topologie du réseau et sa taille. HEED sélectionne les cluster-heads selon un critère hybride regroupant l'énergie restante des nœuds et un second paramètre tel que le degré des nœuds. Il vise à réaliser une distribution uniforme des cluster-heads dans le réseau et à générer des clusters équilibrés en taille. Un nœud u est élu comme cluster-head avec une probabilité P_{CH} égale à :

$$P_{CH} = C_{prob}\frac{E_u}{E_{Total}} \tag{2.6}$$

où E_u est l'énergie restante du nœud u, E_{Total} l'énergie globale dans le réseau et C_{prob} est le nombre optimal de clusters. Cependant, l'évaluation de E_{Total} présente certaine difficulté puisque HEED opère sans d'autres protocoles de routage et en absence de toute commande centrale ainsi que le calcul le nombre optimal de clusters C_{prob}. D'autre part, avec HEED, la topologie en clusters ne réalise pas la consommation minimale d'énergie dans les communications intra-cluster et les clusters générés ne sont pas tellement équilibrés en taille.

PEGASIS (Power-Efficient Gathering in Sensor Information Systems)

Dans [77], Lindsey et Raghavendra ont proposé une version améliorée de LEACH appelée PEGASIS. L'idée principale de PEGASIS est de former une chaîne entre les nœuds de sorte que chaque nœud reçoive et communique à un voisin proche. Les données collectées sont transmises d'un nœud à un autre qui les agrège jusqu'à ce qu'elles arrivent à un nœud particulier qui les transmet à la station de base. Les nœuds qui transmettent les données à la station de base, sont choisis tour à tour selon un round-robin dans le but est de réduire l'énergie moyenne dépensée par un nœud durant une période (round). Contrairement à LEACH, PEGASIS évite la formation des clusters et procure à un seul nœud dans la chaîne l'envoi de données à la station de base. Les résultats de simulation ont montré que PEGASIS peut prolonger de deux à trois fois la durée de vie d'un réseau de capteurs relativement à LEACH en fonction du critère choisi pour évaluer la durée de vie d'un réseau i.e. quand 1%, 20%, 50% ou 100% des nœuds épuisent leurs batteries. Un tel gain de performance est réalisé par l'élimination de l'overhead causé par le processus de formation de clusters dans LEACH, et par réduction du nombre de transmissions et de réceptions par utilisation de l'agrégation de données. Bien que l'overhead du clustering soit évité, PEGASIS exige toujours un ajustement dynamique de la topologie puisqu'un nœud devrait connaître le niveau d'énergie de ses voisins avant de relayer ses données. Cependant, un tel ajustement de la topologie pourrait causer un overhead important en particulier dans les réseaux les plus utilisés. En outre, PEGASIS suppose que tout nœud est capable de communiquer directement avec la station de base. Or, cette supposition est loin de la réalité car les capteurs communiquent généralement en mode multi-sauts pour atteindre la station de base. D'autre part, PEGASIS suppose que tous les nœuds maintiennent une table contenant les localisations de tous les autres nœuds dans le réseau. En résumé, PEGASIS est adapté seulement aux capteurs sans fil dont les nœuds sont immobiles. Son évaluation dans des environnements mobiles pourrait dégrader considérablement ses performances.

TEEN (Threshold-sensitive Energy Efficient sensor Network protocol)

Manjeshwar et Agrawal [79] ont proposé une technique de clustering appelée TEEN pour les applications critiques où le changement de certains paramètres peut être brusque. L'architecture du réseau est basée sur un groupement hiérarchique à plusieurs niveaux où les nœuds les plus proches forment des clusters. Puis ce processus de clustering passe au deuxième niveau jusqu'à ce que la station de base soit atteinte. Après la formation des clusters, chaque cluster-head transmet à ses membres deux seuils : un seuil Hard H_T (hard threshold), qui est la valeur seuil du

paramètre contrôlé (surveillé) et un seuil Soft S_T (soft threshold) représentant une petite variation de la valeur du paramètre contrôlé. L'occurrence de cette petite variation S_T permet au nœud qui la détecte de la signaler à la station de base en transmettant un message d'alerte. Par conséquent, le seuil Soft réduira le nombre de transmissions puisqu'il ne permet pas la transmission s'il y a peu ou pas de variation de la valeur du paramètre contrôlé.

Au début, les nœuds écoutent le médium continûment et lorsque la valeur captée du paramètre contrôlé dépasse le seuil Hard, le nœud transmet les données i.e. un changement brusque d'un certain paramètre est survenu. La valeur captée est stockée dans une variable interne appelée SV. Puis, les nœuds ne transmettront des données que si la valeur courante du paramètre contrôlé est supérieure au seuil hard H_T ou diffère du SV d'une quantité égale ou plus grande que la valeur du seuil Soft S_T. Puisque la transmission d'un message consomme plus d'énergie que la détection des données, alors la consommation d'énergie dans TEEN est moins important que dans les protocoles proactifs ou ceux qui transmettent des données périodiquement tels que LEACH. Cependant, l'inconvénient principal de ce protocole est que, si les seuils H_T et S_T ne sont pas reçus, les nœuds ne communiqueront jamais, et aucune donnée ne sera transmise à l'utilisateur, ainsi la station de base ne connaît pas les nœuds qui ont épuisés leur énergie. TEEN n'est pas souhaitable pour les applications qui nécessitent des envois périodiques de données.

Pour remédier à ces limitations, les auteurs ont proposé une extension de TEEN appelée APTEEN (Adaptive Threshold-sensitive Energy Efficient sensor Network protocol). APTEEN est un protocole hybride qui change la périodicité et les valeurs seuils utilisées dans TEEN selon les besoins de l'utilisateur et le type d'application. Dans APTEEN, les cluster-heads transmettent à leurs membres les paramètres suivants :

- l'ensemble de paramètres physiques auxquels l'utilisateur est intéressé pour obtenir des informations (A).
- Les seuils : seuil Hard H_T et seuil Soft S_T.
- Un Schedule TDMA permettant d'assigner à chaque nœud un intervalle fini de temps appelé slot.
- Un compteur de temps (CT) : c'est la période de temps maximum entre deux transmissions successives d'un nœud.

Dans APTEEN, les nœuds surveillent en continu l'environnement. Ainsi, les nnœuds qui détectent une valeur d'un paramètre qui dépasse le seuil H_T, transmettent leurs données. Une

fois qu'un nœud détecte une valeur qui dépasse H_T, il ne transmet les données au cluster-head que si la valeur de ce paramètre change d'une quantité égale ou plus supérieure à S_T. Si un nœud ne transmet par de données pendant une période de temps CT, il devrait faire une capture de données et les retransmettre.

APTEEN offre une grande flexibilité qui permet à l'utilisateur de choisir l'intervalle de temps CT, et les valeurs seuils H_T et S_T pour que la consommation d'énergie soit contrôlée par la variation de ces paramètres. Cependant, APTEEN nécessite une complexité supplémentaire pour implémenter les fonctions de seuils et de périodes de temps CT. Ainsi, l'overhead et la complexité associés à la formation des clusters à plusieurs niveaux par TEEN et APTEEN sont assez élevés.

Virtual Grid Architecture routing (VGA)

Dans [5], les auteurs ont proposé une approche de clustering pour maximiser la durée de vie dans les réseaux de capteurs dont les nœuds sont immobiles ou se déplacent avec une faible vitesse. Ils ont utilisé l'approche GPS-free [94] pour construire des clusters fixes, disjoints, et homogènes en taille avec des formes symétriques. Dans [5], la zone de déploiement des réseaux de capteurs est divisée en une topologie virtuelle rectiligne contenant des petites zones ayant la forme d'un carré, et dans chacune, un nœud est choisi comme cluster-head. L'agrégation de données est réalisée en deux niveaux : local et global. L'agrégation locale est réalisée par l'ensemble des cluster-heads appelés aussi Local Aggregators(LAs), alors que l'agrégation globale est réalisée par un sous ensemble de LAs appelés Master Aggregators (MAs). Cependant, la détermination de l'ensemble MAs est un problème NP-difficile. Les heuristiques qui ont été proposées pour former l'ensemble MAs à partir de l'ensemble LAs, avaient comme objectif la maximisation de la durée de vie des réseaux de capteurs. Par exemple, dans l'heuristique CBAH [4] (Cluster-Based Aggregation Heuristic), l'ensemble MAs est choisi selon la capacité des éléments de LAs. Les membres d'un même cluster surveillent le même phénomène, et leurs lectures (détections) sont corrélées par leur LA correspondant. Ce dernier à son tour transmet ces données corrélées à son MA correspondant.

Sensor aggregates routing

Fang et al. [40] ont proposé une approche de clustering pour l'agrégation des données dans les réseaux de capteurs surveillant plusieurs cibles. Ces cibles peuvent être stationnaires ou se déplaçant à n'importe quel moment indépendamment des états des autres cibles. L'objectif de cet algorithme est de déterminer le nombre de cibles et les localisations approximatives des clus-

ters associés aux cibles dans la zone d'intérêt. Puis on recalcule quand les cibles se déplacent, joignent, ou quittent la zone d'intérêt. Au début, les nœuds examinent les caractéristiques spatiales des signaux associés aux cibles quand plusieurs cibles sont dans la proximité des unes des autres. Puis, ils sont groupés en clusters selon la puissance du signal détecté de sorte qu'il y ait un seul pic par cluster. Un pic pourrait représenter une cible, plusieurs cibles proches, ou aucune cible quand le pic serait produit par les atténuations du signal. Le processus d'élection des cluster-heads se fait dans un 1-voisinage et le nœud ayant le plus grand paysage de champ de signal parmi ses 1-voisins se déclarera cluster-head. Ainsi, le choix des cluster-heads (Sensor Aggregates) se fait selon l'allocation des ressources aux tâches de détection et de communication.

Dans [40], Fang et al. ont proposé trois algorithmes pour la création des nœuds qui agrègent les données (Node Aggregates) :

- DAM (Distributed Aggregate Management) est un protocole distribué conçu pour surveiller une cible. Il comporte un prédicat de décision P pour chaque nœud afin qu'il décide s'il devrait participer à l'agrégation de données et un arrangement d'échange de message M concernant (au sujet) la façon d'appliquer le prédicat de groupement aux nœuds. Le but de DAM est d'élire les cluster-heads et maintenir les informations locales sur le paysage du signal. Dans DAM, seuls les nœuds avec une puissance de signal dépassant le seuil *ThresholdElection* peuvent participer au processus d'élection de cluster-head.
- EBAM (Energy-Based Activity Monitoring) est une extension de DAM. Il fournit une solution pour déterminer le nombre de cibles dans un cluster. Il suppose que chaque cible a la même puissance. Ainsi, lorsque la puissance d'une cible dans un cluster est connue, le nombre de cibles peut être déduit de la puissance totale du signal dans le cluster.
- EMLAM (Expectation-Maximization Like Activity Monitoring)

Geographic Adaptive Fidelity (GAF)

GAF [111] est un protocole de routage basé sur la localisation des nœuds. Il est conçu pour les réseaux ad hoc et les réseaux de capteurs. L'information de localisation utilisée dans GAF pourrait être fournie à l'aide d'un GPS ou d'autres techniques ou systèmes de localisation [18, 37]. GAF consiste à partitionner la zone où les nœuds sont déployés en des petites zones formant des grilles virtuelles telles que, pour deux grilles adjacentes G_i et G_j, tous les nœuds de G_i peuvent communiquer avec tous les nœuds G_j. Ainsi, avec cette politique de partitionnement,

la fidélité du routage (routing fidelity) est assurée i.e. il existe au moins un chemin entre tout nœud du réseau et la station du base. Dans chaque grille, les nœuds élisent parmi eux un seul nœud pour rester à l'état actif et les autres passent à l'état sommeil pour une certaine période de temps. Ce nœud sera responsable pour surveiller et transmettre les données à la station de base. Par conséquent, GAF conserve l'énergie en faisant passer les autres nœuds de la grille à l'état sommeil sans affecter le niveau de la fidélité du routage.

Dans GAF, les nœuds peuvent être dans l'un des états suivants : Sommeil (Sleeping), Actif (Active), et Découvert (Discovery). Initialement, tous les nœuds sont dans l'état Discovery. Cet état a pour durée T_d et au cours de laquelle, les nœuds échangent les messages Discovery pour trouver les autres nœuds dans la même grille. Après l'écoulement de la période T_d, les nœuds diffusent leurs messages Discovery et rentrent dans l'état actif. Puis au niveau de chaque grille, les nœuds élisent parmi eux un nœud pour rester à l'état actif pour une durée T_a et eux passent à l'état sommeil. Afin de supporter la mobilité, chaque nœud dans la grille estime son temps de départ de la grille et l'envoie à ses voisins. Les nœuds se trouvant dans l'état sommeil ajustent leur temps de sommeil T_s pour garder la fidélité du routage. Ainsi, après l'expiration de la période T_a, les nœuds se trouvant dans l'état sommeil se réveillent et un parmi eux passe à l'état actif. Dans [111], les auteurs ont supposé que les nœuds étaient uniformément distribués sur une surface topographique A et que chacun d'eux avait un rayon de transmission R. Les clusters générés ont la forme carrée et ont la même dimension. La dimension du carré dépend de la puissance de transmission d'un nœud et elle est égale à $\frac{R}{\sqrt{5}}$.

GAF assure la fidélité du routage tout en minimisant l'énergie dissipée dans tout le réseau. Cependant, dans les environnements où les nœuds sont fortement mobiles, le nœud actif pourrait quitter la grille. Ainsi, la fidélité du routage est réduite et par conséquent, le nombre de paquets perdus sera important.

Dans cette première partie, nous avons étudié plusieurs techniques de clustering conçues pour les réseaux ad hoc et les réseaux de capteurs. Nous avons constaté que ces techniques présentent certaines limitations pour le passage à l'échelle et la gestion de la mobilité. Ainsi, les techniques de clustering conçues particulièrement pour les réseaux ad hoc, ne peuvent pas être appliquées aux réseaux de capteurs vu qu'ils ne prennent pas en considération les spécificités de ces réseaux, et la plupart des techniques de clustering conçues pour les réseaux de capteurs telles que LEACH et ses variantes génèrent des 1-clusters et ne supportent pas la mobilité. Par conséquent, le nombre de clusters générés pourrait être très grand ainsi que le moindre changement de

topologie pourrait déclencher le processus de restructuration de tout le réseau.

2.4 Autres techniques de diffusion

Dans cette partie, nous présentons une analyse détaillée d'autres techniques de diffusion conçues pour les réseaux ad hoc et de capteurs, afin qu'on puisse tirer profit de leurs avantages et éviter leurs limitations.

2.4.1 Diffusion par relais multipoints (MPR)

Pour que la diffusion soit efficace, il faudrait garantir que tous les voisins à deux sauts d'un nœud reçoivent correctement le message. Donc, si ce procédé est exécuté efficacement dans un réseau connexe alors tous les nœuds pourront être joints (accessibles) à partir du nœud source.

L'objectif des algorithmes dépendants de la source est de minimiser le nombre des nœuds qui réémettent le message de diffusion i.e. maximiser SRB. Ainsi, un sous-ensemble des voisins à un saut d'un nœud sera choisi pour joindre l'ensemble des voisins à deux sauts de ce nœud. Par suite, une fois que ce sous-ensemble est choisi, tout nœud émet son message de diffusion avec la liste de ses voisins à un saut qui devront réémettre le message de nouveau. On parle alors de nœuds relais. Le problème est de déterminer ce sous-ensemble de nœuds relais. En outre, trouver un sous-ensemble optimal des nœuds relais est considéré comme un problème NP-complet [88]. D'où, l'utilisation d'une approche heuristique pour déterminer ce sous-ensemble de telle sorte qu'il soit proche de la solution optimale.

Qayyum et al. [88] ont proposé le protocole de diffusion par relais multi-points (Multi-Point Relays MPR). Le principe de ce protocole est de déterminer un sous-ensemble de nœuds relais pour chaque nœud qui vont expédier des messages de diffusion pendant le processus d'inondation de telle sorte qu'ils joignent tous les voisins à deux sauts de ce nœud. Lors de la diffusion, chaque nœud ajoute à son message la liste de ses voisins à un saut (nœuds relais MPRs) qui devront relayer le message pour joindre l'ensemble de ses 2-voisins. Ce processus se répète de nœud en nœud jusqu'à ce que la diffusion engendre tout le réseau. Si un nœud reçoit plusieurs fois le même message à diffuser alors seule la première réception sera prise en compte et les réceptions postérieures seront tout simplement ignorées.

L'efficacité de ce protocole est fortement liée à la méthode heuristique utilisée pour déterminer le sous-ensemble de nœuds relais pour chaque nœud : plus la cardinalité de ce sous-ensemble est petite, plus la diffusion sera efficace. Dans ce protocole, chaque nœud possède une connaissance de la topologie réseau à deux sauts. Par suite, après chaque changement de la topologie locale, chaque nœud doit déterminer un ensemble minimal de nœuds relais pouvant joindre l'ensemble de ses voisins à deux sauts. Ce problème étant considéré comme NP-complet, les auteurs ont proposé une méthode heuristique pour trouver une bonne approximation du sous-ensemble minimal.

Soit un nœud *u*, $N_1(u)$ l'ensemble de ses voisins à un saut, et $N_2(u)$ l'ensemble de ses voisins se trouvant exactement à deux sauts. La détermination de l'ensemble des nœuds relais du nœud u, noté $MPR(u)$ par la méthode heuristique proposée par Qayyum et al. [88] se fait de la manière suivante :

Pseudo-code de l'algorithme de Qayyam et al. : Calcul de *MPR(u)*

---

1. Placer dans $MPR_1(u)$ l'ensemble des 2-voisins du nœud u,
   - $MPR_1(u) = N_2(u)$

   Initialiser $MPR(u)$ par l'ensemble vide
   - $MPR(u) = \oslash$

2. Ajouter à $MPR(u)$ tous les voisins imposés
   - Un voisin est dit imposé, s'il est le seul à pouvoir communiquer avec au moins un voisin à deux sauts isolé. Après l'ajout des voisins imposés, il faudrait retirer de $MPR_1(u)$ tous les voisins à deux sauts de u accessibles à partir des voisins imposés.

   Tant que $\exists w \mid w \in MPR_1(u) \ \wedge \ \exists! v \in N_1(u) \mid w \in N_1(v)$ **faire**

   $MPR(u) = MPR(u) \cup \{v\}$
   $MPR_1(u) = MPR_1(u)/N_1(v)$

   Fin Tant que

3. Tant que tous les 2-voisins de u ne sont pas couverts :
   - Choisir parmi les 1-voisins de u, le nœud qui n'est pas présent dans $MPR(u)$ et couvre plus de nœuds dans $MPR_1(u)$. En cas d'égalité de couverture de plusieurs nœuds, choisir celui qui a le plus grand degré pour les départager, ajouter le à $MPR(u)$, et retirer ses voisins de $MPR_1(u)$. Soit v ce nœud :

Tant que $MPR_1(u) \neq \oslash$ **faire**

– Choisir $v \in N_1(u)$:
$|N_1(v) \cap MPR_1(u)| = Max(|N_1(v) \cap MPR_1(u)| : v_i \in N_1(u))$

– $MPR(u) = MPR(u) \cup \{v\}$

– $MPR_1(u) = MPR_1(u)/N_1(v)$

Fin Tant que

Pour illustrer l'heuristique MPR, nous l'exécutons sur l'exemple présenté par la figure 2.3. Nous obtenons {*g*,*f*} comme ensemble de nœuds relais pour le nœud *d*.

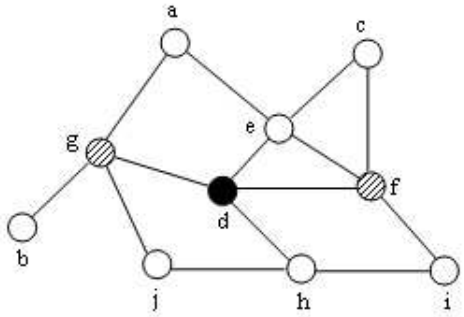


FIG. 2.3 – {*f*, g} nœuds relais du nœud *d*

Cette technique repose sur le concept clef des relais multipoints (MPRs). Les MPRs sont des nœuds sélectionnés pour relayer les messages à diffusion générale. Elle réduit de manière très significative l'overhead de la diffusion par rapport au mécanisme classique d'inondation i.e. inondation aveugle, où chaque nœud retransmet chaque message reçu pour la première fois. Dans le protocole de routage OLSR (Optimized Link State Routing Protocol) [53], l'information d'état de lien n'est produite que par des nœuds élus comme MPRs pour calculer les routes dans le réseau. OLSR fournit de bonnes performances pour les réseaux denses quand MPRs s'exécute à plein rendement dans ce contexte. Cependant, les performances de cette technique se dégradent quand nous l'appliquons dans un environnement non idéal où l'atténuation du signal pourrait y avoir une influence sur la réception avec succès d'un message.

2.4.2 Diffusion basée sur le mécanisme d'élimination de voisins (NES)

Dans le but d'éliminer les messages redondants au cours d'un processus de diffusion, Stojmenović et Seddigh [100, 101] ont proposé le mécanisme d'élimination de voisins (Neighbor

Elimination Scheme) NES. Ce mécanisme se base sur la politique *"Wait and See"*, où un nœud *u* ne relaye pas immédiatement un message reçu mais il attend un certain laps de temps timeout pendant lequel il écoute les communications de ses 1-voisins. Le temps d'attente pourrait être choisi d'une manière aléatoire, ou en fonction de la topologie du réseau. à l'expiration du timeout, s'il existe certains 1-voisins du nœud *u* qui n'ont toujours pas reçu le message, alors *u* le réémettra, sinon la réémission sera annulée. Toutefois, il est possible que certains 1-voisins de *u* aient reçu le message par l'intermédiaire de ses 2-voisins sans que le nœud *u* ne le sache. Dans ce cas, le nœud *u* réémettra le message malgré que cette réémission soit inutile.

En outre, pour optimiser le processus de diffusion, une solution a été proposée. Cette solution consiste à favoriser les nœuds ayant plus de 1-voisins non couverts pour relayer le message de diffusion, ce qui permet d'assurer la couverture de plus de nœuds par une seule transmission, et par suite minimiser le nombre de nœuds relais. Ainsi, le temps d'attente avant toute diffusion peut être choisi inversement proportionnel au nombre de voisins non encore couverts $t_u = 1/(Nb\_Voisins\_Non\_Couverts(u))$.

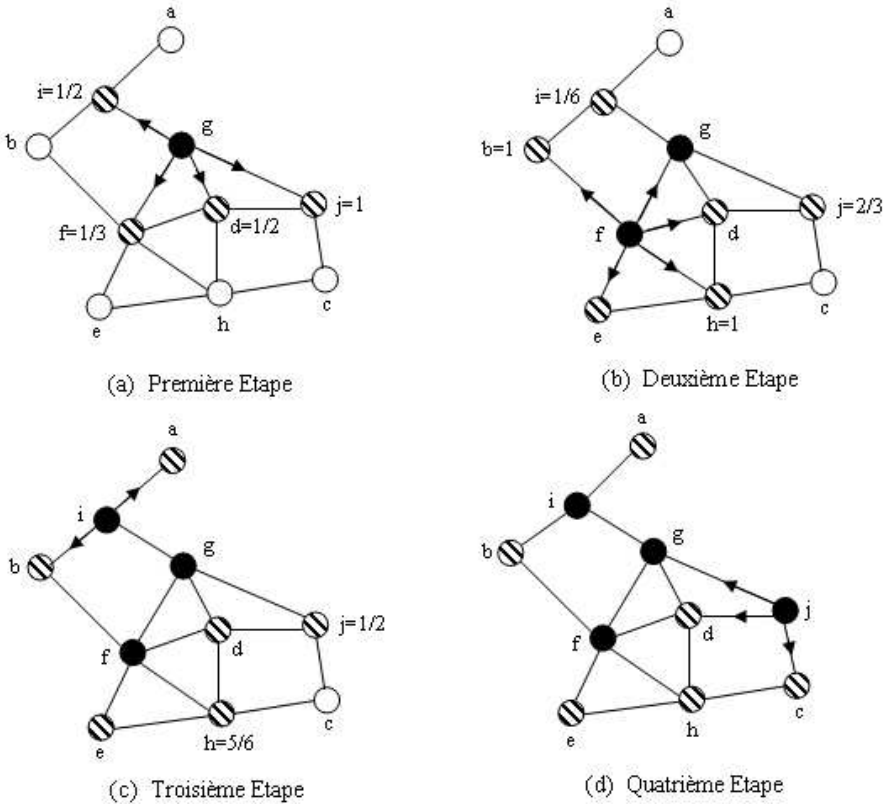


FIG. 2.4 – Application de NES

Pour illustrer le mécanisme NES, nous l'exécutons sur l'exemple présenté par la figure 2.4. Nous supposons que le temps d'attente est exprimé en unités de temps arbitraires, ainsi qu'il est fixé à $t_u = 1/(Nb\_Voisins\_Non\_Couverts(u))$ avant toute éventuelle réémission par un nœud *u*. A la première étape le nœud *g* (nœud source) envoie le message à ses voisins *d,f,i* et *j*. Ces derniers fixent leur temps d'attente respectivement à $t_d = 1/2$, $t_f = 1/3$, $t_i = 1/2$ et $t_j = 1$. La deuxième étape sera exécutée après un tiers de temps, le nœud *f* réémet le message à ses voisins *b, d, e, g* et *h*. Ces derniers calculent leurs temps d'attente avant toute éventuelle réémission. Le nœud *b* fixe son temps d'attente à $t_b = 1$ car il ne sait pas que le nœud *i* a été déjà couvert par le nœud *g*, de même le nœud *h* fixe son temps d'attente à $t_h = 1$, alors que les nœuds *d* et e ne font rien puisque tous leurs 1-voisins sont couverts. Les nœuds *i* et *j* mettent à jour leurs temps d'attente respectivement à $t_i = 1/6$ et $t_j = 2/3$. Puis, à la troisième étape le nóeud réémet le message à ces voisins *a* et *b*. Puisque tous les voisins de ces deux nœuds sont couverts, alors ils ne font rien. Ainsi, les nœuds *j* et *h* mettent à jour leurs temps d'attente respectivement à $t_h = 5/6$ et $t_j = 1/2$. Finalement, à la quatrième étape, le nœud *j* relaye le message après avoir consommé son temps d'attente ($1/2$ unités de temps). Le nœud *h* annule sa transmission puisque tous ses voisins sont couverts et la diffusion est terminée. Donc, parmi tous les nœuds du réseau seulement quatre nœuds ont réémi le message de diffusion.

Dans le mécanisme d'élimination de voisins (NES), nous avons remarqué qu'il y avait des réémissions inutiles dans le cas où il y avait des voisins qui étaient déjà couverts par des voisins à deux sauts. D'où, pour améliorer l'efficacité de NES, nous proposons de l'appliquer conjointement avec d'autres protocoles de diffusion ou permettre à NES d'avoir une connaissance de la topologie du réseau à deux sauts. D'autre part, ce mécanisme ne pourrait pas être utilisé dans les réseaux de capteurs conçus pour la surveillance des applications critiques parce qu'il génère une latence très importante.

2.4.3 Diffusion basée sur les ensembles dominants connexes

Un ensemble dominant V_d d'un graphe $G = (V, E)$ est un sous-ensemble de V ($V_d \subseteq V$) tel que tout nœud *u* du graphe G ($u \in V$) appartient à V_d ou il est voisin à un saut de certains nœuds de l'ensemble dominat V_d. Formellement, il est défini :

$$\forall u \in V,\ u \in V_d \ \ \vee \ \ \exists v \in V_d | u \in N(v)$$

La technique de diffusion basée sur les ensembles dominants peut être utilisée comme une épine dorsale (Backbone) pour les communications dans les réseaux ad hoc et de capteurs sans fil, puisque les nœuds de l'ensemble dominant peuvent couvrir l'ensemble des nœuds du réseau. Cependant, le processus de propagation d'un message de diffusion d'un nœud dominant à un autre reste un vrai handicap pour cette technique. Pour cela, il est nécessaire d'établir une certaine connexité entre les nœuds de l'ensemble dominant et par suite obtenir un ensemble dominant connexe.

Cette technique de diffusion est une solution possible pour les réseaux dans lesquels les nœuds restent toujours actifs. Ainsi, elle ne peut pas être adaptée aux réseaux de capteurs puisqu'elle exige que les nœuds appartenant à l'ensemble dominant, doivent garder toujours leurs équipements radio actifs utilisés pour relayer les messages de diffusion dans le réseau. Ces deux fonctions pénalisent grandement les nœuds de l'ensemble dominant et épuisent rapidement leurs batteries. Par conséquent, cette technique ne permet pas de prolonger la durée de vie des réseaux de capteurs et mener à une bonne fin la mission pour laquelle ces réseaux étaient déployés. Pour remédier à cette limitation, il est nécessaire de choisir les nœuds ayant plus de capacités dans leur voisinage pour supporter efficacement la diffusion comme nœuds dominants, d'autre part ces nœuds choisis doivent jouer ce rôle pour une durée limitée dans le temps.

L'efficacité de la diffusion dépend de l'algorithme utilisé pour déterminer l'ensemble dominant connexe : plus la cardinalité de cet ensemble est petite plus le processus de diffusion est efficace. Toutefois, trouver un ensemble dominant connexe de taille minimale (Minimum Connected Dominating Set MCDS) est un problème NP-complet [80] même dans les graphes à disque unitaire. Par conséquent, plusieurs heuristiques ont été proposées dans la littérature pour trouver un MCDS [29, 30, 105]. Deux stratégies sont utilisées pour construire un CDS :

- Construction d'un ensemble dominant, puis procédure de connexité,
- Construction d'un ensemble dominant connexe.

La première stratégie s'exécute en deux phases. La première phase consiste à construire un ensemble dominant par construction d'un ensemble indépendant maximal (MIS). Puis on connecte ses éléments en ajoutant d'autres nœuds dominants.

L'heuristique de Cardei et al.

Cardei et al. [21] ont proposé l'algorithme suivant pour générer un MIS :

- Un nœud est soit dominant, dominé, ordinaire, actif,
- Initialement, tous les nœuds du graphe G sont dans l'état ordinaire,
- Un nœud leader initie la construction et devient dominant,
  - Tout voisin d'un dominant devient dominé,
  - Tout voisin d'un dominé devient actif. Ces nœuds actifs concurrencent pour le statut dominant,
  - Le nœud ayant le plus fort degré dans le voisinage de ce nœud dominé devient dominant. Si deux ou plusieurs nœuds ont le même degré, celui qui a le plus petit ID devient dominant.
- Le processus se répète jusqu'à ce qu'il y aura dans le graphe G que des nœuds dominant et dominé.

Au cours de la première phase, chaque paire de nœuds du MIS sont séparés au moins de deux sauts .i.e. il n'existe pas de dominants adjacents. Ainsi, la deuxième phase consiste à interconnecter les nœuds dominants dans MIS en ajoutant d'autres nœuds dominants parmi les nœuds dominés. Cardei et al. utilisent des explorations itératives pour rechercher les dominés qui ont plus de voisins dominants. Puis, ilsles interconnectent via ces dominés qui deviennent dominants.

Heuristique proposée par Alzoubi et al.

Alzoubi et al. [6, 7] ont proposé un autre algorithme pour générer un CDS. Cet algorithme s'exécute en deux phases. Ces deux phases consistent à construire respectivement un MIS et un arbre dominant. Au cours de la première phase, un nœud racine *u* est choisi pour construire un arbre couvrant T. Après la construction de l'arbre T, tout nœud identifie son niveau dans l'arbre T comme suit : au début le nœud racine *u* diffuse son niveau 0 à ses voisins via un message d'annonce de niveau. Tout autre nœud, lors de la réception du message d'annonce de niveau de son père, calcule son propre niveau en incrémentant de 1 celui de son père, et le diffuse avec son propre niveau. En outre, chaque nœud enregistre le niveau de ses voisins. Le processus se poursuit jusqu'à l'arrivée aux nœuds feuilles. Ainsi, lorsqu'un nœud feuille a déterminé son niveau, il transmet un autre message appelé message LEVEL-MESSAGE à son père dans l'arbre

T. Puis, chaque nœud interne le retransmet à son pére .i.e. vers le haut une fois qu'il le reçoit de ses fils. Quand la racine reçoit le message LEVEL-MESSAGE, chaque nœud connaît son rang représenté par le couple $(Niveau, ID)$ et les rangs de ses voisins. Les rangs sont triés par ordre croissant en fonction des niveaux et des identifiants des nœuds. Ainsi, le nœud racine aura le plus petit rang parmi tous les nœuds du graphe. La coloration de chaque nœud est basée sur son rang. D'où, le MIS sera construit comme suit :

- Initialement, chaque nœud qui possède le plus petit rang parmi ses voisins est coloré en noir et se déclare dominant. Puis, il diffuse son statut via un message appelé "Message Dominant",
- Lorsqu'un nœud reçoit le message dominant pour la première fois, il sera coloré en gris, et se déclare dominé. Puis, il diffuse son statut par le biais du message dominé.
- Si un nœud reçoit les messages dominé de tous ses voisins avec des rangs inférieurs au sien, il sera marqué en noir et se déclarera dominant.
- Quand un nœud feuille est marqué, il transmet un message MIS-COMPLETE à son père dans l'arbre. Chaque nœud interne recevant le message MIS-COMPLETE, le transmet à son ascendant,
    - Le processus se répète jusqu'à l'aboutissement à la racine.

La seconde phase consiste à construire un arbre dominant (T_d). Tous les nœuds dans cet arbre forment un CDS. La construction de l'arbre T_d s'exécute comme suit :

- Initialement, l'arbre dominant est vide : $T_d = \oslash$;
- La racine est le premier nœud qui joint l'arbre.
- Quand chaque nœud noir joint l'arbre T_d, il envoie un message d'invitation à tous les nœuds noirs se trouvant à deux sauts de lui, et qui n'appartiennent pas à T_d pour joindre l'arbre T_d. Ce message d'invitation est relayé par les nœuds gris.
- Chaque nœud noir joindra l'arbre T_d quand il reçoit le message d'invitation pour la première fois, ainsi que le nœud gris qui a relayé le message.
- Le processus se répète jusqu'à ce que tous les nœuds noirs joignent l'arbre T_d.
- le CDS contient tous les nœuds qui appartiennent à l'arbre T_d.

Puisque la structure arbre couvrant est utilisée pour générer des CDS dans l'algorithme proposé Alzoubi et al., ceci rend difficile la maintenance des CDS. Par exemple, quand un nœud

éteint son lien radio ou se déplace loin de ses voisins, l'arbre se détruit et un nouvel arbre devrait être construit. D'autre part, les complexités en temps de convergence et en messages sont respectivement $\theta(n)$ et $\theta(n * log\ n)$.

L'heuristique de Guha et Khuller

Guha et Khuller [45] ont proposé une heuristique gloutonne centralisée pour déterminer l'ensemble dominant connexe. Cette heuristique permet de déterminer directement un CDS sans passer par un MIS pour déterminer un ensemble dominant puis ajouter d'autres nœuds dominants pour interconnecter les nœuds du MIS généré.

Etant donné un réseau sans fil représenté un graphe connexe non orienté $G = (V, E)$ et V_p l'ensemble dominant connexe à déterminer, l'exécution de l'heuristique proposée par Guha et Khuller est décrite comme suit :

Algorithme de Guha et Khuller : Détermination de CDS

---

1. Phase d'initialisation : Colorer tous les nœuds de l'ensemble V en blanc.
   - Les nœuds blancs ne sont ni dominants ni voisins d'aucun nœud dominant.
2. Choisir parmi les nœuds blancs, celui qui a le plus grand degré et le colorer en noir. Colorer tous ses voisins blancs en gris.
   - Les nœuds noirs sont dominants et les nœuds gris ont au moins un voisin dominant.
   - S'il y a égalité en degré entre plusieurs nœuds blancs, choisir celui qui a le plus petit identifiant pour les départager.
3. Tant qu'il existe des nœuds blancs dans le graphe,
   - Choisir un nœud gris ayant le plus grand nombre de voisins blancs. En cas d'égalité entre plusieurs nœuds gris, choisir toujours le nœud ayant le plus petit ID.
   - Colorer ce nœud en noir et ses voisins en gris.
   - Répéter cette étape jusqu'à ce que tous les nœuds dans le graphe soient noirs ou gris c'est-à-dire dominants ou ont au moins un voisin dominant.

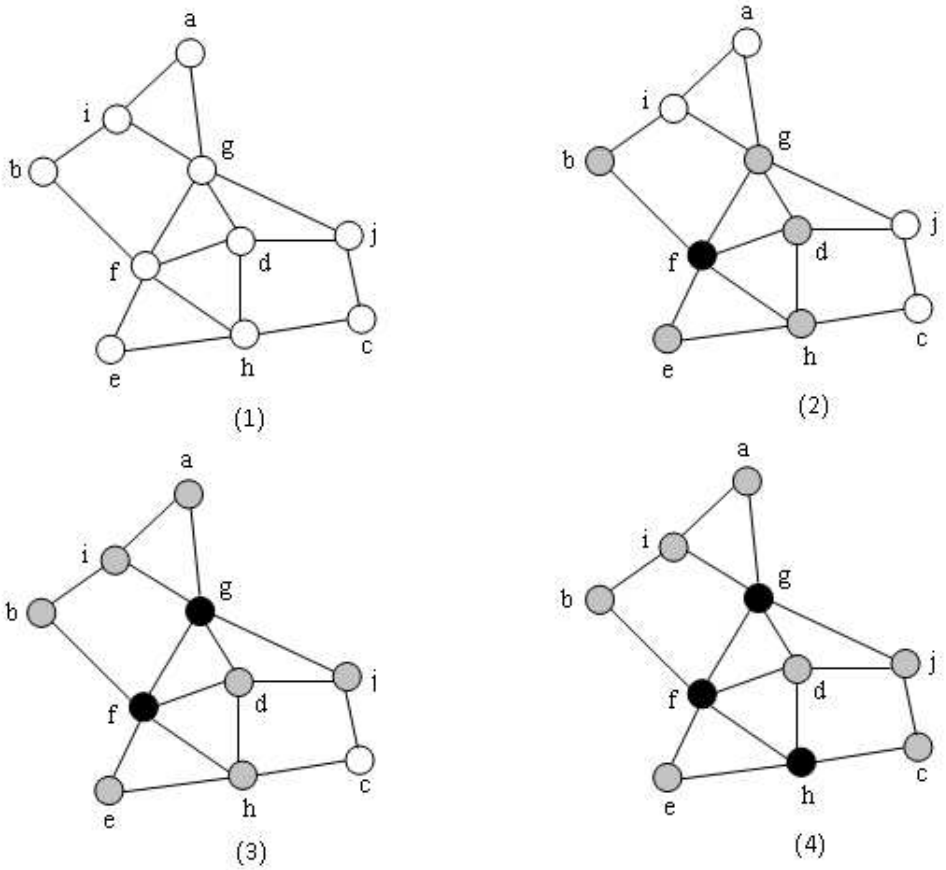


FIG. 2.5 – Application de l'heuristique de Guha et Khuller

Pour illustrer l'heuristique de Guha et Khuller, nous l'exécutons sur l'exemple présenté par la figure 2.5. Au début, tous les nœuds sont colorés en blanc. Les nœuds *f* et *g* ont les plus grands degrés dans le graphe. Le nœud *f* a le plus petit ID donc il est choisi comme nœud dominant. Par suite, il sera coloré en noir et ses voisins en gris. A la deuxième étape, nous remarquons que le nœud *g* a plus de voisins blancs que les autres nœuds gris. D'où, il est choisi comme nœud dominant (coloré en noir) et ses voisins blancs *a*, *i* et *j* sont colorés en gris. A la troisième étape, les nœuds *h* et *j* ont tous les deux un seul voisin blanc. Donc le nœud *h* sera choisi comme dominant puisqu'il a le plus petit ID. Finalement, aucun nœud blanc ne figure dans le graphe, le processus de détermination de l'ensemble dominant connexe est terminé. Ainsi, nous obtenons $V_p = \{d, f, g\}$ comme ensemble dominant connexe pour le graphe G.

L'heuristique de Guha et Khuller exige une reconnaissance globale de la topologie des réseaux. Or, cette exigence n'est pas pratique dans les réseaux denses et les réseaux mobiles.

2.4.4 Ensemble dominant basé sur les relais multipoints (DS-MPR)

Cette heuristique a été proposée par Adjih et al. [2] pour déterminer l'ensemble dominant connexe. Elle est décrite comme suit :

1. Chaque nœud calcule son ensemble des nœuds relais en se basant sur l'approche MPRs, puis il le transmet dans son voisinage.
2. Les nœuds ayant le plus petit ID dans leur voisinage se déclarent nœuds dominants.
3. Les nœuds choisis comme nœuds relais par leur voisin de plus petit ID se déclarent nœuds dominants.

Le critère ID reflète une priorité obsolète. Il n'est pas judicieux pour choisir les nœuds dominants dans les réseaux de capteurs. Nous proposons de choisir ces nœuds dominants en fonction de leur poids qui reflètent leur capacité d'être dominant et la topologie du réseau. Par exemple, il est commode de choisir le poids d'un nœud en fonction de sa k-densité et son énergie restante. D'où, nous remodelons l'heuristique de Adjih et al. de la manière suivante :

Heuristique remodelée de Adjih et al.

---

1. Chaque nœud *u* du graphe G calcule son poids en fonction de son degré et son énergie restante.

$$Weight(u) = \alpha \times P_{\textit{k-densité}}(u) + \beta \times P_{Energie}(u)$$

2. Chaque nœud *u* du graphe G calcule son ensemble de nœuds relais $MPR(u)$ et le transmet avec son poids dans son voisinage. $MPR(u)$ est calculé comme suit :
   - Placer dans $MPR_1(u)$ l'ensemble des 2-voisins de *u*,
     $MPR_1(u) = N_2(u)$
   - Initialiser $MPR(u)$ par l'ensemble vide,
     $MPR(u) = \oslash$
   - Ajouter à $MPR(u)$ tous les voisins imposés,
     Un 1-voisin de u est dit imposé, s'il est le seul à pouvoir communiquer avec au moins un 2-voisin de u. Après l'ajout des 1-voisins imposés à $MPR(u)$, il faudrait retirer de $MPR_1(u)$ tous les 2-voisins de *u* accessibles à partir des 1-voisins imposés.

     Tant que $\exists w \mid w \in MPR_1(u) \ \wedge \ \exists! v \in N_1(u) \mid w \in N_1(v)$ **faire**
     $MPR(u) = MPR(u) \cup \{v\}$

$MPR_1(u) = MPR_1(u)/N_1(v)$

Fin tant que

3. Tant que $MPR_1(u)$ n'est pas vide faire,
   - Ajouter à $MPR(u)$ le 1-voisin de *u* qui ne figure pas dans $MPR(u)$ et qui a le plus fort poids dans $N_1(u)$. En cas d'égalité, choisir comme nœud relais celui qui a plus d'énergie restante pour les départager.
   - Retirer de $MPR_1(u)$ tous les 2-voisins du nœud *u* qui sont accessibles à partir du nœud ajouté à $MPR(u)$.

   Tant que $MPR_1(u) \neq \oslash$ **faire**

   Choisir $v \in N_1(u)$ tel que :

   $Weight(v) = Max(Weight(v_i) \mid v_i \in N_1(u) \ \wedge \ v_i \notin MPR(u))$

   $MPR(u) = MPR(u) \cup \{v\}$

   $MPR_1(u) = MPR_1(u)/N_1(v)$

   Fin tant que

Un nœud est dominant si et seulement si :

- C'est le nœud de poids le plus fort dans son voisinage,
  ou
- Il appartient à l'ensemble des nœuds relais du nœud de plus fort poids.

Heuristique de Wu et Li

Avant de présenter l'heuristique proposée par Wu et Li [110], il est nécessaire de présenter une notion sur laquelle se base cette heuristique et les versions améliorées de cette heuristique :

- Un nœud est dit intermédiaire s'il a au moins deux voisins qui ne sont pas directement connectés. Formellement :

$$\exists v, w \in N_1(u) \mid (u, v) \in E \wedge (u, w) \in E \wedge (v, w) \notin E$$

Pour la formation de l'ensemble dominant des nœuds relais appelés aussi passerelles, Wu et Li ont pris en considération les contraintes suivantes :

1. Le processus de formation de CD devrait être distribué et simple, car il nécessite seulement une information localisée et un nombre réduit de messages échangés entre les nœuds voisins.
2. L'ensemble dominant résultant devrait être connecté et avoir une cardinalité minimale.
3. L'ensemble dominant devrait inclure tous les nœuds intermédiaires.

Le processus de marquage proposé par Wu et Li permet de marquer tous les nœuds dans un graphe connecté et non orienté $G = (V, E)$, où à tout nœud $u \in V$ est assigné une marque $m(u)$ qui prend deux valeurs distinctes T et F : si le nœud *u* est marqué alors $m(u) = T$ sinon $m(u) = F$. Ainsi, les nœuds marqués forment un ensemble dominant connexe. Le processus de marquage s'exécute comme suit :

Heuristique de Wu et Li

– Initialement, assigner la marque F à tout nœud $u \in V$

pour tout $u \in V$ **faire**
$m(u) = F$
fin pour

– Tout nœud $u \in V$ échange son ensemble de voisins $N_1(u)$ avec tous ses voisins,

– Tout nœud *u* assigne à sa marque la valeur T s'il existe parmi ses voisins au moins deux voisins qui ne sont pas directement connectés,

pour tout $u \in V$ **faire**
si $\exists v, w \in N_1(u) \mid \quad (u, v) \in E \ \wedge \ (u, w) \in E \ \wedge \ (v, w) \notin E$
alors $m(u) = T$
fsi
fin pour

/* Soit $V^{'}$ l'ensemble dominant connexe généré par le processus de marquage précédent.
/* Wu et Li ont impliqué deux règles pour réduire la cardinalité de $V^{'}$

1. Un nœud marqué peut devenir non marqué (non-passerelle) si tous ses voisins sont aussi voisins d'un nœud qui les couvre, et qui possède une priorité plus élevée. Soient deux nœuds $u, v \in V^{'}$,

si $(N_1[v] \subseteq N_1[u]) \ \wedge \ (clef(v) < clef(u))$
alors $m(v) = F$ i.e. changer la marque du nœud *v* et le supprimer de l'ensemble $V^{'}$,

$V' = V' - \{v\}$

fsi

2. Un nœud marqué peut devenir non marqué si tous ses voisins sont aussi voisins à deux nœuds directement connectés qui les couvrent et qui possèdent une priorité plus élevée. Soient les nœuds $u, v, w \in V'$ tels que $u, w \in N_1(v)$:

si $(N_1(v) \subseteq N_1(u) \cup N_1(w)) \ \wedge \ (clef(v) = Min\{clef(u), clef(v), clef(w)\})$

alors $m(v) = F$ i.e. changer la marque du nœud *v* et le supprimer de l'ensemble V',

$V' = V' - \{v\}$

fsi

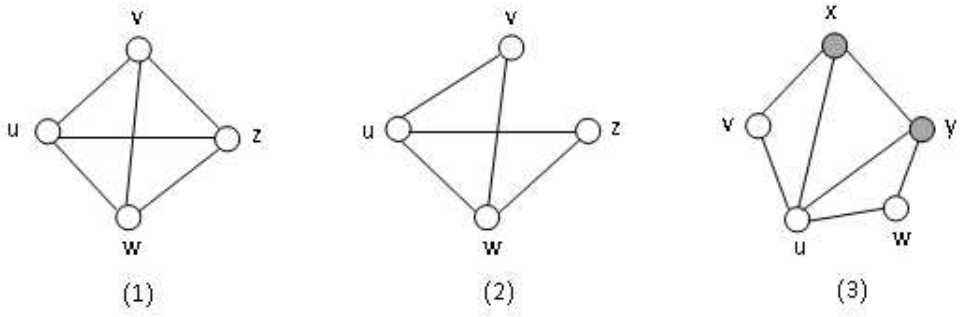


FIG. 2.6 – Application de l'heuristique de Wu et Li

- D'après la première règle, dans la figure 3.8.(2), le nœud *u* peut ne pas être marqué car les nœuds *v* et *z* sont couverts par le nœud *w* qui a une priorité plus élevée que *u*.
- D'après la deuxième règle, dans la figure 3.8.(3), le nœud *u* peut ne pas être marqué car ces voisins *v* et *w* sont couverts par deux voisins ayant une priorité plus élevée.

Les passerelles générées par ce mécanisme, sont utilisées comme des nœuds relais dans un processus de diffusion. Par exemple, le nœud *u* dans la figure 3.8.(1) ne pourra pas être marqué car tous ses voisins sont connectés entre eux, deux à deux. Evidemment, quand un de ses voisins relaye un message de diffusion, il sera reçu par tous les autres voisins de *u*. Par contre, dans la figure 3.8.(2), le nœud *u* pourrait être marqué comme nœud relais puisque ses voisins *v* et *z* ne sont pas directement connectés.

La notion de clef ou de poids d'un nœud désigne sa priorité pour être dans l'ensemble dominant. Elle peut être un identifiant, un degré, énergie restante,..., ou une combinaison de valeurs

associés aux nœuds. Pour simplifier, nous avons choisi dans cette présentation l'identifiant des nœuds comme clef. Or, dans les réseaux de capteurs mobiles, il est souhaitable de choisir sa valeur basée sur l'énergie restante et la k-densité des nœuds.

Vu que chaque nœud doit disposer d'une information positionnelle de ses voisins à deux sauts, alors la complexité est $\theta(\Delta^2)$ pour le processus du marquage (la première phase de l'algorithme) et $\theta(\Delta^3)$ pour l'application de la première et la deuxième règle, dans le but de réduire la cardinalité de l'ensemble dominant CDS généré au cours de la première étape.

Heuristique de Stojmenović et al.

Stojmenović et al. [101] ont proposé une version améliorée de l'heuristique de Wu et Li [110], dans le but de minimiser le nombre de nœuds relais par élimination des voisins. L'algorithme sous-jacent à l'heuristique proposée s'exécute comme suit : quand un nœud *u* reçoit un message de diffusion, au lieu de l'envoyer immédiatement, il devra attendre un certain temps (backoff) au cours duquel il supervise les transmissions de ses voisins. Ainsi, pour chaque voisin *v* qui a diffusé le message, le nœud *u* élimine $N_1(v)$ de $N_1(u)$ i.e. ses voisins communs avec le nœud *v*. Si $N_1(u)$ n'est pas vide après l'expiration de la période d'attente (backoff), alors le nœud *u* transmettra le message de diffusion, sinon il ignore le message de diffusion et se déclare comme nœud non dominant. Cet algorithme a été modifié et reformulé par Carle et Simplot-Ryl [22] de la manière suivante : au début, chaque nœud intermédiaire *u* supprime tous ses voisins à un saut ayant une priorité plus faible que lui, et construit son sous-graphe G_h formé de ses voisins ayant une priorité plus élevée. Si le sous-graphe généré G_h est vide ou non connexe, alors *u* est dominant. De même si G_h est connexe et il existe un voisin de *u* possédant une faible priorité qui n'est pas voisin à un saut d'un nœud de G_h, alors *u* est également dominant. Sinon *u* se déclare non dominant.

Cet algorithme pourrait être appliqué pour assurer la couverture de la zone surveillée et acheminer les données collectées par les capteurs vers la station de base dans les réseaux de capteurs. Il pourrait économiser grandement l'énergie dans les réseaux de capteurs puisqu'il pourrait impliquer seulement une fraction de capteurs dans le processus de diffusion d'une interrogation de la station de base (Request monitoring) ou d'acheminement de données collectées par les capteurs (event-driven) à la station de base. Ainsi, les autres capteurs pourront éteindre leurs liens radio. Cependant, si l'un des nœuds de l'ensemble dominant connexe cesse de fonctionner alors le réseau sera paralysé : la station de base ne pourra consulter les capteurs et les capteurs ne pourront transmettre leurs données à la station de base.

Heuristique de Dai et Wu

Dans le but de réduire la cardinalité de l'ensemble dominant connexe généré, Dai et Wu [33] ont généralisé les règles proposées par Wu et Li [110] par utilisation d'une règle générale d'ordre k appelée règle k. D'après cette règle, un nœud marqué *v* peut être non marqué si tous ses voisins sont aussi voisins de k nœuds connectés ayant une priorité supérieure, autrement, un nœud *v* est couvert par un sous-ensemble A_n de ses voisins à un saut :

si $((A_n$ est connecté$) \wedge (N_1(v) \subseteq N_1(A_n)) \wedge (\forall u \in A_n\, Clef(u) > Clef(v)))$

alors $m(v) = F$ i.e. changer la marque de *v* et le supprimer de V' ,

$V' = V' - \{v\}$

fsi

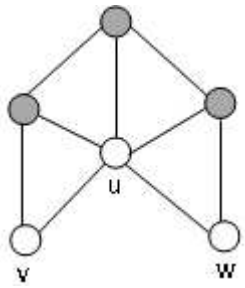


FIG. 2.7 – Application de l'heuristique de Dai et Wu

Par exemple, dans la figure 2.7, suivant la règle k, le nœud *u* peut ne pas être marqué, puisque ses voisins sont couverts par trois nœuds de priorité plus élevée (nœuds colorés en gris).

Toutefois, l'algorithme de Dai et Wu ne peut pas être bien adapté aux réseaux de capteurs sans fil mobiles puisqu'il exige une connaissance de l'information positionnelle à deux sauts pour chaque nœud, ainsi que le choix des nœuds relais qui dépend des voisins, et les nœuds de relais des voisins dépendent des voisins ainsi de suite ...Ceci augmente considérablement sa complexité en temps de convergence et en messages échangés entre nœuds.

Toutes les heuristiques que nous avons présentées, génèrent des CDS dans lesquels tout nœud dominé a un et un seul nœud dominant dans son voisinage, appelé aussi des 1-CDS. Or, dans les protocoles de routage et de diffusion basés sur les 1-CDS, seulement les nœuds dominants doivent maintenir l'information de routage dans une approche proactive et rechercher les routes

dans une approche réactive. Ainsi, si un des nœuds dominants formant le backbone virtuel du réseau cesse de fonctionner, alors la fidélité du routage n'est pas assurée entre tous les nœuds. Par conséquent, les 1-CDS ne peuvent pas être adaptés aux réseaux de capteurs parce qu'un capteur pourrait (être défaillant ou inhibé) facilement cesser de fonctionner en raison de l'épuisement de sa batterie ou de sa panne. En outre, ces réseaux sont sensés surveiller les événements pertinents dans une zone d'intérêt, donc il est nécessaire de maintenir un certain degré de redondance dans le backbone virtuel pour la tolérance aux fautes et la flexibilité de routage. Plusieurs travaux ont été présentés récemment pour traiter cette problématique [72, 73, 34]. Dans ces travaux, les auteurs ont proposé de générer des k-CDS dans lesquels un nœud dominé a k nœuds dominants dans son voisinage.

2.5 Conclusion

Dans ce chapitre, nous avons présenté une analyse détaillée des principales techniques de diffusion dans les réseaux ad hoc et de capteurs et les concepts sur lesquels se basent ces techniques. Au début, nous avons étudié plusieurs techniques de clustering conçues pour les réseaux ad hoc et les réseaux de capteurs. Nous avons constaté que le clustering imite l'architecture centralisée et tire profits de ses avantages dans les réseaux de petite ou moyenne taille. Il permet une réutilisation spatiale des fréquences radio pour minimiser les interférences et le surcoût du trafic de contrôle. Il est bien adapté aux réseaux de capteurs puisque ceux-ci disposent de faible mémoire pour stocker toute la topologie du réseau. Nous avons constaté également qu'il est nécessaire de bien choisir la ou les métrique(s) d'élection de cluster-head pour construire des clusters plus ou moins stables. Ces métriques doivent tenir compte de la capacité des nœuds pour jouer le rôle de cluster-head et de la topologie du réseau. Toutefois, la contrainte énergie est considérée comme une contrainte forte dans les réseaux de capteurs. De plus, pour assurer un équilibre de charge entre les nœuds, les clusters formés devront être homogènes en taille i.e. tous les clusters générés doivent contenir presque le même nombre de nœuds et rayon i.e. tous les membres d'un cluster doivent être à au plus k-sauts de leur cluster-head afin de limiter le trafic de contrôle induit par l'implémentation des fonctions de routage, découverte de services,.... Aussi le rôle du cluster-head devra être joué pour un intervalle de temps limité et le clustering devra être distribué.

Dans la deuxième partie de ce chapitre, nous avons présenté les algorithmes dépendants de la source et en particulier le protocole MPR. Puis, nous avons exposé la technique NES qui se base sur la politique "Wait and See" et nous avons constaté qu'elle génère une latence importante.

Par conséquent, elle n'est pas adéquate pour les réseaux de capteurs dédiés à des applications critiques i.e. les applications de surveillance orientées événements où le temps d'acquisition de données est un facteur déterminant pour prendre les mesures nécessaires. Ensuite, nous avons présenté des techniques basées sur le concept des ensembles dominants connexes (CDS), et nous avons remarqué que'elles ne sont pas adaptées aux réseaux de capteurs car elles impliquent un seul nœud dans son voisinage pour acheminer l'information dans le réseau. Or, les réseaux de capteurs sont tolérants aux fautes parce qu'à tout moment un capteur pourrait cesser de fonctionner normalement. D'où, la défaillance d'un capteur élu comme dominant, peut causer un goulet d'étranglement dans le réseau et ne pas permettre l'arrivée de l'information à la station de base. Enfin, nous avons vu les techniques qui combinent le concept MPR et DS et nous avons constaté qu'elles sont adaptées aux applications de couverture de zone. Cependant, elles génèrent une latence importante car les nœuds génèrent un temps d'attente (backoff) avant toute réémission pour éviter la redondance des réémissions. En outre l'ensemble des algorithmes que nous avons présentés, sont conçus pour un environnement idéal.

Cette étude nous a permis de tirer profits des algorithmes de diffusion pour proposer un protocole de diffusion dans un environnement réaliste et un algorithme d'auto-organisation efficaces pour les réseaux de capteurs mobiles. Ces deux algorithmes tiennent compte des spécificités de ces réseaux en termes d'énergie, puissance de calcul et radio de transmission. Ainsi, ils surpassent les limitations des algorithmes existants. La conception de ces deux algorithmes fait l'objet des chapitres suivants.

Chapitre 3

Notre contribution pour la diffusion dans un environnement réaliste

3.1 Introduction

Les protocoles de diffusion existants utilisent souvent le modèle du disque unitaire pour modéliser les communications radio entre les nœuds. Cependant, ce modèle ne peut pas être considéré comme un modèle réaliste puisqu'il suppose que les commuincations sont toujours faiables et que les messages sont toujours reçus sans erreur tant que la distance euclidienne séparant un nœud émetteur u d'un nœud récepteur v est inférieure ou égale au rayon de transmission R_c ($d(u, v) \leq R_c$). Donc, ce modèle ne prend pas en considération les fluctuations aléatoires du signal radio. Or, ces dernières peuvent avoir un impact significatif sur les transmissions à cause des erreurs qu'elles injectent dans les messages échangés entre les nœuds.

Dans ce chapitre, nous analysons le protocole de diffusion MPR dans un environnement réaliste pour montrer la dégradation des performances de ce protocole dans ce type d'environnement. Pour cela, nous utilisons le modèle Lognormal [89] pour illustrer l'impact des fluctuations du signal radio sur la probabilité d'une réception sans erreur d'un message par un nœud. Dans ce modèle, la probabilité de la réception correcte d'un message par un nœud est calculée en fonction de la distance euclidienne qui le sépare avec le nœud émetteur. Ce modèle est considéré plus réaliste que le modèle du disque unitaire.

Nous présentons une comparaison entre les probabilités d'une réception sans erreur dans le modèle du disque unitaire et le modèle Lognormal, et son influence sur le taux d'accessibilité

(reachability) dans les deux modèles.

3.2 Modèle Lognormal

Puisque le calcul de la probabilité de réception sans erreur d'un message par un nœud est influencé par plusieurs facteurs tels que la puissance de signal, la distance séparant l'émetteur du récepteur et la présence des obstacles, il pourrait être difficile d'obtenir une évaluation précise pour tous ces facteurs qui sont eux-mêmes sujettes à des erreurs. Pour cela, nous supposons que le signal transmis est reçu correctement quand la puissance du signal reçu par un nœud dépasse une certaine valeur seuil p_0 et diminue graduellement avec la distance. Par conséquent, la probabilité d'une réception sans erreur d'un message peut être calculée en fonction de la distance séparant deux nœuds. Ainsi, nous proposons d'utiliser le modèle Lognormal [89] pour évaluer cette probabilité. Ce modèle permet de générer un graphe pondéré dont le poids de chaque arête u, v est égal à la probabilité d'une réception sans erreur *p(u,v)* entre les nœuds *u* et *v*. Ainsi, pour évaluer cette probabilité, nous avons utilisé la fonction d'approximation *P(x)* [60] décrite comme suit :

$$P(x) = \begin{cases} 1 - \frac{(\frac{x}{R_c})^{2\alpha}}{2} & \text{si } 0 < x < R_c \\ \frac{(\frac{(2R_c - x}{R_c})^{2\alpha}}{2} & \text{si } R_c < x \leq 2R_c \\ 0 & \text{sinon} \end{cases} \tag{3.1}$$

Dans cette fonction, α représente le facteur d'atténuation du signal qui dépend de l'environnement et *x* la distance séparant l'émetteur du récepteur. Cette fonction suppose que la probabilité d'une réception sans erreur est égale à 0.5 quand la distance séparant deux nœuds est égale à R_c. La figure 3.1 illustre cette fonction pour $\alpha = 2$ et $R_c = 1$.

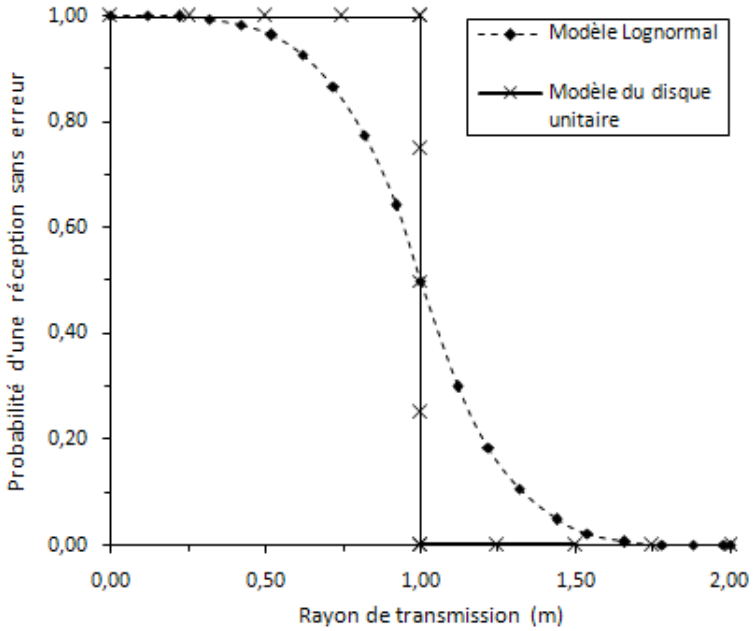


FIG. 3.1 – Probabilité d'une réception sans erreur en fonction de la distance séparant deux nœuds

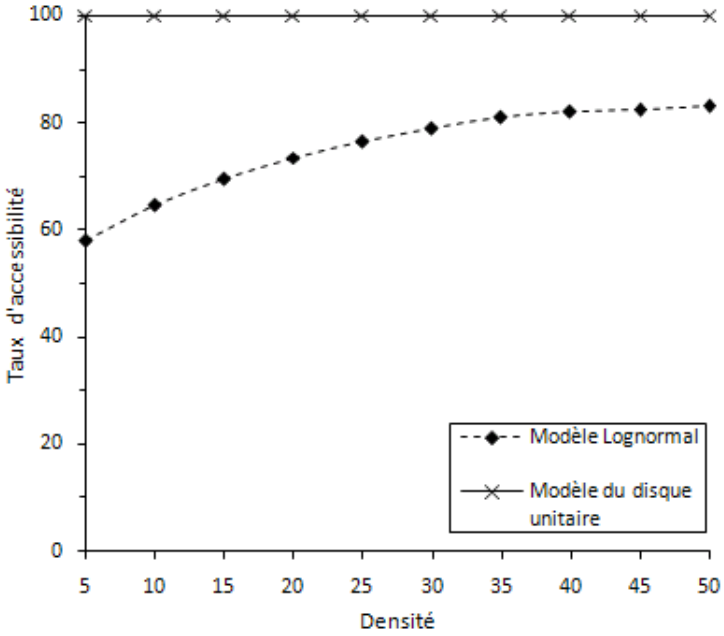


FIG. 3.2 – Comparaison des taux d'accessibilité dans les deux modèles

Nous avons créé un environnement de simulation incorporant le modèle Lognormal pour évaluer l'heuristique originale. La figure 3.2 présente une comparaison en termes d'accessibilité dans le modèle du disque unitaire et le modèle Lognormal. Ce graphe illustre l'impact des fluctuations du signal sur les performances du MPR. Nous avons constaté que tous les nœuds sont accessibles avec le modèle du disque unitaire alors qu'avec le modèle Lognormal, le taux d'accessibilité se dégrade à 58% quand la densité est 5, et à 83% quand la densité est 50. Ceci est dû au fait qu'il n'existe pas de garantie que tout nœud reçoit le message correctement.

3.3 Notre contribution

L'analyse précédente montre qu'il s'avère nécessaire d'améliorer les performances de la technique MPR, de sorte qu'elle s'adapte à un environnement réaliste. En effet, les auteurs dans [52] ont proposé trois heuristiques basées sur le protocole MPR pour la diffusion dans les réseaux sans fil. Cependant, les solutions qu'ils ont proposées, nécessitent des améliorations. Premièrement, les auteurs n'ont pas donné assez d'importance au problème de maximisation de la probabilité d'une réception sans erreur entre un nœud et ses 1-voisins qui relayent le message de diffusion. Deuxièmement, vu que la couverture de tous les 2-voisins d'un nœud n'est pas garantie, alors un seuil devrait être utilisé pour trouver l'ensemble des nœuds relais d'un nœud, dans le but d'atteindre un grand nombre de ses 2-voisins. Ainsi, il est nécessaire de spécifier un taux de couverture comme critère d'arrêt des algorithmes proposés. De ce fait, nous avons proposé deux versions améliorées de la technique MPR afin de dépasser ses limitations [66].

Nos contributions prennent en considération la présence de liens de communications incertains entre les nœuds. Ainsi, nous avons supposé qu'un nœud est considéré comme voisin d'un autre si la probabilité de recevoir des messages Hello entre eux est supérieure à un certain seuil p_0 et chaque nœud est capable d'évaluer la distance qui le sépare de ses voisins. Cette distance lui permet de calculer la probabilité d'une réception sans erreur. Par conséquent, afin d'éviter la sélection des nœuds incertains qui peuvent bloquer le processus de diffusion, un nœud émetteur devrait choisir comme nœuds relais parmi ses 1-voisins, ceux qui ont une probabilité élevée pour recevoir son message correctement. Ainsi, pour favoriser cette sélection, nous proposons d'élever au carré cette probabilité entre le nœud émetteur et ses 1-voisins.

Dans l'heuristique proposée, le nœud émetteur *u* choisit les nœuds voisins qui relayent le message de diffusion comme suit. Dans la première étape, nous supprimons seulement les nœuds

isolés de l'ensemble $MPR_1(u)$ au lieu d'enlever tous les voisins du nœud *v* choisi comme nœud relais. Dans la deuxième étape, nous favorisons le choix des 1-voisins de *u* qui peuvent recevoir le message sans erreur avec une probabilité élevée. En outre, pour garantir une réception correcte du message par les 2-voisins de *u*, nous proposons dans le deuxième algorithme de choisir parmi les 1-voisins de *u*, ceux qui peuvent transmettre le message correctement avec une grande probabilité aux 2-voisins de *u*.

Puisque nous avons supposé qu'il pourrait exister des liens de communications incertains, alors le réseau pourrait ne pas être connecté. Par conséquent, il nécessaire de modifier le critère d'arrêt de l'algorithme de diffusion. Ainsi, au lieu d'utiliser la diffusion du message dans tout le réseau comme critère d'arrêt, nous proposons d'utiliser la couverture de tous les nœuds, et si nécessaire une accessibilité qui dépasse 90% quand le temps de diffusion est expiré.

3.3.1 Première contribution : algorithme simple

Le but de cet algorithme est de choisir les meilleurs 1-voisins d'un nœud *u* comme nœuds relais. Au début, le nœud émetteur *u* vérifie si la probabilité de recevoir un message par son 1-voisin *v* dépasse un certain seuil p_0, *u* calculera le poids de ce nœud dénoté $W_u(v)$ comme suit :

$$W_u(v) = p(u,v)^2 \times \sum_{i=1}^{|MPR_1(u) \cap N_1(u)|} p(v, w_i) \tag{3.2}$$

Puis, le nœud *u* choisit comme nœud relais celui qui a le plus grand poids parmi ses 1-voisins. Nous résumons l'algorithme comme suit :

Pseudo-code de l'algorithme 1 : Calcul de $MPR(u)$

---

1. Initialement,
   - $MPR(u) = \oslash$
   - $MPR_1(u) = N_2(u)$
2. Trouver les nœuds isolés dans $MPR_1(u)$: les nœuds qui sont accessibles seulement par un et un seul 1-voisin de *u*, ajouter à $MPR(u)$ les 1-voisins de u qui permettent cet accès, et supprimer leurs nœuds isolés de $MPR_1(u)$.

   Tant que $\exists w \mid w \in MPR_1(u) \ \wedge \ \exists! v \in N_1(u) \mid w \in N_1(v)$ **faire**

   $- \ MPR(u) = MPR(u) \cup \{v\}$

$-\ MPR_1(u) = MPR_1(u)/\{w\}$

Fin tant que

3. **Tant que** $MPR_1(u) \neq \oslash$ **faire**

– $MPR_2(u) = MPR_1(u)$

– Calculer le poids de tout 1-voisin *v* de *u* qui a une probabilité de réception sans erreur dépassant p_0,

Pour tout nœud $v \in N_1(u)$ **faire**

si $p(u,v) > p_0$ **alors** calculer $W_u(v)$

– Choisir le nœud ayant le plus grand poids parmi les 1-voisins de *u*, soit *v* ce nœud.

Choisir $v \in N_1(u) : W_u(v) = Max(W_u(v_i) : v_i \in N_1(u))$

$-\ MPR(u) = MPR(u) \cup \{v\}$

$-\ MPR_1(u) = MPR_1(u)/(N_1(v) \cap MPR_1(u))$

– Vérifier s'il est possible de couvrir d'autres nœuds, sinon vérifier que le taux d'accessibilité dépasse 90% sinon diminuer le seuil p_0 d'une certaine valeur ξ.

si $MPR_2(u) = MPR_1(u)$

alors $RE = E[N_r]/N$

si $(RE < 0.9)$ **and** $(p_0 > p_{00})$

alors $p_0 = p_0 - \xi$

sinon Break ;

fsi

fsi

Fin Tant que

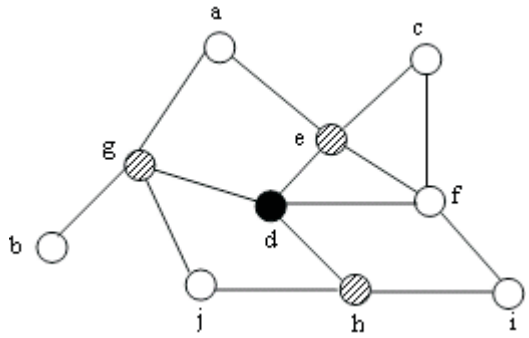


FIG. 3.3 – {*e,g,h*} nœuds relais du nœud *d*

TAB. 3.1 – Graphe pondéré représentant la probabilité d'une réception sans erreur entre les nœuds

arête	(a,e)	(a,g)	(b,g)	(c,e)	(c,f)	(d,e)	(d,f)	(d,g)	(d,h)	(e,f)	(f,i)	(g,i)	(h,i)	(h,j)
Poids	0.5	0.6	0.5	0.8	0.5	0.6	0.5	0.5	0.7	0.5	0.5	0.6	0.6	0.5

Le tableau 3.1 illustre un exemple d'un graphe où le poids d'une arête représente la probabilité d'une réception correcte entre deux nœuds. En appliquant notre approche sur la figure 3.3 avec $p_0 = 0.5$, nous obtenons $MPR(d) = \{e, g, h\}$ comme relais pour le nœud *d*.

3.3.2 La deuxième contribution : algorithme robuste

Nous avons développé un algorithme qui assure une certaine robustesse pour le processus de diffusion. Ainsi, au lieu d'utiliser un seul seuil pour le choix des nœuds relais, nous avons proposé d'impliquer un autre seuil p_1 pour garantir que les nœuds choisis comme relais transmettront le message de diffusion sans erreur à leurs voisins avec une probabilité élevée. En effet, le nœud émetteur évalue aussi le poids de connectivité $WW_u(v)$ entre ses 1-voisins et ses 2-voisins comme suit :

$$WW_u(v) = 1 - |MPR_1(u) \cap N_1(u)| \times \prod_{i=1}^{|MPR_1(u) \cap N_1(u)|} (1 - p(v, w_i)) \tag{3.3}$$

Puis, le nœud *u* vérifie si $WW_u(v)$ est supérieur au seuil p_1, si oui, il calcule le poids $W_u(v)$ de son 1-voisin *v*, et finalement, il choisit comme relais, le nœud ayant le plus grand poids parmi

ses 1-voisins. En cas d'égalité, il choisit le nœud ayant la plus grande probabilité pour recevoir un message sans erreur de sa part. Nous résumons cet algorithme comme suit :

Pseudo-code de l'algorithme 2 : Calcul de $MPR(u)$

---

1. $MPR(u) = \oslash, MPR_1(u) = N_2(u)$
2. **Tant que** $\exists w \mid w \in MPR_1(u) \ \wedge \ \exists! v \in N_1(u) \mid w \in N_1(v)$ **faire**
   - $MPR(u) = MPR(u) \cup \{v\}$
   - $MPR_1(u) = MPR_1(u)/\{w\}$

   Fin tant que
3. **Pour** tout nœud $v \in N_1(u)$ **faire**

   si $p(u,v) > p_0$

   alors calculer $WW_u(v)$

   si $WW_u(v) \geq p_1$

   alors $W_u(v) = p(u,v)^2 \times WW_u(v)$

   fsi

   fsi

   – Choisir le nœud ayant le plus grand poids parmi les 1-voisins de *u*,

   Choisir $v \in N_1(u) : \ W_u(v) = Max(W_u(v_i) : \ v_i \in N_1(u))$
   - $MPR(u) = MPR(u) \cup \{v\}$
   - $MPR_1(u) = MPR_1(u)/(N_1(v) \cap MPR_1(u))$
4. **si** $MPR_2(u) = MPR_1(u)$

   alors $RE = E[N_r]/N$

   si $(RE < 0.9)$ **and** $(p_0 > p_{00})$

   alors $p_0 = p_0 - \xi$

   sinon Break ;

   fsi

   fsi

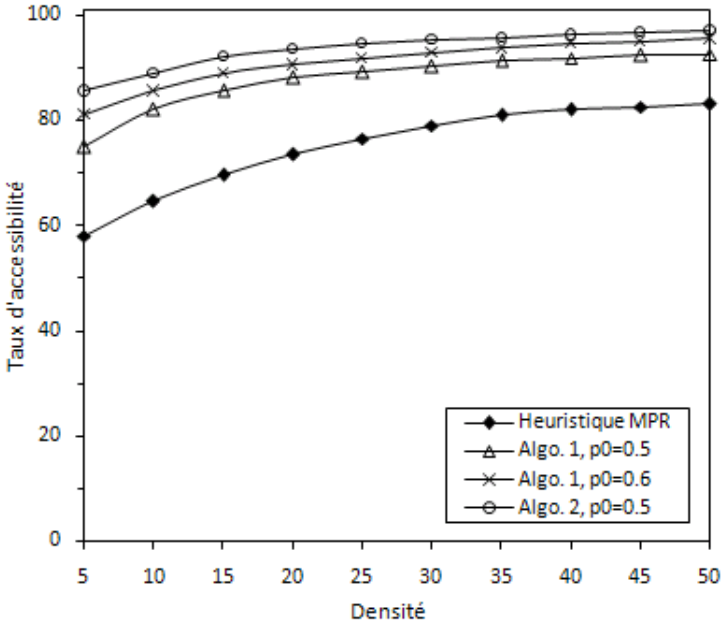


FIG. 3.4 – Comparaison des taux d'accessibilité

La figure 3.4 compare les taux de l'accessibilité observée quand nous appliquons les deux algorithmes proposés et l'heuristique originale dans le processus de diffusion. Nous remarquons que le taux d'accessibilité augmente quand la densité des nœuds augmente. Quand la densité dépasse 20, les algorithmes proposés donnent de bons résultats en particulier le deuxième algorithme. Ceci est dû à la robustesse de ce dernier, qui choisit les meilleurs nœuds comme nœuds relais, ainsi il évite les nœuds dont les liens de communications sont incertains.

3.4 Conclusion

Qayyum et al. [88] ont utilisé le modèle du disque unitaire pour concevoir le protocole de diffusion MPR. Cependant, l'évaluation des performances de ce protocole dans un environnement réaliste, a montré que les fluctuations du signal radio ont un impact négatif sur ces performances. De ce fait, nous avons proposé deux contributions pour améliorer les performances de MPR dans un environnement non idéal. Pour cela, nous avons utilisé le modèle lognormal et nous avons supposé que le choix des nœuds relais par le nœud source dépend de la qualité des liens entre ces nœuds et du degré des nœuds relais.

Chapitre 4

Algorithme d'auto-organisation des réseaux de capteurs mobiles

4.1 Introduction

Les réseaux de capteurs mobiles sont généralement déployés aléatoirement et en grand nombre dans une zone d'intérêt pour la surveillance d'un événement ou l'acquisition périodique des données environnementales. Ces capteurs s'auto-organisent et collaborent ensemble pour former un réseau de capteurs capable d'envoyer les données collectées à la station de base. Cependant, le déploiement des réseaux de capteurs exige des solutions pour un certain nombre de défis. Ces solutions dépendent des contraintes imposées par les capteurs : petite capacité de stockage, basse capacité de traitement, durée de vie limitée de la batterie et portée radio réduite, ainsi que leur topologie dynamique à cause de la mobilité des capteurs qui peuvent être placés sur des patients, des animaux, ou des véhicules. Ainsi, le traitement de ces défis exige une technique efficace permettant l'économie de l'énergie. L'architecture basée sur le clustering est considérée comme une approche prometteuse pour cette finalité. Par conséquent, nous devrons impliquer des paramètres déterminants pour produire un nombre réduit de clusters stables et équilibrés. La technique de clustering consiste à partitionner virtuellement le réseau en groupes appelés clusters.

Obtenir de meilleures performances avec les réseaux de capteurs résulte d'une auto-organisation efficace de ces réseaux. Ainsi, la proposition d'une technique d'auto-organisation s'avère nécessaire pour que ces réseaux puissent accomplir leurs missions sans que leurs performances soient dégradées. En plus, pour que cette technique d'auto-organisation soit réaliste,

elle doit tenir compte des spécificités des réseaux de capteurs et de leurs contraintes imposées. Le routage et la diffusion sont des éléments primordiaux dans l'acheminement de l'information d'un capteur à la station de base ou de la station de base à tous les capteurs, par exemple lors d'une mise à jour logicielle de tous les capteurs ou lorsque la station de base interroge tous les capteurs du réseau. Compte tenu du changement fréquent de la topologie du réseau à cause de la mobilité des capteurs, d'autres problèmes peuvent se poser en l'occurrence : la rupture d'une communication à cause de la défaillance d'un capteur, la perte de connectivité entre un capteur et ses voisins, ou l'acheminement d'informations redondantes vers la station de base. Dans le but de pallier au problème de mobilité et de changement fréquent de topologie, et pour apporter plus de stabilité à la structure réseau adoptée, nous proposons dans ce chapitre un algorithme d'auto-organisation basé sur les techniques de clustering pour les réseaux de capteurs mobiles. Ceci dans le but de garantir la stabilité de la topologie virtuelle du réseau et minimiser l'overhead de la diffusion dans les réseaux de capteurs mobiles et par conséquent améliorer leurs performances.

Dans ce chapitre, nous présentons tout d'abord quelques notions et définitions qui sont nécessaires à la présentation de l'algorithme proposé, puis nous exposons CSOS et son contexte d'exécution [69]. Ensuite, nous évaluons ses performances en termes de stabilité et de connectivité. Enfin, pour illustrer les atouts de CSOS, nous comparons les différents résultats obtenus à ceux obtenus par d'autres algorithmes.

4.2 Préliminaires

Avant d'aborder les détails techniques de l'algorithme CSOS, nous présentons quelques définitions et notations qui s'avèrent nécessaires à sa compréhension.

4.2.1 Modélisation du réseau

Considérons un réseau sans fil multi-saut où tous les nœuds coopèrent dans le but d'assurer des communications entre eux. Un tel réseau sans fil peut être représenté par un graphe connexe $G = (V, E, P)$; où V est l'ensemble des nœuds (hôtes, ou capteurs), $E \subseteq V^2$ l'ensemble des arcs reflétant les communications directes possibles entre les nœuds : la paire orientée (u, v) appartient à E si et seulement si le nœud *u* peut envoyer directement un message au nœud *v*, et nous disons que *v* est voisin de *u* ou également *u* couvre *v*. P est une fonction associant à chaque sommet $u \in V$ un poids $W(u) \in R$ représentant sa capacité d'être cluster-head. Les couples appartenant à E dépendent de la position des nœuds et de leur portée de transmission.

Nous supposons que tous les nœuds ont la même portée de transmission R_{Tx}, et tous les liens dans le réseau sont bidirectionnels, c'est-à-dire que si *u* est un voisin de *v* alors *v* est un voisin de *u*. Pour chaque nœud *u*, nous attribuons une valeur unique qui le caractérise, appelée identifiant et notée $Node_{Id}(u)$. La distance ***d(u,v)*** entre deux nœuds *u* et *v* est exprimée en nombre de sauts i.e. le nombre minimal de sauts qu'un message doit parcourir pour se rendre de *u* à *v* alors que ***dist(u,v)*** représente la distance euclidienne entre *u* et *v*.

Nous avons utilisé le modèle du disque unitaire (Unit Disc Graph) [31], avec R_{Tx} comme rayon de transmission. Ce modèle est très répondu pour modéliser les communications entre les nœuds dans les protocoles de diffusion conçus pour les réseaux ad hoc et de capteurs. Dans ce modèle, il est supposé que deux nœuds peuvent communiquer entre eux si la distance euclidienne ***dist(u,v)*** qui les sépare n'est pas supérieure à une portée de transmission donnée R_{Tx}, et les messages sont toujours reçus sans aucune erreur. D'où, l'ensemble E peut être défini comme suit :

$$E = \{(u, v) \in V^2 \mid \quad dist(u, v) \leq R_{Tx}\} \tag{4.1}$$

L'ensemble des voisins $N_1(u)$ d'un nœud *u* est défini par l'équation (4.2) et son degré $\delta_1(u)$ (ou 1-degré) représente le nombre de ses voisins c'est-à-dire le cardinal de l'ensemble $N_1(u)$.

$$N_1(u) = \{v \in V \mid \quad v \neq u \ \wedge \ (u, v) \in E\} \tag{4.2}$$

$$\delta_1(u) = |N_1(u)| \tag{4.3}$$

et l'ensemble étendu des voisins $N_1[u]$ est représenté par :

$$N_1[u] = N_1(u) \cup \{u\} \tag{4.4}$$

L'ensemble des voisins à deux sauts $N_2(u)$ d'un nœud *u* représente l'ensemble des nœuds qui sont les voisins des voisins du nœud *u* et qui ne sont pas les voisins de *u*. Il est défini comme suit :

$$N_2(u) = \{w \in V | \ (v, w) \in E : \quad w \neq u \ \wedge \ w \notin N_1(u) \ \wedge \ v \in N_1(u)\} \tag{4.5}$$

La réunion des ensembles $N_1(u)$ et $N_2(u)$ représente l'ensemble de tous les nœuds présents à une distance inférieure ou égale à deux sauts de *u*. Elle est notée soit $N_{12}(u, v)$ ou $N^2(u)$ et définie comme suit :

$$\begin{aligned} N_{12}(u) &= N_1(u) \cup N_2(u) \\ &= \{v \in V \mid v \neq u \ \wedge \ d(u,v) \leq 2\} \end{aligned} \tag{4.6}$$

D'une manière générale, l'ensemble des k-voisins $N^k(u)$ (ou voisins à k sauts) d'un nœud u contient tous les nœuds se retrouvant à une distance inférieure ou égale à k sauts de *u*. Il est défini par l'équation suivante :

$$N^k(u) = \{v \in V \mid v \neq u \ \wedge \ d(u,v) \leq k\} \tag{4.7}$$

et l'ensemble étendu des k-voisins $N^k(u)$ du nœud *u* par :

$$N_k[u] = N_k(u) \cup \{u\} \tag{4.8}$$

Le k-degré d'un nœud *u* noté $\delta_k(u)$ est le nombre de ses k-voisins (le cardinal de $N_k(u)$) :

$$\delta_k(u) = |N_k(u)| \tag{4.9}$$

La k-densité d'un nœud *u* représente le rapport entre le nombre de liens dans son k-voisinage (liens entre *u* et ses voisins et liens entre deux k-voisins de *u*) et le k-degré de *u* $\delta_k(u)$; formellement, elle est représentée par l'équation suivante :

$$\textit{k-densité}(u) = \frac{\mid (v,w) \in E : v,w \in N^k[u] \mid}{\delta_k(u)} \tag{4.10}$$

Cependant, dans notre contribution, nous se limitons seulement au calcul de la 2-densité des nœuds pour de ne pas affaiblir les performances de l'algorithme proposé. D'où, l'équation présentée dans 4.11 résulte de l'équation générale 4.10.

$$\textit{2-densité}(u) = \frac{\mid (v,w) \in E : v,w \in N^2[u] \mid}{\delta_2(u)} \tag{4.11}$$

Pour illustrer la métrique *k-densité*, nous proposons l'exemple représenté par la figure 4.1. Le tableau 4.1 illustre le calcul de la 2-densité des nœuds composant le réseau présenté dans la figure 4.1.

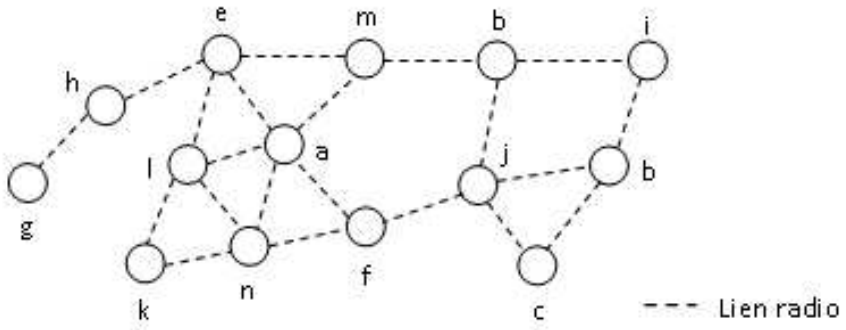


FIG. 4.1 – Exemple d'un réseau sans fil modélisé par un graphe non orienté

TAB. 4.1 – Calcul de la 2-densité

Nœud	a	b	c	d	e	f	g	h	i	j	k	l	m	n
1-degré	5	3	2	3	3	3	1	2	2	4	2	3	3	4
1-densité	1.60	1.00	1.66	1.33	1.33	1.33	1.00	1.00	1.00	1.25	1.66	1.66	1.33	1.75
2-degré	9	7	5	5	8	10	2	4	5	8	4	6	9	7
2-densité	1.55	1.50	1.40	1.40	1.37	1.60	1.00	1.25	1.40	1.50	1.75	1.60	1.44	1.57

Nous proposons de générer des clusters homogènes dont la taille est comprise entre deux seuils : $Thresh_{Lower}$ et $Thresh_{Upper}$. Ces seuils sont choisis arbitrairement ou dépendent de la topologie du réseau. Ainsi, si leurs valeurs dépendent de la topologie du réseau, ils sont calculés comme suit :

– Soit *u* le nœud ayant le nombre maximum de voisins à deux sauts

$$\delta_{12}(u) = Max(\delta_{12}(u_i) \ : \ u_i \in V) \tag{4.12}$$

– Soit *v* le nœud ayant le nombre minimal de voisins à deux sauts

$$\delta_{12}(v) = Min(\delta_{12}(v_i) \ : \ v_i \in V) \tag{4.13}$$

– Soit *Avg* le cardinal moyen des ensembles à deux sauts de tous les nœuds du réseau

$$Avg = \frac{\sum_{i=1}^{n} \delta_{12}(u_i)}{n} \tag{4.14}$$

où n est le nombre de nœuds dans le réseau

Ainsi, les deux seuils sont calculés :

$$Thresh_{Upper} = \frac{1}{2}(\delta_{12}(u) + Avg) \tag{4.15}$$

$$Thresh_{Lower} = \frac{1}{2}(\delta_{12}(v) + Avg) \tag{4.16}$$

Dans ce travail, nous supposons que chaque capteur possède une antenne omnidirectionnelle lui permettant par une transmission simple de couvrir tous les capteurs se trouvant dans son voisinage, et que les capteurs sont déployés dans un espace bidimensionnel. Nous considérons aussi que les capteurs sont stables durant une période raisonnable pendant l'exécution du processus de clustering, ainsi que chaque capteur a un poids générique et qu'il est capable de l'évaluer. Le poids représente la capacité d'un capteur pour être un cluster-head : un poids plus grand signifie une priorité plus élevée.

4.3 Principe de CSOS

Les contraintes imposées par les capteurs rendent la conception d'une technique efficace pour la diffusion et le prolongement de la durée de vie dans les réseaux de capteurs comme un vrai défi. Pour traiter ce défi, nous avons proposé une technique d'auto-organisation basée sur l'approche de clustering pour optimiser la consommation de l'énergie dans ces réseaux. Cette technique consiste à regrouper les nœuds proches géographiquement en clusters et elle implique des paramètres déterminants pour produire un nombre réduit de clusters homogènes en taille et en rayon, et qu'ils soient stables. Le poids de chaque capteur est calculé en fonction des paramètres suivants : 2-densité, énergie restante, et mobilité. Le capteur ayant le plus grand poids dans son 2-voisinage devient cluster-head. En outre, la taille des clusters générés est comprise entre deux seuils $Thresh_{Lower}$ et $Thresh_{Upper}$, qui représentent respectivement le nombre minimal et maximal de capteurs dans un cluster. Ces deux seuils sont choisis arbitrairement ou dépendent de la topologie du réseau. Dans un cluster, chaque capteur membre est au plus à deux sauts de son cluster-head correspondant, contrairement à LEACH [47]et sa variante LEACH-C [48], qui permettent seulement des 1-clusters d'être construits.

Dans les approches heuristiques proposées pour les réseaux de capteurs, basées sur la technique de clustering, les membres d'un cluster ne transmettent pas leurs données collectées direc-

tement à la station de base mais à leur cluster-head correspondant. En conséquence, les cluster-heads sont responsables pour coordonner les membres du cluster, agréger leurs données capturées, et de les transmettre à une station de base distante, directement ou via un mode de transmission multi-sauts. De ce fait, puisque les cluster-heads reçoivent plus de paquets et consomment plus d'énergie pour les transmettre avec une longue portée, ils sont donc ceux dont l'énergie sera épuisée le plus rapidement dans les clusters s'ils sont élus pour une longue période. Par conséquent, une technique de clustering devrait éviter une élection fixe des cluster-heads, parce que ces derniers sont contraints par l'énergie, et peuvent rapidement épuiser leurs batteries à cause de leur forte utilisation. Ainsi, cela peut causer des goulets d'étranglement dans les clusters et déclencher par suite le processus de réélection des cluster-heads de nouveau. Pour cela, nous avons prévu dans notre contribution que le processus d'élection des cluster-heads soit périodique après l'écoulement d'une certaine période Δt pour distribuer équitablement la consommation de l'énergie parmi les capteurs durant la durée de vie du réseau.

Dans la technique proposée, nous supposons que les capteurs ont une connaissance topologique à deux sauts, et opèrent d'une manière asynchrone et sans contrôle centralisé. Chaque capteur utilise le critère poids pour élire un cluster-head dans son 2-voisinage. Ce poids est calculé en fonction de la k-densité, l'énergie résiduelle et la mobilité, puis diffusé dans le 2-voisinage de chaque capteur. Ainsi, le capteur ayant le plus grand poids parmi ses 2-voisins qui ne sont pas encore affiliés à d'autres clusters est choisi comme cluster-head pour la période courante.

4.3.1 Formation des clusters

Le processus de formation de cluster consiste à partitionner virtuellement le réseau en groupant l'ensemble des capteurs dans des groupes disjoints appelés clusters. De ce fait, il donne au réseau une organisation hiérarchique. Chaque cluster a un cluster-head qui est choisi dans son 2-voisinage selon la valeur de son poids qui est une combinaison de : k-densité, énergie résiduelle, et mobilité, comme présenté par l'équation (4.17). Le coefficient de chaque paramètre (α, β, γ) peut être choisi selon l'application. Par exemple, dans les réseaux de capteurs déployés dans des zones hostiles et inaccessibles, nous favorisons le paramètre énergie sur les autres pour assurer une longue durée de vie au réseau. Par contre, dans les réseaux de capteurs à forte mobilité, nous favorisons les paramètres k-densité et mobilité pour garantir la stabilité de l'architecture hiérarchique générée et éviter par la suite les réactions en chaîne résultantes de l'instabilité de la structure.

$$Weight(u) = \alpha * \textit{2-densité(u)} + \beta * \textit{Res-Energie}(u) + \gamma * \textit{Mobilité(u)} \quad \alpha + \beta + \gamma = 1 \tag{4.17}$$

Puisque le cluster-head est responsable de la réalisation de plusieurs tâches telles que : coordonner entre les membres du cluster, transmettre des données collectées par les membres à une station de base distante, et contrôler son propre cluster, nous proposons d'exécuter périodiquement le processus d'élection des cluster-heads pour ne pas épuiser leurs batteries rapidement. De ce fait, nous cherchons à générer des clusters homogènes en taille et rayon. Pour cela, nous avons utilisé deux seuils $Thresh_{Lower}$ et $Thresh_{Upper}$ d'une part pour distribuer équitablement la consommation d'énergie entre les cluster-heads et d'autre part pour générer un nombre réduit de clusters. En effet, si le nombre des cluster-heads est grand alors la consommation d'énergie reste presque la même que dans une architecture à plat, de même si le nombre de clusters est petit, les cluster-heads vont se trouver loin de la station de base et par suite ils déploieront plus d'énergie pour transmettre leurs données. En outre, dans les clusters générés, les membres sont à deux sauts de leur cluster-head correspondant.

Dans CSOS, chaque capteur est identifié par un vecteur d'état comme suit : $(Node_{Id}, Node_{CH}, Weight, Hop, Size, Thresh_{Lower}, Thresh_{Upper})$ où $Node_{Id}$ est l'identifiant du capteur, $Node_{CH}$ représente l'identifiant de son cluster-head, Hop indique le nombre de sauts le séparant de son cluster-head respective, et $Size$ représente la taille du cluster auquel il appartient. Chaque capteur est responsable du maintien d'une table appelée "$Table_{Cluster}$" pour stocker les informations des membres locaux du cluster. Le format de cette table est défini ainsi $Table_{Cluster}(Node_{Id}, Node_{CH}, Weight)$. Les capteurs collaborent entre eux pour construire et mettre à jour cette table en utilisant des messages *"Hello"*. Nous avons proposé d'utiliser les messages *"Hello"* pour alléger l'overhead de la diffusion et ne pas dégrader les performances de la technique CSOS. En plus, chaque cluster-head a une table appelée "$Table_{CH}$", dans laquelle l'information des cluster-heads est stockée. Le format de cette table est défini ainsi $Table_{CH}(Node_{CH}, Weight)$.

Le processus de formation des clusters est exécuté en trois phases : set-up, ré-affiliation, et état permanent (steady-state) comme illustré par la figure 4.2.

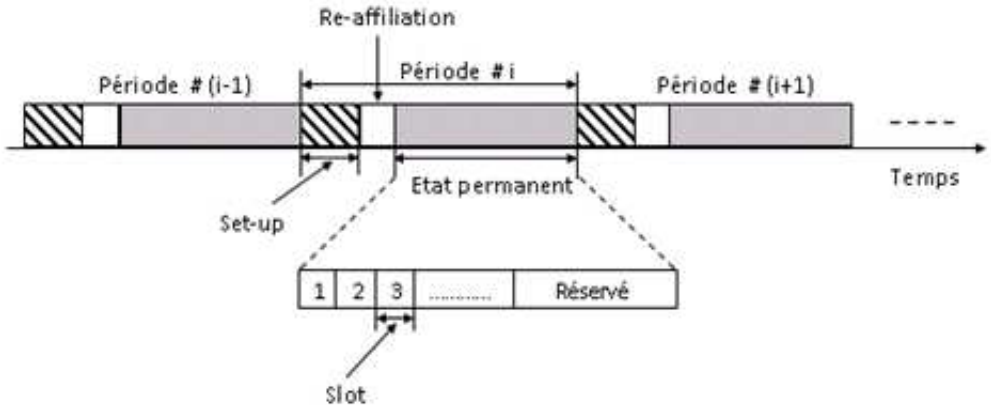


FIG. 4.2 – Les phases de formation de clusters

La phase set-up (figure 4.3)

Au début de chaque période, chaque capteur calcule son poids et génère un message *"Hello"* avec deux champs supplémentaires en plus à son contenu habituel : $Weight$ et $Node_{CH}$, où $Weight$ contient son poids calculé selon l'équation (4.17) et $Node_{CH}$ est initialisé à zéro .i.e. $Node_{CH} = 0$. Puis, il le diffuse à ses 2-voisins et écoute les messages *"Hello"* de ses 2-voisins. Le capteur ayant le plus grand poids parmi ses 2-voisins non encore affiliés est élu cluster-head *(*CH) pour la période courante. Ce dernier met à jour son vecteur d'état en assignant la valeur de son identifiant $Node_{Id}$ à $Node_{CH}$ et attribue respectivement à Hop et $Size$ les valeurs 0 et 1. Puis, il diffuse le message d'avertissement *(ADV_CH)* contenant son vecteur d'état dans son 2-voisinage pour les inviter à le rejoindre comme illustré par la figure 4.3. Tout capteur de ses 1-voisins qui reçoit le message de diffusion et qui n'est pas encore affilié à d'autres clusters, et dont le poids est inférieur à celui de *CH*, transmet le message *REQ_JOIN* au *CH* pour le rejoindre. Le cluster-head correspondant *CH* vérifie si la taille de son cluster n'a pas encore atteint $Thresh_{Upper}$, il transmettra un message *ACCEPT_CH* à ce capteur demandant l'affiliation ; sinon il ignorera le message de demande d'adhésion *REQ_JOIN*. Si l'adhésion d'un nœud est acceptée alors *CH* incrémente de 1 la valeur du paramètre $Size$ et le nouveau nœud affilié assigne respectivement aux paramètres Hop et $Node_{CH}$ les valeurs 1 et $Node_{Id}$ de son cluster-head *(*CH). Puis, les nouveaux capteurs affiliés transmettront de nouveau le message reçu avec la même puissance de transmission à leurs voisins. De la même manière, chaque nœud appartenant aux 2-voisins de *CH* $(N_2(Node_{CH}))$, qui n'est pas encore affilié et dont le poids est inférieur à celui du *CH*, transmet un message *REQ_JOIN* au *CH* correspondant. Ainsi, *CH* vérifie si la valeur du paramètre $Size$ est toujours inférieure à $Thresh_{Upper}$, il transmettra *ACCEPT_CH* un message à ce nœud et il

mettra à jour son vecteur d'état, sinon il ignorera le message de demande d'adhésion. Après la fin de l'exécution de cette phase, tout nœud connaît à quel cluster il appartient et quel est son cluster-head correspondant.

FIG. 4.3 – Procédure d'affiliation d'un nœud *u* à un cluster

Pseudo-code de la phase set-up

1. Assigner des valeurs aux coefficients α, β et γ
2. Initialiser $Time_{Cluster}$ /* Temps nécessaire à l'exécution de la phase set-up */
3. **pour** tout nœud $u \in G$ **faire** /* Initialiser le vecteur d'état de tous les nœuds */
   - Attribuer un identifiant $Node_{id}$ au nœud *u*
   - $Node_{CH} = 0$;
   - $Size = 0$;
   - $Hop = 0$;

   fin pour
4. /* Tout nœud $u \in G$ calcule son poids en fonction de sa 2-densité, son énergie restante, et sa mobilité */

   pour tout nœud $u \in G$ **faire**

   $Weight(u) = \alpha * \textit{2-densité}(u) + \beta *$ Res-Energie$(u) + \gamma * \textit{Mobilité}(u)$
5. Répéter
   - Tout nœud *u* diffuse un message *"Hello"* dans son 2- voisinage ;

     /* Choisir le nœud ayant le plus grand poids dans son 2-voisinage comme cluster-head */
   - Choisir $v \in N_{12}[u] : \ Weight(v) = Max(Weight(w) \mid \ w \in N_{12}[u])$;
   - *Update_CH_State(CH)* /* Mettre à jour le vecteur d'état du cluster-head élu *(CH)* */

     $CH \rightarrow Node_{CH} = CH \rightarrow Node_{id}$;

     $CH \rightarrow Size = 1$;

     $CH \rightarrow Hop = 0$;

– Envoi du message *ADV_CH* par *CH* à ses 2-voisins ($N_{12}[CH]$)

– **si** (un nœud $u \in N_{12}[CH]$ reçoit le message *ADV_CH*) && $(u \rightarrow Node_{CH}! = 0)$

alors

- *u* envoie un message *REQ_JOIN* à CH /* *u* demande d'affilier à *CH* */

  /* *CH* vérifie si la taille du cluster n'atteint pas $Thresh_{Upper}$ */

- **si** $(CH \rightarrow Size < Thresh_{Upper})$

  alors

  - *CH* envoie le message *ACCEPT_CH* au nœud *u* ;
  - *CH* exécute la procédure d'adhésion

    $CH \rightarrow Size = CH \rightarrow Size + 1$;
  - *u* exécute la procédure d'adhésion

    $u \rightarrow Node_{CH} = CH \rightarrow Node_{CH}$
  - **si** $(u \in N_1[CH])$

    alors $u \rightarrow Hop = 1$

    sinon $u \rightarrow Hop = 2$

    fsi

  fsi

fsi

6. *Update(*$Table_{Cluster}$*)* ;
7. **Jusqu'à ce que** $Expired(Time_{Cluster})$;

La phase de ré-affiliation

Durant la phase set-up, il peut ne pas être possible que tous les clusters atteignent le seuil $Thresh_{Upper}$. En outre, il est possible qu'il y ait des clusters dont la taille inférieure à $Thresh_{Lower}$ soient créés, puisqu'il n'y a aucune contrainte imposée concernant la génération de ce type de clusters dans la première phase. De ce fait, dans la phase de ré-affiliation, nous proposons de ré-affilier les nœuds appartenant aux clusters qui n'ont pas atteint la taille $Thresh_{Lower}$ à ceux qui n'ont pas atteint $Thresh_{Upper}$ afin de réorganiser les clusters formés, réduire leur nombre et obtenir des clusters homogènes en taille.

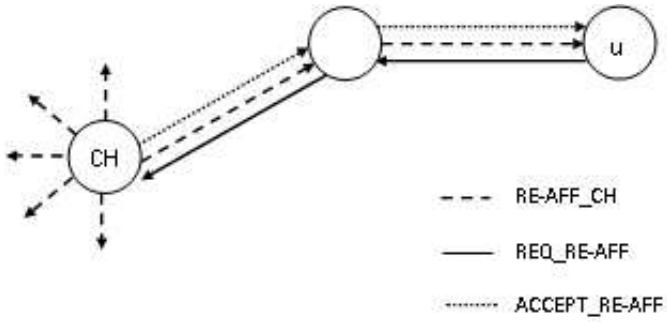


FIG. 4.4 – Procédure de ré-affiliation d'un nœud *u* à un cluster

L'exécution de cette phase procède de la façon suivante (i.e figure 4.4) : les cluster-heads qui appartiennent aux clusters dont la taille est strictement inférieure à $Thresh_{Upper}$ et plus grande que $Thresh_{Lower}$ émettent un nouveau message appelé *RE-AFF_CH* pour demander aux nœuds appartenant aux clusters dont la taille est inférieure à $Thresh_{Lower}$ de les rejoindre. Tout nœud qui reçoit le message et qui appartient à un cluster dont la taille est inférieure à $Thresh_{Lower}$ devrait être ré-affilié au cluster-head le plus proche en se basant sur la puissance du signal reçu. Or, cette puissance du signal est calculée en fonction de la distance séparant ce nœud de *CH*. Finalement, chaque cluster-head crée un calendrier (schedule) dans lequel des intervalles de temps appelés slots sont assignés pour la communication intra-cluster, l'agrégation de données, la communication inter-cluster et les opérations de maintenance. Ce mode de gestion permet aux capteurs de rester dans l'état sommeil aussi longtemps que possible et évite les collisions.

Pseudo-code de la phase de ré-affiliation

/* Tous les cluster-heads dont la taille du cluster est inférieure au seuil $Thresh_{Upper}$ envoient un message de demande de ré-affiliation aux nœuds appartenant à des clusters dont la taille ne dépasse pas $Thresh_{Lower}$ */

1. **si** $(Size(Cluster_{num\_cl}, CH) < Thresh_{Upper})$

   alors

   – CH envoie un message $(RE - AFF\_CH)$ à ses 2-voisins,

   – **si** (un nœud $u \in N_1[CH]$ reçoit le message) &&
   (*u* appartient à un cluster dont la taille est inférieure à $Thresh_{Lower}$)

   alors

   - *u* envoie un message $(REQ\_RE - AFF)$ au CH le plus proche

- **si** $(Size(Cluster_{num\_cl}, CH) < Thresh_{Upper})$

  alors

  - CH envoie un message $(ACCEPT\_RE - AFF)$ à u
  - CH met à jour son vecteur d'état

    $$CH \rightarrow Size = CH \rightarrow Size + 1$$

  - *u* met à jour son vecteur d'état

    $$u \rightarrow Node_{CH} = CH \rightarrow Node_{CH}$$

  - **si** $(u \in N_1[CH])$

    alors $u \rightarrow Hop = 1$

    sinon $u \rightarrow Hop = 2$

    fsi

  fsi

fsi

fsi

2. Update($Table_{Cluster}$)

Après l'exécution de la phase set-up et ré-affiliation, il y a formation des 2-clusters i.e. les clusters dont les membres sont au plus de deux sauts de leur cluster-head correspondant, et chaque CH connaît ses 2-voisins qui constituent les membres du cluster. Ensuite, chaque cluster-head enregistre toutes les informations de ses membres dans la table "$Table\_Cluster$", et réciproquement tous les nœuds membres enregistrent le $Node_{Id}$ de leur cluster-head correspondant. De cette façon, les tables de routage seront réduites et par conséquent la latence sera réduite et l'acheminement de l'information d'un capteur à la station base ne prendra pas beaucoup de temps.

Vu que l'environnement considéré est dynamique alors la topologie du réseau change à cause des déplacements des nœuds capteurs. De ce fait, il est nécessaire d'implémenter des fonctions de maintenance légères pour ne pas dégrader la technique proposée de ses performances. En outre, nous avons évité l'élection des cluster-heads pour une longue durée afin de ne pas les pénaliser en consommation d'énergie. Ainsi, l'exécution du processus de clustering s'effectue d'une manière périodique après chaque Δt afin de distribuer équitablement la consommation d'énergie entre l'ensemble des nœuds composant le réseau. Le choix de la période Δt dépend de la mobilité des nœuds. Si la mobilité des nœuds est faible, la période de Δt sera choisie plus longue, alors que

si les nœuds sont fortement mobiles, Δt sera choisie plus courte. Ceci dans le but de minimiser les réélections fréquentes des cluster-heads.

4.3.2 La maintenance des clusters

Dans notre contribution, le processus de maintenance des clusters devrait être déclenché quand un cluster-head épuise sa batterie ou migre vers d'autres clusters. Le processus de réélection du nouveau cluster-head concerne seulement les clusters ayant perdu leurs cluster-heads et le futur cluster-head est choisi parmi les membres du cluster. Nous avons adopté cette solution pour ne pas affaiblir les performances de cette technique et éviter les réactions en chaîne qui peuvent se produire pendant le déclenchement du processus de clustering. En outre, le processus de maintenance est effectué de la même manière que la phase set-up, où un nœud aléatoire parmi les membres du cluster initie le processus de clustering.

4.3.3 Métriques utilisées pour l'élection des cluster-heads

L'élection des cluster-heads est basée sur les poids des nœuds. Le poids d'un nœud est une combinaison des métriques suivantes : 2-densité, énergie restante, et mobilité.

$$Weight(u) = \alpha * \textit{2-densité}(u) + \beta * \textit{Res-Energie}(u) + \gamma * \textit{Mobilité}(u)$$
$$\text{avec} \quad \alpha + \beta + \gamma = 1$$

Calcul de la 2-densité

Le calcul de la 2-densité d'un nœud *u* nécessite la connaissance de l'information positionnelle de ses 2-voisins i.e. ses 2-voisins et les liens entre ses 2-voisins, et non pas la connaissance de l'information topologique de ses 2-voisins. La figure 4.5 illustre la différence entre une information positionnelle et une information topologique d'un nœud. Formellement, la 2-densité d'un nœud *u* est calculée selon l'équation suivante :

$$\textit{2-densité(u)} = \frac{|(v,w) \in E : \ v,w \in N_{12}[u]|}{\delta_{12}(u)} \tag{4.18}$$

où

$$N_{12}[u] = \{v \in E \ : \ d(u,v) \leq 2\} \tag{4.19}$$

$$\delta_{12}(u) = |N_{12}(u)| \tag{4.20}$$

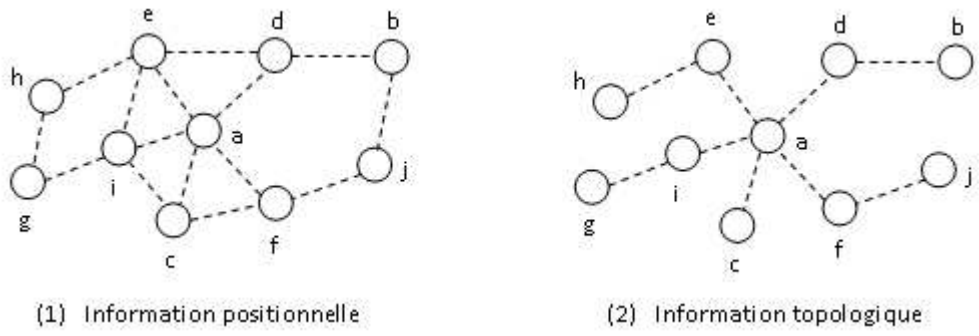


FIG. 4.5 – Comparaison entre l'information positionnelle et topologique du nœud *a*

Calcul de la mobilité

La mobilité d'un nœud *u* représente la vitesse de déplacement de *u* relativement à tous ses 1-voisins durant un intervalle de temps Δt. Elle est calculée comme suit :

$$\textit{Mobilité}_u(t) = \frac{1}{\Delta t}|D_u(t+\Delta t) - D_u(t)| \tag{4.21}$$

où

$$D_u(t) = \frac{1}{|N_1(u)|}\sum_{w \in N_1(u)} dist_{u,w}(t) \tag{4.22}$$

$$dist_{u,w}(t) = \sqrt{(x_u(t) - x_w(t))^2 + (y_u(t) - y_w(t))^2} \tag{4.23}$$

- $D_u(t)$: la distance euclidienne moyenne séparant le nœud *u* de ses 1-voisins à l'instant *t*.
- $dist_{u,w}(t)$:la distance euclidienne entre les nœuds *u* et *w* à l'instant *t*.
- $x_u(t),\ y_u(t)$: les coordonnées du nœud *u* à l'instant *t* puisque nous avons supposé que les capteurs sont déployés dans un espace bidimensionnel.

Calcul de l'énergie restante

Le niveau d'énergie d'un capteur est calculé en fonction du nombre de transmissions et réceptions au cours d'une période Δt. Nous avons négligé l'énergie consommée par les opérations de traitement et d'agrégation de données.

Pour l'évaluation de la consommation d'énergie $E_u(\Delta t)$ (4.24) par un capteur *u* pendant une période Δt, nous avons utilisé le modèle énergétique décrit dans [48]. Dans ce modèle,

la consommation d'énergie concerne principalement les transmissions et les réceptions. Ainsi, pour effectuer une transmission, la consommation d'énergie a besoin d'une énergie additionnelle pour amplifier le signal. Cette consommation additionnelle dépend de la distance séparant l'émetteur du récepteur. En effet, pour transmettre un message de taille *k* bits à une distance *d*, la radio dépense une quantité d'énergie E_{Tx} comme décrit par l'équation (4.25), où ϵ_{elec} est l'énergie consommée par les composants radio pour préparer le message à émettre, $\epsilon_{friss-amp}$ et $\epsilon_{two-ray-amp}$ sont les énergies nécessaires pour amplifier le message selon la distance *d*. La consommation d'énergie de réception E_{Rx} est représentée par l'équation (4.26).

Le tableau 4.2 résume les paramètres du modèle énergétique pour un paquet de données de taille 500 octets, plus un en-tête de 25 octets.

$$E_u(\Delta t) = N_{Tx} \times E_{Tx} + N_{Rx} \times E_{Rx} \tag{4.24}$$

où

$$E_{Tx} = \begin{cases} \epsilon_{elec} \times k + \epsilon_{friss-amp} \times k \times d^2 & \text{si } d < d_{Crossover} \\ \epsilon_{elec} \times k + \epsilon_{two-ray-amp} \times k \times d^4 & \text{si } d \geq d_{Crossover} \end{cases} \tag{4.25}$$

$$E_{Rx} = \epsilon_{elec} \times k \tag{4.26}$$

– N_{Tx} : représente le nombre de transmissions pendant la période Δt.

– N_{Rx} :représente le nombre de réceptions pendant la période Δt.

Si $E_u(t)$ l'énergie restante à l'instant *t* alors l'énergie restante après l'écoulement de la période Δt est :

$$E_u(t + \Delta t) = E_u(t) - E_u(\Delta t) \tag{4.27}$$

TAB. 4.2 – Paramètres du modèle énergitique utilisé

Paramètre	Valeur
ϵ_{elec}	$50nJ/bit$
$\epsilon_{friss-amp}$	$10\ pJ/bit/m^2$
$\epsilon_{two-ray-amp}$	$0.0013\ pJ/bit/m^4$
$d_{Crossover}$	$87\ m$
Taille d'un paquet de données	$500\ octets$
Taille de l'en-tête d'un paquet	$25\ octets$

4.4 Exemple d'application

Pour expliciter la technique de clustering proposée, nous considérons une topologie arbitraire composée de 20 nœuds générés aléatoirement (figure 4.6) par la fonction suivante :

```
void Area::generateNodes(int node_num)
{
  if(node_num <= 0)
      throw "Area::generateNodes: invalid node_num";
  for(int i = 0; i<node_num; i ++)
     nodes[i].setPos (rand()% (Max_x + 1),rand()% (Max_y + 1));
}
```

où les variables *Max_x* et *Max_y* représentent les dimensions de la zone de déploiement des capteurs. Nous supposons que tous les capteurs ont la même quantité d'énergie au début. D'où, nous pourrons négliger les paramètres énergie restante et mobilité au cours de la première exécution du processus de clustering. Ainsi, le calcul des poids des nœuds sera réduit à :

$$Weight(u) = 2 - \textit{densité}(u) \tag{4.28}$$

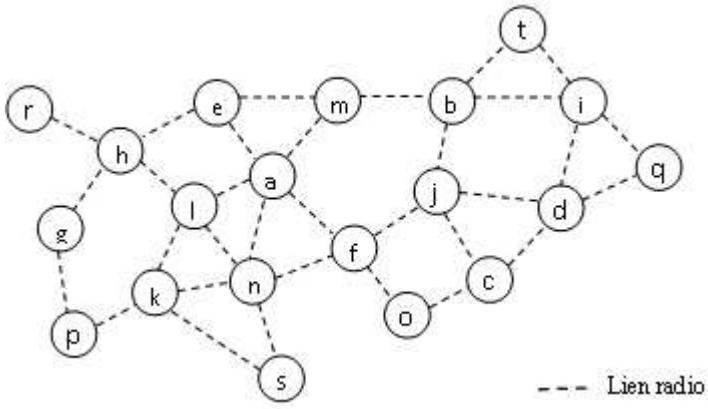


FIG. 4.6 – Exemple d'un réseau sans fil modélisé par un graphe non orienté

TAB. 4.3 – Informations topologiques des nœuds du réseau

Nœud	1-degré	2-degré	1-densité	2-densité
a	5	11	1.60	1.63
b	4	10	1.25	1.60
c	3	7	1.33	1.57
d	4	8	1.50	1.62
e	3	9	1.33	1.44
f	4	12	1.25	1.66
g	2	6	1.00	1.17
h	4	9	1.00	1.55
i	4	7	1.50	1.57
j	4	11	1.25	1.64
k	4	8	1.50	1.62
l	4	11	1.50	1.64
m	3	10	1.33	1.60
n	5	11	1.80	1.54
o	2	6	1.00	1.50
p	2	6	1.00	1.50
q	2	6	1.50	1.67
r	1	4	1.00	1.00
s	2	6	1.50	1.67
t	2	6	1.50	1.50

Au cours de la première phase, chaque nœud calcule son poids et le diffuse dans son 2-voisinage. Ensuite, le nœud ayant le plus grand poids (la grande 2-densité) dans son 2-voisinage sera élu comme cluster-head. Le tableau 4.3 illustre les données topologiques nécessaires à l'élection des cluster-heads .i.e. la formation des clusters.

Vu que CSOS est un algorithme distribué et son exécution se fait d'une manière asynchrone, nous pouvons obtenir des structures hiérarchiques différentes après chaque exécution. Ceci est dû au premier nœud qui initie le processus de clustering durant l'exécution de CSOS. Ainsi, le premier cluster-head élu sera dans le 2-voisinage du nœud qui a initié le processus de clustering et ses 2-voisins ayant une 2-densité inférieure que celle du cluster-head élu, devront le rejoindre dans la limite de sa capacité $Thresh_{Upper}$.

Pour illustrer l'exécution distribuée et asynchrone de CSOS, nous proposons de présenter deux exécutions de CSOS pour le même modèle topologique présenté par la figure 4.6. Nous choisissons pour cela les seuils arbitrairement .i.e. indépendamment de la topologie : $Thresh_{Upper} = 10$ et $Thresh_{Lower} = 4$.

Première exécution de CSOS

L'exécution de la première phase de CSOS donne lieu à la formation des clusters présentés par la figure 4.7.

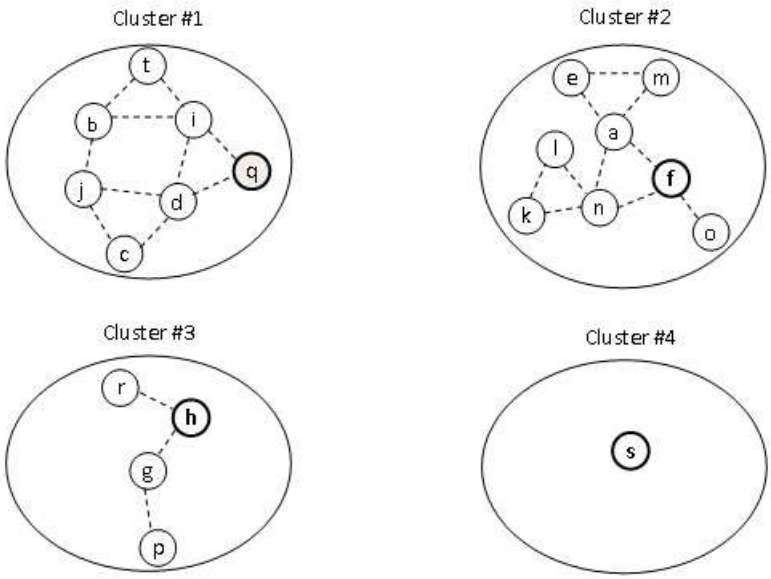


FIG. 4.7 – Les clusters formés après la phase set-up de la première exécution de CSOS

Nous remarquons que durant cette première phase, il y a formation d'un cluster dont la taille est inférieure au seuil $Thresh_{Lower}$ (figure 4.7).

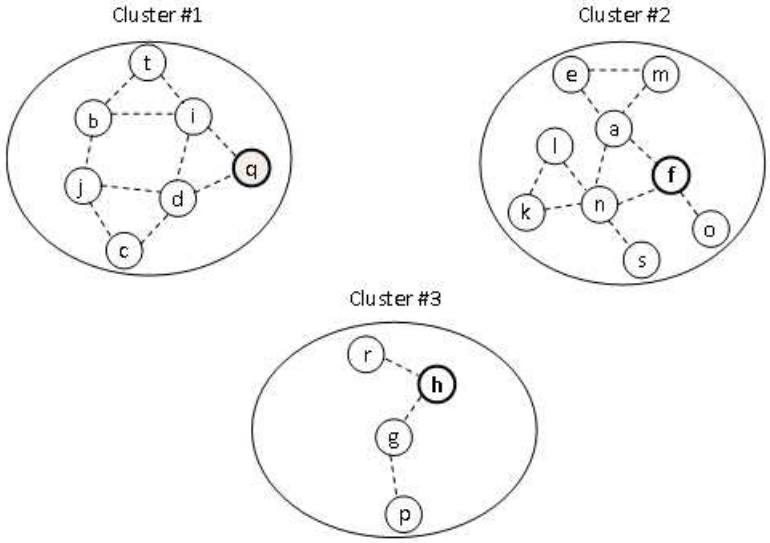


FIG. 4.8 – Les clusters formés après la phase de ré-affiliation de la première exécution de CSOS

L'application de la deuxième phase (la phase de ré-affiliation) achève la phase de formation de clusters. Elle permet de réorganiser les clusters en affiliant les nœuds appartenant aux clusters dont la taille est inférieure à $Thresh_{Lower}$ à ceux dont la taille n'a pas encore atteint $Thresh_{Upper}$. Dans cet exemple, le cluster numéro 4 est le seul cluster ayant une taille inférieure à $Thresh_{Lower}$. En effet, durant cette deuxième phase, ses membres vont être affiliés aux clusters dont les cluster-heads se trouvent à deux sauts au maximum de ses membres. S'il existe plus d'un cluster-head dans le 2-voisinage d'un membre, ce dernier est affilié au cluster-head dont la taille du cluster est inférieure à celles des autres. Ceci est dans le but de générer des clusters homogènes en taille. Le nœud *(s)* a un seul cluster-head *(f)* dans son 2-voisinage. Donc, il essaie de s'affilier à *(f)* selon la taille de son cluster .i.e. taille ne dépassant pas $Thresh_{Upper}$. Ainsi, après l'exécution de la deuxième phase, la topologie sera structurée selon la figure 4.8.

Deuxième exécution

Dans cette deuxième exécution (figure 4.9), il y a formation d'un cluster $(Cluster\#4)$ dont la taille est inférieure à la taille seuil $Thresh_{Lower}$ au cours de la première phase. L'exécution de la deuxième phase consiste à affilier les nœuds du $(Cluster\#4)$ vers les cluster-heads se trouvant dans le 2-voisinage de chacun de ses membres. Or, il existe deux cluster-heads (*f* et *l*) dans

le 2-voisinage du nœud *(s)*, donc ce dernier devrait affilier vers le cluster-head faisant parti du cluster de petite taille $(Cluster\#3)$. Mais, le nœud *n* qui permet de lier le cluster-head *f* au nœud *s* n'appartient pas au $(Cluster\#3)$, donc le nœud *(s)* affilie vers le cluster $(Cluster\#2)$ (figure 4.10).

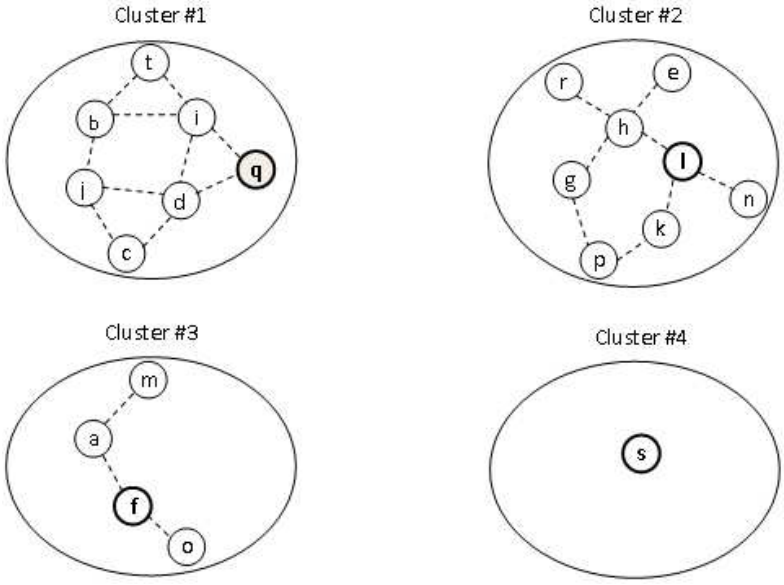


FIG. 4.9 – Les clusters formés après la phase set-up de la deuxième exécution de CSOS

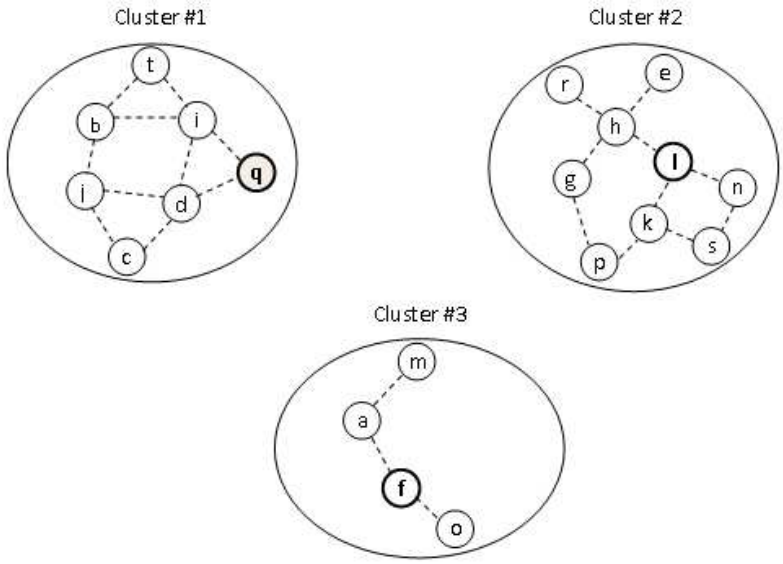


FIG. 4.10 – Les clusters formés après la phase de ré-affiliation de la deuxième exécution de CSOS

4.5 Contexte d'exécution de notre contribution

Le principal objectif de notre contribution est de proposer un algorithme générique pour l'acheminement de l'information dans les réseaux de capteurs. Pour que cet algorithme soit efficace, il fallait qu'il prenne en considération les spécificités des réseaux de capteurs et qu'il soit réalisable dans des conditions qui se rapprochent de la réalité. En effet, quelques facteurs ont été pris en compte à savoir :

- La distribution aléatoire des capteurs dans la zone d'intérêt,
- La génération aléatoire des vitesses de déplacement des capteurs,

En plus, certains paramètres sont considérés fondamentaux pour mettre en œuvre cet algorithme. Ces paramètres concernent :

- La position des capteurs : la topologie dynamique des réseaux de capteurs exige une mise à jour périodique des positions des capteurs. La position d'un capteur peut être calculée à l'aide d'un GPS ou estimer en fonction des signaux reçus de leurs 1-voisins. En revanche, l'utilisation d'un GPS pour les capteurs n'est pas une solution pratique puisque les capteurs sont contraints en énergie et la consommation d'énergie des GPS est importante.
- Connaissance du voisinage : la diffusion ou le routage d'une information exige la connaissance du voisinage par tout capteur du réseau. Donc, un capteur doit connaître si un autre capteur se trouve toujours parmi ses 1-voisins ou non pour que la diffusion ou le routage soit effectué normalement. Les échanges périodiques des messages *"Hello"* ou des beacons sont utilisées pour cette finalité. Dans notre contexte, il est nécessaire que chaque capteur ait une connaissance positionnelle de tous ses 2-voisins.
- Connectivité : est un facteur déterminant dans les réseaux ad hoc et de capteurs. Elle permet d'assurer que tout nœud destinataire est joignable par tout nœud source du réseau. En particulier, dans les réseaux de capteurs, il devrait exister au moins un chemin entre tout capteur et la station de base pour garantir l'acheminement de l'information en tout moment et que tous les nœuds soient joignables à partir de la station de base.
- Couverture de zone d'intérêt : est un élément important pour les réseaux de capteurs. Tout algorithme conçu pour les réseaux de capteurs devrait assurer que l'occurrence d'un tel événement dans la zone de déploiement des capteurs pourrait être détectée par au moins un capteur (sensing fidelity).

4.6 Analyse de CSOS

Nous présentons dans cette section, les performances de CSOS et l'environnement d'exécution des simulations.

4.6.1 Environnement de simulation

Nous avons réalisé plusieurs simulations pour évaluer les performances de l'algorithme CSOS en termes de clusters formés et le nombre de changements des cluster-heads [67]. Puis, nous avons comparé ces résultats à ceux de l'algorithme WCA, à ceux proposés respectivement par Lin et al., et Chu et al. en termes de nombre de clusters formés, puis à l'algorithme LCC en termes de nombre de changements de cluster-heads. Pour cela, nous avons exploité le simulateur NS-2 [125] pour implémenter l'algorithme proposé. En plus, puisque la mobilité est prise en considération dans le modèle d'exécution, nous avons utilisé le modèle "Random WayPoint" (RWP) [83] pour générer des modèles de mobilité des nœuds. Dans ce modèle, chaque nœud choisit sa direction et sa vitesse de déplacement après chaque intervalle de temps et il peut y avoir de même des périodes de pause.

Pour comparer les performances de CSOS aux autres algorithmes, nous avons réalisé les simulations dans des contextes d'exécution différents. Ainsi, afin d'analyser la robustesse de CSOS face à la mobilité des nœuds, nous avons choisi d'évaluer les métriques suivantes et les comparer à d'autres algorithmes :

- Le nombre moyen de clusters formés en fonction de la portée de transmission,
- Le nombre moyen de clusters formés en fonction de la vitesse de déplacement des nœuds,
- Le taux de connectivité en fonction du rayon de transmission et de la vitesse de déplacement des nœuds,
- Le nombre moyen de changements de cluster-heads.

4.6.2 Evaluation de CSOS en termes de clusters formés

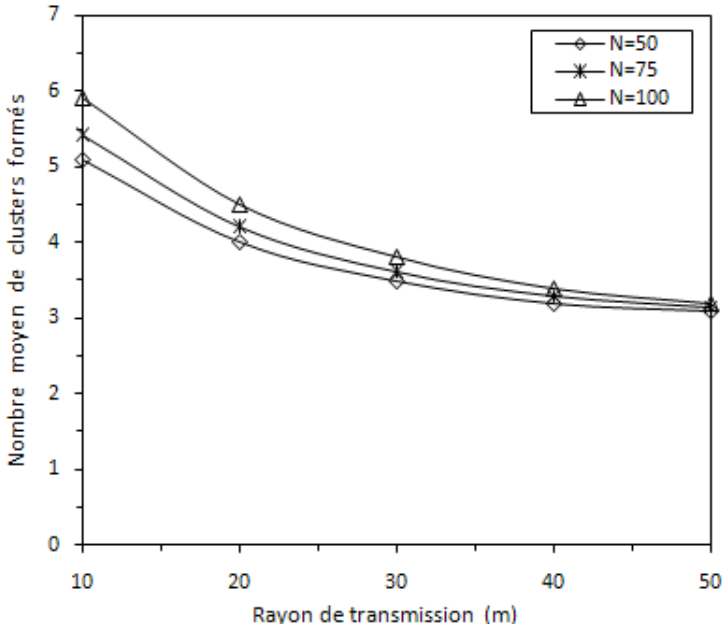
Nous évaluons CSOS en termes de clusters formés en fonction de la portée de transmission et de la vitesse de déplacement des nœuds.

Nombre de clusters formés en fonction de la portée de transmission

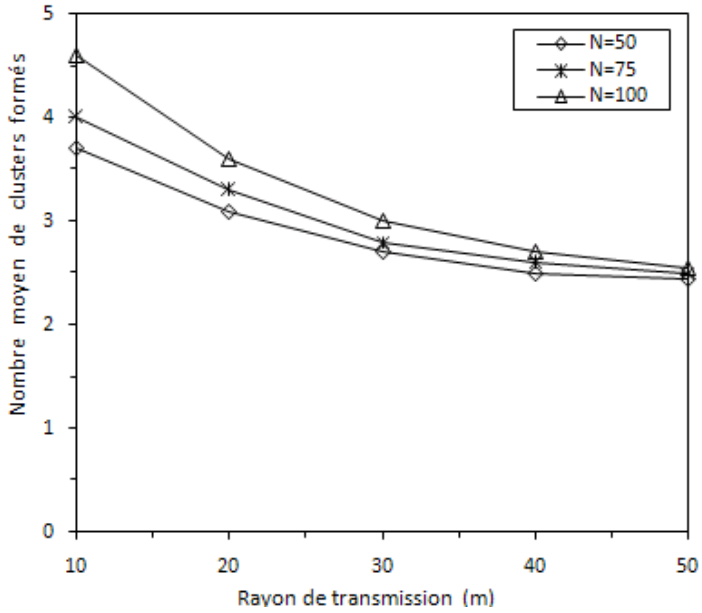
Au début, nous avons mené des simulations pour calculer le nombre de clusters formés par rapport au rayon de transmission [69]. Nous avons pris des réseaux dont la taille était : 50, 75, 100, et des nœuds mobiles. La vitesse de déplacement des nœuds est générée d'une manière aléatoire suivant une loi uniforme sur l'intervalle $[0, 10]m.s^{-1}$. Le tableau 4.4 résume les paramètres de simulation.

TAB. 4.4 – Paramètres de simulation pour calculer le nombre de clusters formés par rapport au rayon de transmission

Paramètres	Valeurs
Surface de déploiement	$100m \times 100m$
Vitesse de déplacement des nœuds	$[0 - 10]\ m.s^{-1}$
Rayon de transmission (R_t)	10,20,30,40,50
Nombre de nœuds	50,75,100
$Thresh_{Upper}$	30,50



(a) $CSOS\_30$



(b) $CSOS\_50$

FIG. 4.11 – Le nombre de clusters formés en fonction du rayon de transmission

Les figures 4.11(a) et 4.11(b) présentent respectivement le nombre de clusters formés en fonction du rayon de transmission pour $Thresh_{Upper} = 30$ et $Thresh_{Upper} = 50$. Nous constatons que le nombre de clusters générés tend à se stabiliser indépendamment de la taille des réseaux quand le rayon de transmission augmente et devient approximativement stable lorsque $R_t = 50m$, 3 clusters lorsque $Thresh_{Upper} = 30$ et 2.5 quand $Thresh_{Upper} = 50$.

Nombre de clusters formés en fonction de la vitesse de déplacement des nœuds

Chen et Stojmenović [27] considèrent que l'efficacité d'un algorithme de clustering est fonction du nombre de clusters produits i.e. plus le nombre de clusters formés est réduit, plus l'algorithme de clustering est efficace. Cependant, cette métrique d'évaluation n'est pas valide pour les réseaux de capteurs parce que si le nombre de clusters est très réduit, les cluster-heads se trouveront très éloignés de la station de base et par suite ils dépenseront plus d'énergie pour une transmission de longue portée. Dans cette partie, nous évaluons le nombre moyen de clusters produits en fonction de la vitesse de déplacement des nœuds dans deux modèles différents : réseaux dans lesquels les nœuds se déplacent avec une faible vitesse et ceux dans lesquels les nœuds se déplacent avec une grande vitesse. De ce fait, nous avons évalué CSOS et l'avons comparé à WCA dans le premier modèle puisque WCA fournit de bons résultats avec ce type de modèle. Puis, nous avons évalué CSOS dans un environnement fortement mobile et comparé aux

algorithmes proposés respectivement par Lin et al., et Chu et al.

Pour comparer CSOS à WCA, nous avons considéré une topologie réseau, dans laquelle les nœuds sont déployés aléatoirement sur une surface $100 \times 100m$ suivant une loi de distribution uniforme, et nous avons supposé que le comportement de la couche MAC n'était pas pris en considération i.e. les collisions et les interférences ne se produisent pas durant la simulation. Le tableau 4.5 résume les paramètres de simulation de cette évaluation.

TAB. 4.5 – Paramètres de simulation pour un environnement moins mobile

Paramètres	Valeurs
Surface de déploiement	$100 \times 100m$
Vitesse de déplacement des nœuds	$\{2, 4, 6, 8, 10\}\ m.s^{-1}$
Rayon de transmission (R_t)	25 m
Nombre de nœuds	40
$Thresh_{Upper}$	10,15

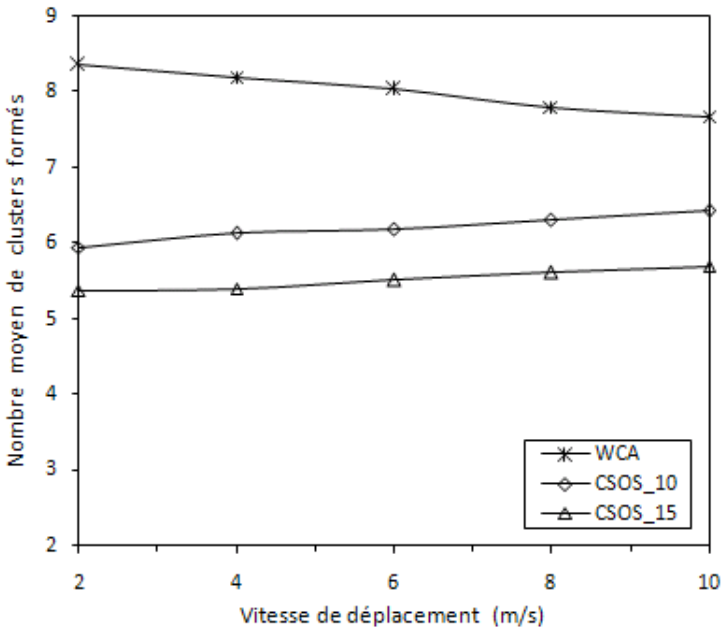


FIG. 4.12 – Comparaison du nombre moyen de clusters formés en fonction de la vitesse de déplacement des nœuds dans un environnement moins mobile

Nous avons exécuté CSOS respectivement avec $Thresh_{Upper} = 10$ et $Thresh_{Upper} = 15$ i.e. *CSOS_10* et *CSOS_15*. Nous avons constaté que le nombre de clusters formés par ces deux variantes de l'algorithme CSOS est inférieur à celui généré par l'algorithme WCA et il est approximativement stable dans les deux cas (figure 4.12). Cette stabilité est illustrée par la petite variation du nombre de clusters formés par *CSOS_10* et *CSOS_15* : par *CSOS_10*, ce nombre varie dans un petit intervalle $[5.95, 6.43]$ et par *CSOS_15* dans l'intervalle $[5.38, 5.7]$. Ceci est dû à l'implication de la métrique k-densité dans l'élection des cluster-heads et à l'exécution de la deuxième phase de CSOS.

Puis, nous avons évalué CSOS dans un environnement très mobile pour connaître sa robustesse et sa stabilité face à la forte mobilité des nœuds et nous avons comparé ses performances à celles obtenues respectivement par les algorithmes proposés par Lin et al., et Chu et al.. Le tableau 4.6 présente les paramètres de simulation pour l'évaluation de CSOS dans un environnement fortement mobile.

TAB. 4.6 – Paramètres de simulation pour un environnement fortement mobile

Paramètres	Valeurs
Surface de déploiement	$100 \times 100m$
Vitesse de déplacement des nœuds	$\{10, 20, 30, 40, 50, 60\}\ m.s^{-1}$
Rayon de transmission (R_t)	25 m
Nombre de nœuds	200
$Thresh_{Upper}$	15
$Thresh_{Lower}$	5

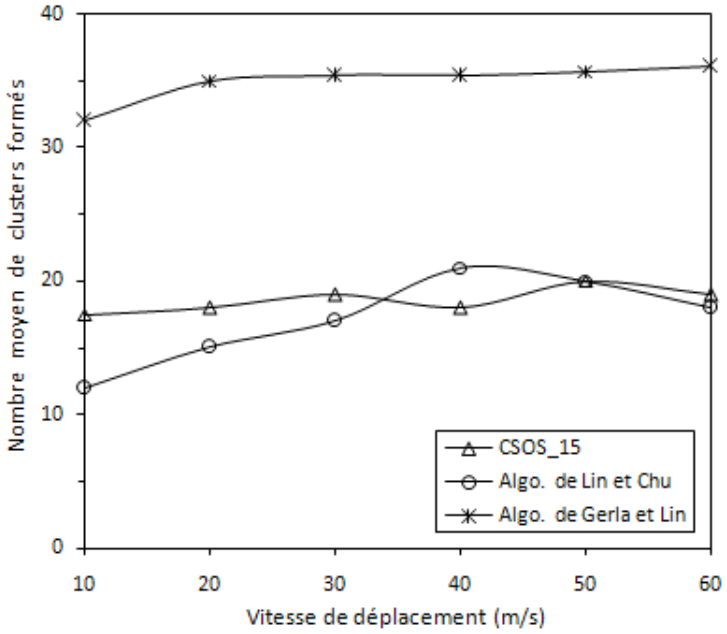


FIG. 4.13 – Comparaison du nombre moyen de clusters formés en fonction de la vitesse de déplacement des nœuds dans un environnement fortement mobile

La figure 4.13 présente le nombre de clusters formés en fonction de la vitesse de déplacement des nœuds dans un environnement fortement mobile. Nous constatons que le nombre moyen de clusters formés est approximativement stable malgré la grande vitesse de déplacement des nœuds. Il varie dans l'intervalle $[17.5, 19]$, ce qui implique que les métriques utilisées pour élire les cluster-heads et l'exécution de la deuxième phase de CSOS permettent de produire un nombre approximativement fixe de clusters.

Nombre moyen de changements de cluster-heads

Il est généralement difficile de gérer le clustering dans les réseaux mobiles parce que le changement de la topologie est fréquent à cause des déplacements de nœuds. En particulier dans les réseaux de capteurs, il pourrait y avoir trois types de mobilité : la mobilité des capteurs, de la station de base et de l'événement. Ceci pourrait avoir des conséquences sur l'overhead des activités locales et dégrade le taux de connectivité des nœuds. La plupart des techniques conçues pour les réseaux mobiles ont été évaluées pour des réseaux de petite ou moyenne taille et peu d'entre elles ont été conçues sur la base d'un modèle concret de mobilité.

Un algorithme de clustering est efficace si les cluster-heads ne changent pas fréquemment leurs statuts parce que l'occurrence de ces changements cause une restructuration locale ou générale du réseau. De ce fait, pour évaluer la robustesse de CSOS face à la mobilité des nœuds, nous proposons d'estimer le nombre moyen de changements des cluster-heads en fonction de la vitesse de déplacement des nœuds. Pour cela, nous avons utilisé le modèle de simulation présenté par le tableau 4.7.

TAB. 4.7 – Paramètres de simulation pour calculer le nombre moyen de changements de cluster-heads

Paramètres	Valeurs
Surface de déploiement	$500 \times 500m$
Vitesse de déplacement des nœuds	$\{2, 4, 6, 8, 10\}\ m.s^{-1}$
Rayon de transmission (R_t)	25 m
Nombre de nœuds	100
$Thresh_{Upper}$	10,15
$Thresh_{Lower}$	5

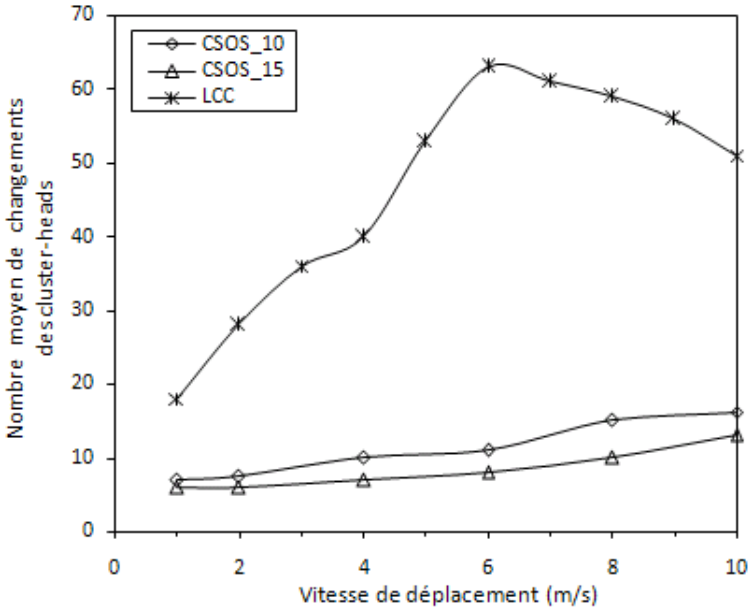


FIG. 4.14 – Comparaison du nombre moyen de changements de cluster-heads en fonction de la vitesse de déplacement des nœuds

Nous avons exécuté CSOS respectivement avec $Thresh_{Upper} = 10$ et $Thresh_{Upper} = 15$ (*CSOS_10* et *CSOS_15*). Nous avons constaté que le nombre moyen de changements de cluster-heads généré par ces deux variantes de l'algorithme CSOS est inférieur à celui produit par l'algorithme LCC et il est approximativement stable dans les deux cas. Cette stabilité est approuvée par la faible variation de ce nombre en fonction de la vitesse de déplacement des nœuds : pour *CSOS_10,* ce nombre varie dans un petit intervalle $[5.95, 6.43]$ et pour *CSOS_15* dans l'intervalle $[5.38, 5.7]$. Ceci est dû à l'implication des métriques favorisant la stabilité (la k-densité et la mobilité) dans l'élection des cluster-heads.

La figure 4.14 illustre l'évolution du nombre moyen de changements de cluster-heads en fonction de la vitesse de déplacement des nœuds. Nous observons que les changements sont fréquents par LCC quand la vitesse de déplacement varie entre 4 et 8 et s'affaiblissent quand la vitesse augmente. Ceci est dû au retour des cluster-heads à leurs clusters d'origine avant l'expiration de la période T_{Seuil} exigée pour déclencher l'opération de maintenance quand deux cluster-heads se trouvent voisins.

Connectivité par rapport au rayon de transmission

La connectivité est un critère de performance déterminant pour les réseaux de capteurs. Elle illustre la fidélité du routage entre les capteurs et la station de base et vice versa. Un algorithme de diffusion est dit efficace si sa connectivité est au voisinage de 1. Pour évaluer la connectivité du réseau par CSOS, nous réalisons des simulations en utilisant les paramètres décrits dans le tableau 4.8.

TAB. 4.8 – Paramètres de simulation pour l'évaluation de la connectivité

Paramètres	Valeurs
Surface de déploiement	$100 \times 100m$
Vitesse de déplacement des nœuds	$[0 - 10]\ m.s^{-1}$
Rayon de transmission (R_t)	$\{10, 20, 30, 40, 50, 60, 70\}\ m$
Nombre de nœuds	40
$Thresh\_Upper$	15
$Thresh\_Lower$	5

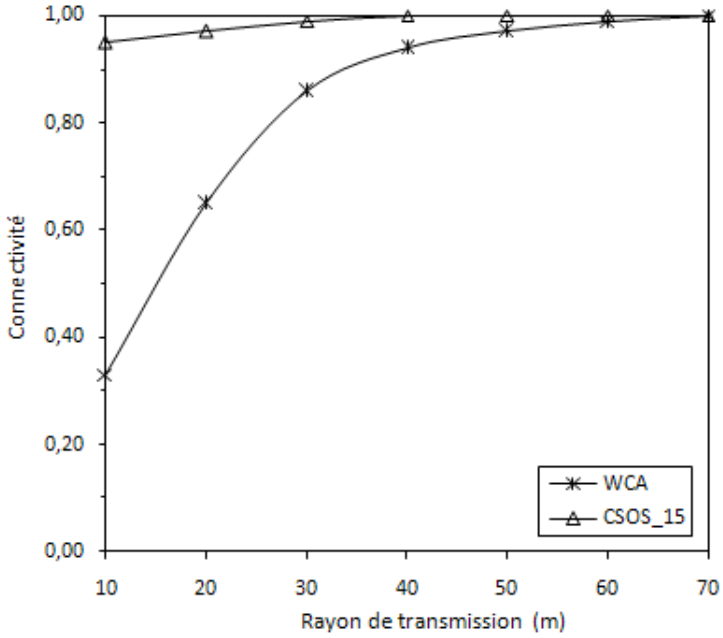


FIG. 4.15 – Connectivité en fonction du rayon de transmission

La figure 4.15 montre l'évolution de la connectivité par rapport au rayon de transmission par WCA et CSOS. Nous constatons que la connectivité générée par WCA croît graduellement en fonction du rayon de transmission et sa valeur est au dessous de 0.9 quand le rayon de transmission est inférieur à 35 m $(R_t < 35)$. Elle atteint la valeur 1 quand le rayon de transmission est égal à 70 m $(R_t = 70)$. Par contre, dans CSOS, elle atteint 0.9 à partir de $R_t = 20$. Ainsi, si nous choisissons un rayon de transmission égal à 25m, alors il existe au moins un lien entre tout nœud du réseau et la station de base.

4.7 Conclusion

Dans ce chapitre, nous avons proposé un nouvel algorithme appelé "CSOS" pour l'auto-organisation des réseaux de capteurs mobiles. CSOS prend en considération les spécificités et les contraintes imposées par les réseaux de capteurs et il est adapté à un type particulier de réseaux de capteurs : les réseaux de capteurs mobiles. Nous avons visé avec CSOS la création d'une topologie virtuelle stable pour minimiser la réélection fréquente et éviter la restructuration globale de tout le réseau. Pour cela, nous avons impliqué des métriques de stabilité pour élire les cluster-heads : k-densité, mobilité et énergie restante. En effet, les résultats obtenus ont prouvé

que CSOS était robuste face à la mobilité des nœuds, et les résultats sont meilleurs que ceux obtenus par d'autres algorithmes de clustering. Nous avons évalué CSOS dans des environnements de simulation différents et nous l'avons comparé à d'autres algorithmes en termes du nombre moyen de clusters formés, de stabilité et de connectivité. Nous avons constaté que les performances de CSOS dépassent celles de WCA en termes du nombre moyen de clusters formés et de connectivité dans les réseaux de petite et de moyenne taille, dans le contexte de faible mobilité. De même, CSOS a présenté de meilleures performances en termes du nombre de clusters formés par comparaison aux algorithmes proposés respectivement par Lin et al., et Chu et al. dans des environnements de forte mobilité. Enfin, nous avons évalué et comparé CSOS à LCC en termes du nombre moyen de changements de cluster-heads et nous avons remarqué que CSOS était plus stable que LCC.

Dans le chapitre suivant, nous évaluons les performances CSOS en termes de durée de vie, de consommation d'énergie et du nombre de paquets envoyés à la station de base durant la vie du réseau pour illustrer l'apport de la stabilité de l'algorithme que nous avons proposé sur ces performances.

Chapitre 5

Evaluation des performances de CSOS

5.1 Introduction

Le prolongement de la durée de vie des réseaux de capteurs mobiles est devenu un vrai défi puisque les capteurs sont dotés de batteries à énergie limitée. De plus, leur rechargement ou leur remplacement reste impossible lorsque les capteurs sont déployés dans des environnements hostiles et inaccessibles. De ce fait, la durée de vie du réseau dépend donc de la durée de vie de leur batterie. Contrairement aux laptops, nous nous attendons à ce que les capteurs fonctionnent pour une longue durée (plusieurs années), ce qui nécessite la mise en place des techniques innovatrices pour minimiser la consommation d'énergie.

Cette consommation d'énergie est principalement fonction des communications dans les réseaux de capteurs puisque les communications consomment plus d'énergie relativement aux autres opérations. Ainsi, la réduction de l'énergie de communication dans les réseaux de capteurs exige que chaque aspect de la communication tel que les couches MAC et réseau, soit dédié pour une application particulière. Un capteur dans un réseau cherche dans son 1-voisinage les meilleurs capteurs candidats pour retransmettre son message de diffusion. Cependant, avec la densité élevée des capteurs, le paradigme de diffusion devrait être reconsidéré. Ainsi, concevoir une technique optimale de diffusion dans un réseau de capteurs reste une tâche très difficile puisque chaque capteur choisit efficacement ses propres capteurs sources pour recevoir des données. Cependant, optimiser à un niveau local ne conduit pas forcément à une optimisation globale. En plus, la caractérisation d'énergie devrait tenir compte de l'overhead de la couche MAC et de la complexité de la technique de diffusion adoptée. De ce fait, un algorithme d'auto-organisation efficace pourrait augmenter considérablement la durée de vie d'un réseau. Dans ce

cadre, nous avons proposé un algorithme d'auto-organisation basé sur l'approche de clustering, dans lequel la contrainte énergie est utilisée pour élire les cluster-heads et l'élection des cluster-heads se fait périodiquement dans le but de distribuer équitablement la consommation d'énergie dans les réseaux de capteurs et donc obtenir une longue durée de vie du réseau. Dans CSOS, chaque capteur calcule son poids en fonction de sa k-densité, son énergie résiduelle et sa mobilité. L'élection des cluster-heads se fait sur la base des valeurs des poids des capteurs dans un 2-voisinage.

Une technique de diffusion efficace en consommation d'énergie permet le passage à l'échelle, supporte le changement de la topologie du réseau à cause de la mobilité des capteurs et reste insensible aux défaillances des capteurs qui cessent de fonctionner à cause d'une panne ou de l'épuisement de leurs batteries. En outre, elle doit maintenir la connectivité du réseau autant que possible pour éviter une telle déconnexion des capteurs, distribuer équitablement la consommation d'énergie entre les capteurs et choisir les routes les plus économiques en terme de conservation d'énergie.

Dans le but d'illustrer les atouts substantiels de CSOS, nous avons réalisé des simulations pour montrer l'impact de la mobilité des capteurs sur les performances de LEACH et LEACH-C dans les environnements mobiles. Malheureusement, les résultats obtenus ont prouvé que la mobilité a un impact significatif sur les performances de ces deux protocoles en termes de durée de vie du réseau et du nombre de paquets de données envoyés à la station de base. Par contre, CSOS a fourni de meilleures performances que ces deux protocoles.

Dans ce travail, nous avons visé la réduction de la consommation d'énergie et le prolongement de la durée de vie des réseaux de capteurs. Pour atteindre ces objectifs, nous avons proposé une technique de clustering, qui implique la k-densité et la mobilité dans le calcul du poids des capteurs afin de garantir la stabilité des clusters générés et l'énergie restante pour assurer une longue durée de vie des cluster-heads. Au début, nous avons réalisé des simulations pour évaluer les performances des deux protocoles : LEACH et LEACH-C en utilisant le même scénario présenté dans [47, 48], mais avec des capteurs mobiles pour voir la robustesse de ces protocoles face à la mobilité des capteurs. Puis, nous avons effectué des simulations pour illustrer les performances de notre contribution en termes de durée de vie des réseaux et du nombre de paquets envoyés à la station de base.

La durée de vie d'un réseau de capteurs est définie en utilisant trois métriques FND (First Node Dies), HNA (Half of the Nodes Alive), et LND (Last Node Dies). Dans notre travail, nous avons évalué la durée de vie d'un réseau de capteurs dans deux contextes. Le premier contexte suit la métrique LND i.e. le temps écoulé jusqu'à ce que le dernier capteur cesse de fonctionner, alors que dans le deuxième contexte nous avons supposé que le réseau est fonctionnel tant que le taux de capteurs morts ne dépasse pas un certain seuil α-coverage et que la couverture totale de la zone d'intérêt est pour un taux au dessus de α-coverage.

5.2 Communication entre capteurs

Les capteurs peuvent ne pas utiliser des adresses explicites au niveau de la couche MAC et réseau, mais plutôt une métrique de leur distance approximative à la station de base la plus proche. Cette métrique peut être propagée dans le réseau via un message d'inondation, à partir des stations de base et tout capteur recevant le message ajoute un facteur constant à la plus petite valeur reçue. Initialement, cette métrique vaut zéro et lorsqu'un capteur reçoit un paquet qui est destiné à la station de base, il vérifie si sa métrique est inférieure à celle se trouvant dans le paquet, si oui il injecte sa nouvelle métrique dans le paquet et la diffuse, sinon il l'ignore puisqu'il se trouve plus éloigné de la station de base que le capteur émetteur. Pour minimiser le nombre de sauts, le capteur le plus proche de la station de base et le plus éloigné du capteur émetteur est un meilleur candidat pour relayer le message.

Un tel comportement peut être mis en place, avec un temporisateur de retard qui est proportionnel à la différence entre la métrique distance véhiculée dans le paquet et celle du capteur relais, ou simplement RSSI (Receiver Signal Strength Indication). Ainsi, le capteur ayant la plus petite valeur Δt doit relayer le message et les autres capteurs l'ignorer.

μAMPS (micro-Adaptive Multidomain Power-Aware Sensors) [123] développent des technologies permettant l'économie d'énergie dans les réseaux de capteurs. Un capteur μAMPS-1 est composé des système de détection, de traitement et radio. Le système de détection se compose d'un capteur et d'un convertisseur A/D. La radio transmet et reçoit à 1 Mbps dans la bande 2.4 GHz. Quand les paquets sont extrêmement courts, l'énergie radioélectrique pourrait dépasser l'énergie réelle d'une transmission. Ainsi, les paquets dont la taille est courte, sont accumulés au niveau des capteurs et transmis à la fois dans le but de réduire la quantité d'énergie radioélectrique dépensée. Par exemple, l'envoi de dix paquets dans des intervalles de temps de 10 secondes permet d'économiser plus que la moitié de l'énergie dépensée des transmissions immédiates. Ce-

pendant, ce procédé a un impact sur la latence et par conséquent il ne peut pas être appliqué dans un scénario event-driven. Dans CSOS, l'agrégation de données est réalisée au niveau des cluster-heads pour éviter les transmissions redondantes de la même donnée environnementale à la station de base.

5.3 Consommation d'énergie des capteurs

La conservation d'énergie est une préoccupation principale dans les réseaux de capteurs : les batteries ont une petite capacité et leur remplacement et rechargement sont généralement difficiles voire impossible. Par conséquent, la consommation d'énergie d'un capteur doit être bien contrôlée parce que lorsque les capteurs épuisent leurs batteries, la connectivité diminue et le réseau peut être devenu dysfonctionnel. Ainsi, pour maximiser la durée de vie des capteurs après leurs déploiements, la technique de leur self-organisation devrait être efficace en économie d'énergie. En plus, une fois le système conçu, des économies d'énergie additionnelles peuvent être atteintes en utilisant une gestion dynamique de puissance permettant aux capteurs d'éteindre leur radio si aucun événement n'est survenu. Si les capteurs consomment équitablement leurs batteries, alors ils continuent à fournir la connectivité pour plus longtemps. Les principales sources de consommation d'énergie sont :

- La transmission : les composants assurant la transmission tels que le transceiver et l'antenne deviennent actifs.
- La réception : les composants assurant la réception deviennent actifs.
- L'écoute libre (idle listening), quand un capteur est prêt à recevoir mais il ne reçoit pas actuellement des données. Dans cet état, plusieurs composants des circuits sont actifs, et d'autres sont éteints. Par exemple, dans les circuits de synchronisation, les éléments assurant l'acquisition sont en activité, alors que les éléments assurant la transmission sont inactifs.
- Les collisions entre trames causent des consommations inutiles d'énergie par l'émetteur et le récepteur. Aussi, dans certains cas, les collisions entraînent des retransmissions.
- La sur-écoute (overhearing) : un nœud dépense de l'énergie inutilement lorsqu'il reçoit un paquet qui ne lui est pas destiné.
- L'état sommeil : la plupart des composants sont inactifs. Les états sommeil diffèrent des composants inactifs et de l'énergie doit être dépensée pour leur activation [106]. Par exemple, quand le transceiver est éteint complètement, les coûts de son démarrage incluent

son initialisation complète tandis qu'en mode sommeil "plus légers", sa configuration et son état opérationnel sont activés.

Quelque soit le scénario utilisé : collecte et envoi périodique de données ou event-driven, un capteur passera plus de temps à ne rien faire. Par conséquent, il est judicieux d'éteindre sa radio pour conserver son énergie. Naturellement, il devrait se réveiller lorsque des stimuli externes apparaissent ou après chaque période de repos. Pratiquement, éteindre complètement un capteur n'est pas possible, mais plutôt adapter son état opérationnel aux tâches courantes. L'utilisation des états permettant l'économie d'énergie et la conception d'une technique de auto-organisation des réseaux de capteurs favorisent largement la réduction de la consommation d'énergie.

5.4 Radio transceivers

Un transceiver a essentiellement deux tâches : la transmission et la réception de données entre une paire de capteurs. Il peut opérer en différents modes, les plus simples sont éteint et actif. Pour l'adapter à une optimisation de consommation d'énergie, le transceiver devrait être placé en mode éteint le plus longtemps possible et être activé au besoin. Cependant, ce mécanisme encourt des overheads additionnels de complexité, de temps et d'énergie qui doivent être pris en considération. Pour comprendre le comportement de la consommation d'énergie des transceivers et son impact sur la technique de diffusion, des modèles de consommation d'énergie par bit pour transmettre et recevoir ont été proposés.

5.5 Modélisation de la consommation d'énergie

La consommation d'énergie dans les réseaux sans fil et en particulier dans les réseaux de capteurs concerne essentiellement les communications : transmission et réception.

5.5.1 Modélisation de la consommation d'énergie pour la transmission

Les transmissions dans les réseaux sans fil utilisent généralement le mode multi-saut pour améliorer les déperditions causées par les longs liens radio, car plus le lien radio est long plus le message pourrait être corrompu à cause des erreurs qui peuvent être injectées dans le message (l'atténuation). Aussi, dans certains cas, une transmission via un long lien radio consomme plus d'énergie qu'une transmission en mode multi-saut via des liens de courte portée.

L'énergie consommée par un émetteur est divisée en deux parties [90] : une partie consiste à générer des signaux *RF*, qui dépendent de la distance séparant l'émetteur du récepteur, de la modulation choisie, et de la puissance de transmission ϵ_{Tx}. La seconde partie est due aux composants électroniques nécessaires pour la synthèse de fréquence, la conversion de fréquence, et ainsi de suite. Le coût énergétique de la seconde partie est constant. La puissance de transmission ϵ_{Tx} est générée par l'amplificateur du transmetteur et son choix pour envoyer un paquet est une tâche cruciale car ϵ_{Tx} dépend des aspects du système tels que l'énergie par bit lors de l'atténuation du signal (E_b/N_0), l'efficacité de la bande passante η_{BW}, la distance *d* séparant l'émetteur de la cible et le coefficient de la déperdition du chemin γ. Dans notre contexte, nous supposons que sa valeur est connue et constante. En plus de la puissance d'amplification, d'autres circuits doivent être activés lors d'une transmission. Cette puissance est représentée par ϵ_{TxElec}. Ainsi, l'énergie nécessaire pour transmettre un paquet de taille *k* bits dépend de la distance séparant l'émetteur du récepteur, de la puissance d'amplification ϵ_{Tx} ($\epsilon_{friss-amp}$ pour les courtes portées et $\epsilon_{two-ray-amp}$ pour les longues portées) et de la puissance ϵ_{TxElec}. L'équation (5.1) récapitule sa valeur :

$$E_{Tx} = \begin{cases} \epsilon_{elec} \times k + \epsilon_{friss-amp} \times k \times d^2 & \text{si } d < d_{Crossover} \\ \\ \epsilon_{elec} \times k + \epsilon_{two-ray-amp} \times k \times d^4 & \text{si } d \geq d_{Crossover} \end{cases} \tag{5.1}$$

5.5.2 Modélisation de la consommation d'énergie pour la réception

La réduction de l'énergie de réception radio est une préoccupation primaire dans les réseaux de capteurs. La radio d'un récepteur se trouvant dans l'état idle consomme une quantité d'énergie substantielle, dans certaines plateformes plus qu'une transmission.

Etant donné que plusieurs capteurs seront probablement dans la portée d'un tel capteur, il est judicieux d'éteindre la radio des capteurs se trouvant dans l'état idle pour éviter les réceptions du message par plusieurs capteurs. Malheureusement, la plupart des protocoles et les couches MAC utilisent des adresses uniques pour acheminer les paquets aux destinations finales, en espérant toujours que ces destinations se trouvent à l'état écoute. De plus, avec l'extinction de la radio, cette hypothèse ne tient plus et les tables de routage deviendront instables si elles ne contiennent pas l'information sur les routes issues de la source ou le prochain saut.

Un récepteur peut être actif ou éteint. Lorsqu'un capteur se trouve dans l'état actif, il est peut-être, soit en cours de recevoir des paquets, soit il supervise le canal et il est prêt à recevoir

des paquets. Dans la plupart des cas, l'énergie consommée au cours de l'état actif est presque la même que l'énergie dépensée lors d'une réception. Ainsi, l'énergie E_{Rx} exigée pour recevoir un paquet de taille *k* bits correspond à l'énergie nécessaire pour préparer le transceiver à une réception. Elle est similaire à celle de l'activation des composants électroniques au cours d'une transmission E_{TxElec} Ainsi :

$$\begin{aligned} E_{Rx} &= E_{TxElec} \\ &= \epsilon_{Elec} * k \quad (bits) \end{aligned} \tag{5.2}$$

5.6 Evaluation de la consommation d'énergie dans un réseau de capteurs

La consommation d'énergie dans les réseaux sans fil concerne essentiellement les communications (transmissions et réceptions). Donc, pour l'évaluer, il suffit de connaître le nombre de transmissions et de réceptions dans le réseau de capteurs. Soient $N_{Tx}(t)$ le nombre de transmissions et $N_{Rx}(t)$ le nombre de réceptions à l'instant *t*, et soient $E_{Tx}(t)$ l'énergie consommée par les transmissions et $E_{Rx}(t)$ l'énergie consommée par les réceptions, l'énergie totale $E(t)$ à l'instant *t* est donc :

$$E(t) = E_{Tx}(t) + E_{Rx}(t) \tag{5.3}$$

Supposons que E_{Tr} et E_{Re} représentent respectivement l'énergie nécessaire à la transmission et à la réception d'un paquet de taille *k* bits, et que tous les paquets ont la même taille. D'où :

$$E(t) = N_{Tx}(t) * E_{Tr} + N_{Rx}(t) * E_{Re} \quad (bits) equation 54 \tag{5.4}$$

5.7 Résultats numériques

Pour illustrer l'apport de la stabilité que rapporte CSOS sur le comportement d'un réseau de capteurs, nous avons évalué les performances de CSOS en termes de durée de vie et quantité de données envoyées à la station de base. Puis, nous avons comparé ces résultats à ceux obtenus avec LEACH [47] et LEACH-C [48] dans un environnement mobile pour illustrer les gains substantiels des performances de CSOS.

Au début, nous avons utilisé le même scénario présenté dans [47, 48] pour évaluer LEACH et sa variante LEACH-C dans un environnement mobile et nous avons comparé les résultats

obtenus à ceux obtenus dans [47, 48] (avec des capteurs stationnaires), ceci dans le but d'illustrer l'impact de la mobilité des capteurs sur la robustesse de LEACH et LEACH-C. Pour cela, nous avons réalisé des simulations avec NS-2 [125] en utilisant les extensions MIT_μAMPS [123] de NS-2 pour les réseaux de capteurs. Nous avons considéré une topologie de réseau de 100 capteurs, où chacun d'entre eux a une portée de transmission de 25 mètres. Les capteurs sont aléatoirement déployés sur une zone de superficie $100 \times 100m$ suivant une loi de distribution uniforme et la station de base est placée à la position (50,175). Au début de la simulation, tous les capteurs ont la même quantité d'énergie (2 Joules) et les simulations ont été exécutées dans deux contextes. Dans le premier contexte, les simulations s'exécutent jusqu'à ce que tous les capteurs aient épuisé leurs batteries, et dans le deuxième, elles s'arrêtent une fois que le taux de capteurs qui cessent de fonctionner, atteint 50%. En outre, nous avons réalisé ces simulations en utilisant deux valeurs distinctes pour le seuil $Thresh_{Upper} : 30, 50$, et une valeur fixe pour le seuil $Thresh_{Lower} = 15$. Les valeurs moyennes des paramètres de performances sont calculées après chaque période.

Comme nous l'avons mentionné ci-dessus, nous avons utilisé le même modèle énergétique discuté dans [48], dans lequel la consommation d'énergie concerne principalement les transmissions et les réceptions. Dans ce modèle, la consommation d'énergie de transmission a besoin d'énergie additionnelle pour amplifier la puissance du signal selon la distance séparant l'émetteur de la destination. Ainsi, pour transmettre un message de *k* bits à une distance *d*, la radio dépense l'énergie décrite par l'équation (5.1), où ϵ_{Elec} est l'énergie consommée par les circuits électroniques, $\epsilon_{friss-amp}$ et $\epsilon_{two-ray-amp}$ pour amplifier le signal. $\epsilon_{friss-amp}$ est utilisé par les capteurs lors de la formation des clusters ou par les membres d'un cluster, alors que $\epsilon_{two-ray-amp}$ est utilisé pour communiquer entre les cluster-heads dans le cas où ils se trouvent très éloignés, ou entre les cluster-heads et la station de base. La consommation d'énergie de réception est représentée par l'équation (5.2). Le tableau 5.1 récapitule les paramètres de simulation utilisés pour ces évaluations.

TAB. 5.1 – Paramètres de simulation pour l'évaluation de CSOS en terme de consommation d'énergie

Paramètres	Valeurs
Surface de déploiement	$(0,0) \times (100,100)$
Position de la station de base	$(50,175)$
Portée de transmission	$25\ m$
Vitesse de déplacement des capteurs	$[0,10]\ m/s$
ϵ_{elec}	50 nJ/bit
$\epsilon_{friss-amp}$	$10\ pJ/bit/m^2$
$\epsilon_{two-ray-amp}$	$0.0013\ pJ/bit/m^4$
$d_{Crossover}$	$87\ m$
Durée d'une période (round)	$20\ s$
Taille d'un paquet de données	$500\ octets$
Taille de l'en-tête d'un paquet	$25\ octets$
Energie initiale d'un capteur	2 Joules
Nombre de capteurs (N)	100
$\alpha - covrage$	50%
$Thresh_{Upper}$	30
$Thresh_{Lower}$	15

5.7.1 Impact de la mobilité sur LEACH et LEACH-C

Au début, nous avons réalisé des simulations pour illustrer l'impact de la mobilité des capteurs sur les performances de LEACH et sa variante LEACH-C.

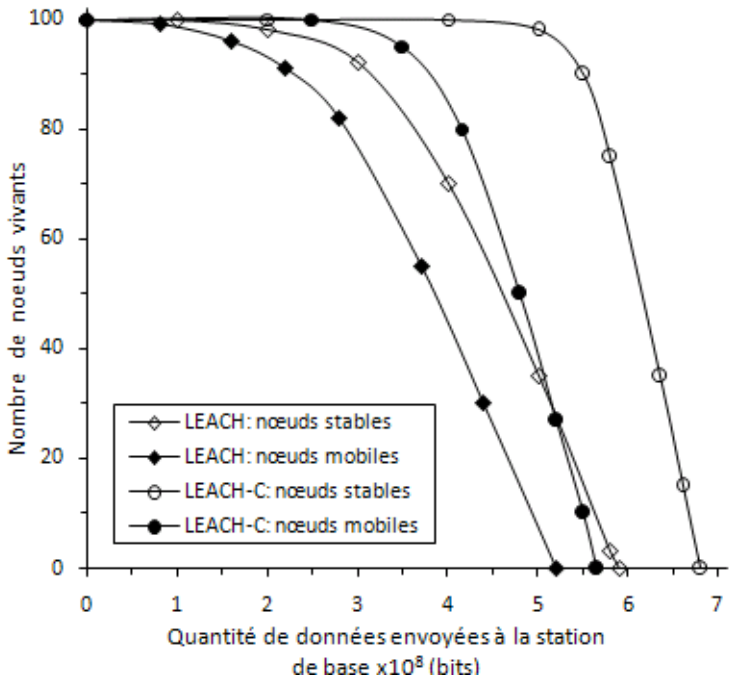


FIG. 5.1 – Nombre de capteurs vivants en fonction de la quantité de données envoyées à la station de base

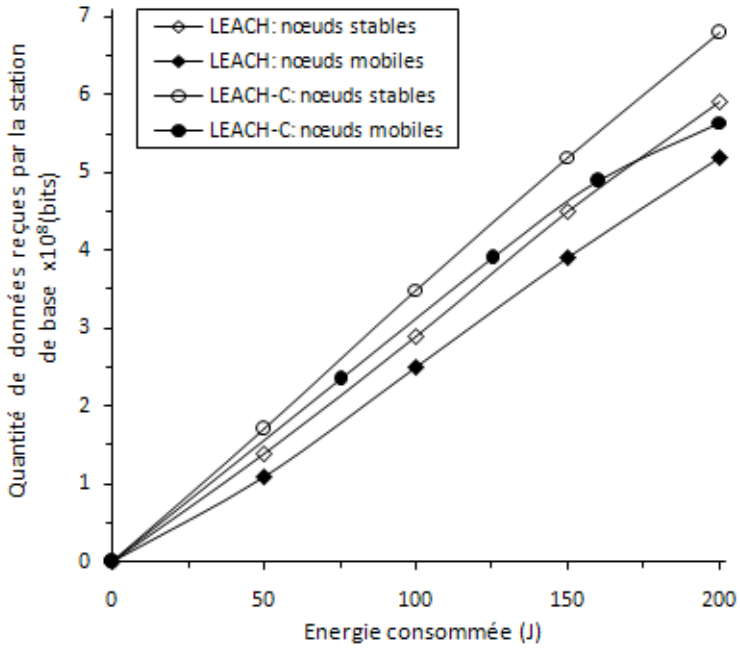


FIG. 5.2 – Quantité de données envoyées à la station de base en fonction de l'énergie consommée

Nous avons évalué les performances de LEACH et sa variante LEACH-C avec le même modèle décrit dans [48] avec deux scénarios : modèle avec des capteurs stationnaires et modèle avec des capteurs mobiles ; ceci dans le but de connaître l'impact de la mobilité des nœuds sur les performances de ces deux protocoles et d'évaluer leur robustesse face à la mobilité des nœuds. En effet, la figure 5.1 et 5.2 montrent que la mobilité des nœuds a un impact sur les performances de ces protocoles. Les performances de LEACH et LEACH-C dégradent respectivement de 14% et 21% en termes de quantité de données envoyées à la station de base durant la vie du réseau.

5.7.2 Premier contexte

Dans ce premier contexte, nous considérons que la durée de vie d'un réseau de capteurs suit la métrique LND i.e. quand le dernier capteur dans le réseau cesse de fonctionner [70, 71].

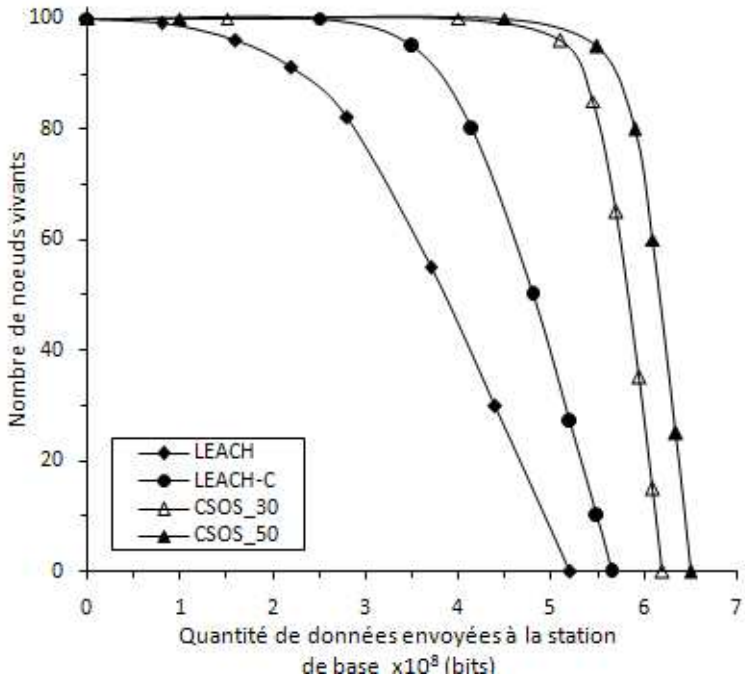


FIG. 5.3 – Comparaison du nombre de capteurs vivants en fonction de la quantité de données envoyées à la station de base dans un environnement mobile

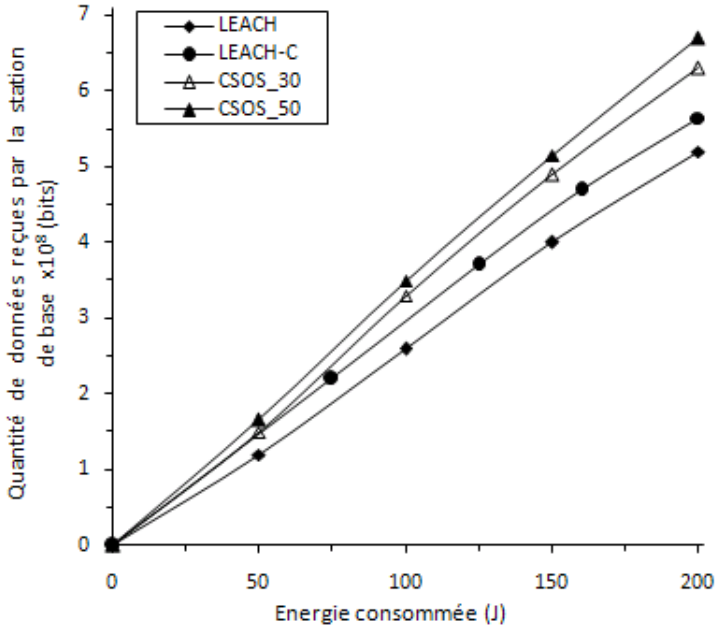


FIG. 5.4 – Quantité de données envoyées à la station de base en fonction de l'énergie consommée dans un environnement mobile

La figure 5.3 illustre que *CSOS_30* dépasse considérablement LEACH et légèrement LEACH-C en terme de quantité de données envoyées à la station de base durant la durée de vie du réseau tandis que *CSOS_50* les dépasse tous les deux largement. En effet, dans la figure 5.3, l'allure des courbes de *CSOS_30* et *CSOS_50* montre le nombre de capteurs vivants qui se dégradent rapidement à la fin de la simulation. Cela signifie que la différence de temps entre la mort du premier capteur et le dernier est très petite contrairement à LEACH où la mort des capteurs se dégrade graduellement pendant la durée de vie du réseau. Ceci est dû d'une part au choix aléatoire des cluster-heads et d'autre part au fait que les cluster-heads dépensent plus d'énergie pour envoyer un paquet de données à la station de base. D'autre part, la figure 5.4 montre que *CSOS_30* et *CSOS_50* dépassent LEACH et LEACH-C en terme de quantité de données envoyées à la station de base avec la même quantité d'énergie.

5.7.3 Deuxième contexte

Le problème de conservation d'énergie et de prolongement de la durée de vie dans les réseaux de capteurs tout en assurant une couverture entière de la zone d'intérêt est principalement contraint au taux de capteurs vivants dans le réseau.

Dans la plupart des applications, la durée de vie du réseau suit la métrique LND i.e. le temps qui s'écoule jusqu'à ce que le dernier capteur cesse de fonctionner. Cependant, cette métrique ne devrait pas être utilisée dans certaines applications critiques telles que les applications de détection et de surveillance d'intrusion, dans lesquelles le taux de couverture de la zone d'intérêt devrait être assez élevé. Ainsi, pour que le réseau accomplisse ses missions pour lesquelles il a été déployé avec succès, nous exigeons souvent que le taux de capteurs vivants soit supérieur à une certaine valeur seuil noté α-coverage. α-coverage représente le taux nécessaire de capteurs vivants pour garantir la couverture totale de la zone d'intérêt. Nous considérons que lorsque le taux de capteurs vivants descend sous le seuil α-coverage, le réseau deviendra dysfonctionnel.

Dans cette partie, nous supposons que le réseau accomplit ses missions avec succès tout en garantissant la couverture totale de la zone d'intérêt tant que α-coverage dépasse 50% [68]. Ainsi, nous considérons dans ce contexte que la durée de vie du réseau suit la métrique HNA.

L'environnement d'exécution des simulations est similaire à celui que nous avons présenté précédemment mais la simulation s'arrête une fois que α-coverage devient inférieure à 50%.

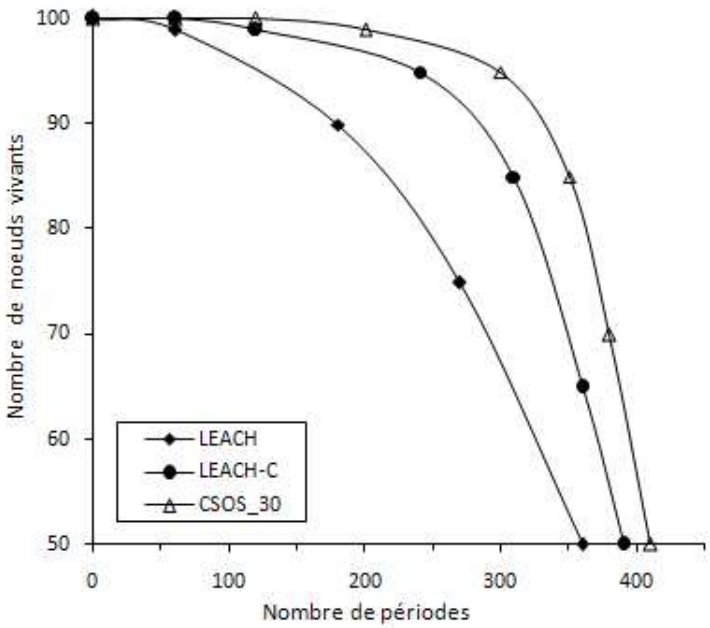


FIG. 5.5 – Durée de vie d'un réseau de capteurs dans un environnement mobile

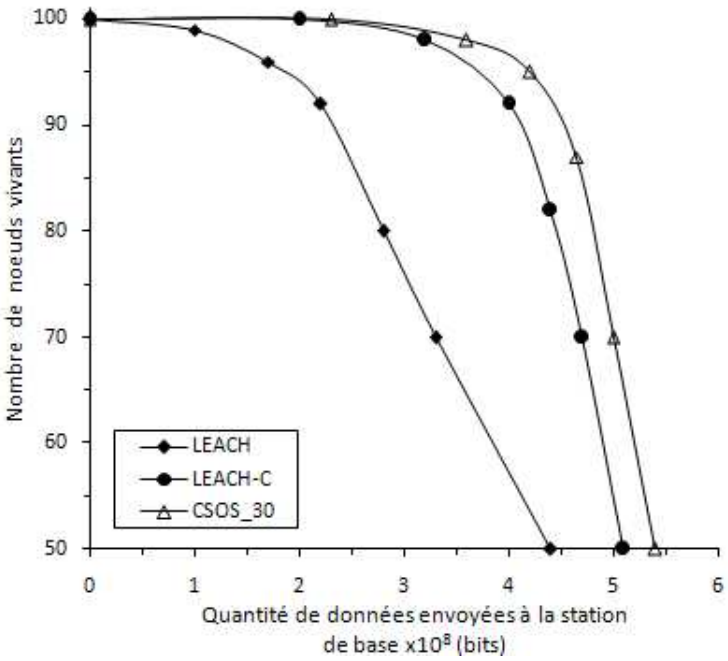


FIG. 5.6 – Comparaison du nombre de capteurs vivants en fonction de la quantité de données envoyées à la station de base dans un environnement mobile

La figure 5.5 montre que la durée de vie du réseau dure plus longtemps avec *CSOS_30* qu'avec LEACH et LEACH-C. Dans LEACH, le premier capteur cesse de fonctionner après 55 périodes (rounds), tandis qu'avec *CSOS_30*, il cesse de fonctionner après 210 périodes. Ceci est dû au choix aléatoire des cluster-heads par LEACH indépendamment de leur énergie restante, alors *CSOS_30* distribue équitablement la consommation d'énergie parmi les capteurs en choisissant toujours le capteur ayant plus d'énergie comme cluster-head. En outre, la différence de temps entre l'arrêt de fonctionnement du premier et du dernier capteur est petite contrairement à LEACH. Nous constatons aussi dans la figure 5.6 que *CSOS_30* envoie plus de paquets à la station de base durant la vie du réseau que LEACH et LEACH-C. Il dépasse respectivement LEACH et LEACH-C de 23% et de 13%.

5.8 Conclusion

Dans ce chapitre, nous avons évalué les performances de *CSOS* en termes de durée de vie et de quantité de données envoyées à la station de base pour illustrer l'apport de la génération de clusters stables et homogènes en taille et en rayon sur les performances d'un réseau de capteurs.

Au début, nous avons évalué la robustesse de LEACH et LEACH-C face à la mobilité des capteurs. Malheureusement, nous avons constaté que la mobilité des capteurs a un impact considérable sur les performances de ces protocoles. Puis, nous avons évalué les performances de CSOS et comparé les résultats obtenus à ceux obtenus par LEACH et LEACH-C dans un environnement mobile où les capteurs peuvent se déplacer à une vitesse dont la valeur est comprise entre 0 et 10 m/s. Nous avons constaté que *CSOS* dépasse LEACH et LEACH-C en termes de durée de vie et de quantité de données envoyées à la station de base.

En résumé, la stabilité des clusters générés par *CSOS* a un impact positif sur les performances des réseaux de capteurs. Les résultats obtenus ont prouvé que *CSOS* s'adapte à la topologie dynamique des réseaux de capteurs. D'où, *CSOS* peut être un bon candidat pour les réseaux de capteurs mobiles comparativement à LEACH et LEACH-C.

Chapitre 6

La couverture de zone dans les réseaux de capteurs

6.1 Introduction

On a vu que l'économie d'énergie et la prolongation de la durée de vie sont des tâches fondamentales dans les réseaux de capteurs. En outre, pour surveiller une zone importante on étend le réseau de capteurs pour qu'il ait une couverture maximale. Dans cette optique, plusieurs stratégies ont été proposées dans la littérature pour ordonnancer l'activité des capteurs déployés dans la zone d'intérêt afin d'assurer la couverture totale de la zone. Par conséquent, le choix des capteurs actifs dans le but d'économiser l'énergie et le maintien d'un niveau élevé de couverture sont deux aspects orthogonaux car la minimisation du nombre de capteurs actifs tout en garantissant un certain taux de couverture de la zone d'intérêt est un problème NP-difficile.

La couverture de zone peut prendre plusieurs formes. La plus simple lorsque tout point *p* de la zone des points cibles est couvert par un et un seul capteur i.e. tout point *p* se trouve dans la zone de détection d'un seul capteur. Dans ce cas on parle de la 1-couverture. En outre, on parle de la k-couverture $(k > 1)$ lorsque tout point *p* de la zone des points cibles est couvert par plus d'un capteur. Par ailleurs, quand les capteurs sont déployés en grand nombre dans une zone de points cibles, la couverture de tout point cible reflète la couverture de la zone des points cibles .i.e. si la zone de détection de tout capteur est couverte par un ensemble de capteurs actifs, la zone des points cibles est entièrement couverte. Dans ce chapitre, nous abordons le problème de sélection d'un ensemble de capteurs actifs dont la cardinalité est minimale parmi les capteurs déployés de telle sorte que ces capteurs assurent la k-couverture totale de la zone des points cibles. Le bon

choix de cet ensemble est essentiel, parce qu'il réduit la consommation d'énergie, et prolonge ainsi la durée de vie du réseau.

Notre contribution se situe dans deux aspects. Dans le premier, nous présentons un algorithme d'ordonnancement d'activité des capteurs, appelé CSA (Cluster-based Scheduling Algorithm) et sa version améliorée CSA_VS (Cluster-based Scheduling Algorithm - Virtual Sensor), pour maintenir la couverture de la zone des points cibles. CSA est basé sur l'algorithme CSOS que nous avons proposé dans le chapitre quatre. Puis, dans le deuxième aspect, nous évaluons le taux de couverture de la zone d'intérêt à l'aide de deux autres algorithmes : couverture de périmètre et capteur virtuel. Si ce taux est inférieur au taux exigé, nous pourrons améliorer notre algorithme distribué de telle sorte qu'il assure le taux de couverture exigé. Les résultats de simulation montrent d'une part que notre algorithme réalise le degré exigé de couverture pour différentes régions à l'intérieur de la zone d'intérêt, et d'autre part qu'il est plus performant que certains algorithmes de k-couverture centralisés et localisés.

Dans l'annexe de cette thèse, nous testons notre algorithme sur une application générique que nous avons développée au laboratoire pour la couverture des zones sensibles (risquées) dans le parc d'une clinique ou d'une maison de retraite où nous avons considéré que les capteurs étaient déployés sur des personnes dépendantes et à différents endroits sensibles du parc. Cette application s'appuie sur la plateforme de capteurs Crossbow.

6.2 Travaux existants

La couverture de zone d'intérêt peut prendre plusieurs formes suivant la nature des applications. Par exemple, dans les applications les moins sensibles telles que la surveillance des champs agricoles, nous pouvons concevoir des protocoles de couverture tels que chaque point dans la zone d'intérêt soit surveillé par un seul capteur et dans certains cas ne garantissent pas forcément la couverture totale de la zone d'intérêt. Dans ce cas on parle de la 1-couverture ou de couverture simple. Cependant, dans les applications sensibles telles que les applications militaires ou liées à la sécurité, il est nécessaire d'assurer la couverture de chaque point dans la zone d'intérêt par plus d'un capteur pour permettre la tolérance aux fautes. Dans ce cas, on parle de la k-couverture ou de couverture multiple.

Dans cette section, nous présentons les travaux existants liés à chacune des deux formes de couverture : la 1-couverture et la k-couverture.

6.2.1 La 1-couverture

Ye et al. [118] ont proposé un algorithme de couverture de zone appelé PEAS (Probing Environment and Adaptive Sleeping) pour les réseaux de capteurs asynchrones. Initialement, tous les capteurs sont dans l'état passif et après l'écoulement d'une certaine période, si un capteur veut passer au mode actif, il envoie alors un message dit de sondage à ses 1-voisins. Ces derniers évaluent la distance qui les sépare du capteur émetteur en fonction de la puissance du signal reçu ou du délai de transmission. Si cette distance est inférieure à une certaine distance seuil P, ses 1-voisins lui demandent de rester toujours dans l'état passif puisqu'ils se trouvent tout près du capteur émetteur et par conséquent ils couvrent sa zone de détection. Dans le cas contraire, le capteur émetteur ne reçoit aucun message de ses 1-voisins et décide ainsi de passer en mode actif qu'il garde jusqu'à épuisement de sa batterie. Dans PEAS, un nœud actif demeure éveillé jusqu'à ce qu'il subisse une défaillance ou que sa batterie soit épuisée. Ensuite, les nœuds en mode sommeil remplacent alors les nœuds défaillants si nécessaire, ce qui rend PEAS tolérant aux fautes. Cette technique peut ne pas être souhaitable, car la densité des nœuds actifs se dégradera en fonction du temps. En outre, la défaillance de nœuds peut causer la division du réseau en sous réseaux non connectés et créer des nœuds isolés. Par ailleurs, dans un environnement où la cause principale de la défaillance des capteurs est l'épuisement des batteries, il est souhaitable d'équilibrer la consommation d'énergie parmi les nœuds du réseau. Pour cela, la sélection des nœuds actifs devrait se faire périodiquement et sur la base de plusieurs facteurs tels que l'énergie et la k-densité. En outre, PEAS ne permet pas de garantir une couverture totale de la zone d'intérêt sauf si une relation étroite est établie entre la distance seuil P et la portée de détection R_s.

Dans [46], Gui et Mohapatra ont proposé une extension au protocole PEAS appelé PECAS (Probing Environment and Collaborating Adaptive Sleeping) pour pallier aux limites de PEAS. Dans PECAS, un nœud demeure dans le mode actif seulement pour une durée indiquée par le paramètre Work_Time_Dur contrairement à PEAS où un nœud actif demeure éveillé jusqu'à ce qu'il subisse une défaillance ou qu'il épuise sa batterie. En outre, chaque nœud a une variable temporisateur appelé Next_Sleep_Time, qui est utilisée pour indiquer au nœud le temps restant avant de passer en mode sommeil. Ainsi, quand un nœud passe au mode actif, il initialisera la variable Next_Sleep_Time par le paramètre Work_Time_Dur. Cette variable se décrémente en fonction du temps jusqu'à l'expiration du temporisateur (Next_Sleep_Time=0).

Cai et al. [19] ont développé un protocole de couverture de zone pour les réseaux de capteurs asynchrones appelé ACOS (Area-based Collaborative Sleeping). Ce protocole améliore les performances de PECAS [46] et introduit le concept de collaboration dans le but d'équilibrer la consommation d'énergie parmi les capteurs. Dans ce protocole, un capteur peut être dans l'un des quatre états suivants : passif, actif, pré-actif, ou pré-passif et chaque capteur *u* connaît la portion de sa surface qui n'est pas couverte par aucun autre capteur (exclusivement par lui-même). En outre, dans ACOS, les auteurs ont considéré deux niveaux de consommation d'énergie : low-power et consuming-power. Le premier mode de consommation d'énergie correspond à l'état actif tandis que le deuxième correspond aux autres états du capteur.

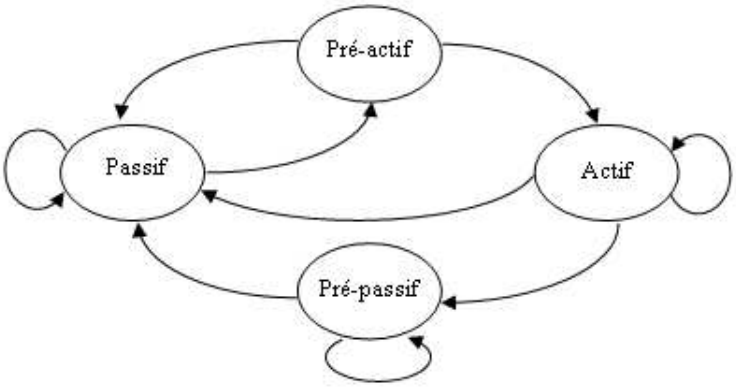


FIG. 6.1 – Les différents modes d'activité d'un capteur

La figure 6.1 résume les différentes transitions d'un capteur aux différents modes. Ainsi, tout capteur *u* en mode passif peut passer à l'état actif sur sa propre décision et en alertant ses voisins par l'envoi d'un message de sondage. Quand le nœud *u* se réveille, son statut passe du passif au pré-actif. Ensuite, *u* envoie un message de sondage appelé "PreWakeUp_Msg" à ses voisins et attend T_w secondes. Tout voisin actif s_j de *u* lui répond par un message "Reply_PreWakeUp_Msg" incluant sa position et le temps d'activité restant. Le nœud *u* extrait des messages reçus de ses voisins leurs positions et leurs temps d'activité restant. Á l'expiration de la durée T_w, le nœud *u* évalue la portion de sa surface qui n'est pas couverte par aucun autre capteur. Si cette portion est inférieure à une certaine surface seuil, *u* passe au mode passif et initialise ainsi sa durée de sommeil par le minimum des temps d'activité restant à ses voisins. Ce mécanisme permet d'éviter l'apparition de plusieurs régions vides dans le cas où plusieurs capteurs se trouvant dans le mode pré-passif. Cependant, il est possible que certains voisins de s_j et qui ne sont pas voisins de *u* se réveillent en même temps avec *u*. Par conséquent, le nœud *u* et les voisins de ses voisins peuvent

passer en mode actif en même temps. Ainsi, pour éviter cette situation, les auteurs ont ajouté un temps aléatoire au temps de sommeil de *u*. Dans cet algorithme, il est difficile de coordonner entre les capteurs se trouvant dans l'état pré-passif et qui veulent passer en mode passif.

Dans [50], les auteurs ont présenté un algorithme de sommeil aléatoire et coordonné pour assurer la couverture d'une zone d'intérêt. Dans cet algorithme, un capteur ne peut passer en mode passif que s'il obtient une permission de ses voisins appelés voisins parrains (un capteur parrain est un capteur actif). Cette permission contient la période durant laquelle le capteur demeurera dans l'état passif. Par ailleurs, avant l'envoi d'une permission, il y a des échanges d'informations entre les capteurs concernant le niveau d'énergie de leurs batteries. Ainsi, un capteur se trouvant dans l'état actif et disposant d'un niveau d'énergie faible a une tendance à devenir passif contrairement à un capteur ayant plus d'énergie. Quoique cet algorithme implique le critère énergétique pour choisir les nœuds parrains et qu'il ait un aspect distribué, il pourrait générer un grand nombre de capteurs parrains ce qui aura un impact sur la durée de vie du réseau de capteurs.

Dans [95], Sheu et al. ont proposé un algorithme localisé basé sur la priorité des capteurs pour la sélection des capteurs actifs connaissant les priorités de leurs voisins. Initialement, tout capteur *u* envoie sa priorité à ses voisins se trouvant dans sa zone de détection. Ensuite, le capteur *u* considère le périmètre de sa zone de détection et les portions de périmètre de ses voisins de plus haute priorité se trouvant dans sa zone de détection. Si ces portions sont couvertes par d'autres voisins de plus haute priorité alors le capteur u pourrait passer en mode passif. Pour décider de l'activité des voisins, chacune de ces portions de périmètre est divisée en segments, joignant les points d'intersection avec les autres périmètres des zones de détection. Pour chaque segment formé, le capteur ayant la plus haute priorité et couvrant ce segment devient actif. Par ailleurs, l'ensemble des capteurs actifs suffit pour construire un arbre connecté servant à acheminer l'information de tout capteur *u* à la station de base.

Dans [102], les auteurs ont proposé une technique pour la couverture de zone. Pour cela, ils ont supposé que tous les capteurs ont le même rayon de détection et de communication et que la portée de détection est égale à la portée de communication Au début, chaque nœud *u* envoie un message HELLO et établit la liste de ses 1-voisins. Ensuite, le nœud *u* évalue les surfaces couvertes par chacun de ses 1-voisins u_i en utilisant la méthode des périmètres. Si l'ensemble des surfaces couvertes des 1-voisins u_i de *u* couvre la zone de détection de *u* pendant une durée aléatoire , le nœud *u* peut devenir passif et par suite il envoie un message de retrait. Tout voisin

de *u* recevant ce message et n'ayant pas décidé son statut, retire *u* de ses 1-voisins pour ne pas le prendre en considération lors de l'évaluation des zones couvertes par ses 1-voisins actifs. Le processus se répète jusqu'à ce que chaque capteur dans le réseau décide son statut. Dans cette technique, si un nœud actif disparaît sans avertir son voisinage i.e. lorsque sa batterie est épuisée ou s'il subit une défaillance, alors il pourrait être considéré lors de l'évaluation des zones couvertes par ses 1-voisins actifs. Les auteurs ont discuté le problème de couverture de zones par les 1-voisins lorsque les capteurs présentent des portées de détection différentes mais ceci n'a aucun effet sur le traitement des limites de la technique présentée.

Dans [55], les auteurs ont présenté plusieurs solutions qui peuvent être utilisées pour traiter les limites de la technique [102]. Pour cela, ils ont supposé et appliqué le critère suivant : un disque D est entièrement couvert si les périmètres des autres disques le couvrant sont complètement couverts par d'autres disques couvrants. Ainsi, si nous utilisons une version améliorée de la technique [102], il suffit de vérifier le critère présenté dans [55] pour tout nœud du réseau et dans l'affirmative introduire une durée aléatoire d'attente avant la prise de décisions d'activité par le nœud.

Dans [121], Zhang et J. Hou ont proposé un algorithme géographique basé sur le contrôle de densité (Optimal Geographic Density Control). OGDC peut configurer un réseau de capteurs de telle sorte qu'il assure la couverture totale de la zone des cibles, la connectivité du réseau et la conservation d'énergie. La conservation d'énergie est basée sur le contrôle de la densité des capteurs actifs : plus les capteurs actifs sont bien distribués sur la zone de déploiement plus l'énergie est conservée dans le réseau. En outre, OGDC suppose que la densité des capteurs est assez élevée et que les portées de détection des capteurs peuvent être différentes. Il suppose également que chaque capteur connaît sa propre position et tous les capteurs sont synchronisés en temps. Les auteurs ont montré que la couverture globale de la zone surveillée permet aussi la connectivité du réseau si et seulement si $R_c \geq 2 \times R_s$ où R_c est le rayon de communication d'un capteur et R_s est son rayon de détection, et que OGDC permet d'optimiser le nombre de capteurs actifs et par conséquent il permet de réduire la consommation d'énergie. OGDC est exécuté après chaque période au cours de laquelle chaque capteur prend une nouvelle décision d'activité en choisissant d'être actif ou passif. Chaque période inclut une phase de sélection de nœuds candidats et une phase d'état stable. Quand $R_c < 2 \times R_s$, OGDC s'exécute comme suit. Au cours de la phase de sélection de nœuds, un nœud aléatoire initie le processus d'ordonnancement d'activité de capteurs. Ce nœud passe en mode actif et diffuse un message "ON" dans son voisinage. Quand un nœud reçoit ce message, il initialisera son temporisateur en fonction de la distance qui le sépare

de l'émetteur. Après l'expiration du temporisateur, si ce dernier ne reçoit pas de messages des autres nœuds, il passera en mode actif et il diffusera à son tour le message "ON" dans son voisinage. De cette manière, le processus d'ordonnancement d'activité de capteurs se propage dans le réseau. Ce qui permet de favoriser la sélection les capteurs situés au plus près des points d'un pavage hexagonal de la zone. Cependant, les auteurs n'ont pas montré comment se fait le choix du capteur qui va initier le processus d'ordonnancement d'activité et comment peut-on résoudre le problème de conflit si plusieurs capteurs initient en même temps le processus de sélection de nœuds actifs. En plus, le processus utilisé pour mettre en mode actif les capteurs qui sont censés assurer la couverture de la zone, ne tient pas compte de la capacité d'un capteur à réaliser cette tâche. En outre, la phase de sélection de nœuds actifs génère une latence importante puisqu'un seul capteur est choisi pour commencer le processus d'ordonnancement d'activité.

A partir des travaux précédents, nous avons présenté des protocoles qui assurent la 1-couverture de zone de cibles. Or, si la zone présente certaines régions sensibles qui nécessitent une surveillance permanente, il est donc indispensable de surveiller cette zone par plusieurs capteurs pour impliquer la tolérance aux fautes et éviter l'apparition de points vides dans ces régions sensibles quand un capteur cesse de fonctionner. On parle donc dans ce type d'application de la k-couverture.

6.2.2 La k-couverture de zone

Dans la plupart des travaux, le problème de k-couverture consiste à trouver un nombre minimal de nœuds où tout point de la zone d'intérêt est couvert par au moins *k* nœuds. Dans cette optique, plusieurs approches ont été proposées dans la littérature. Dans [20, 96], les auteurs choisissent un sous-ensemble de nœuds aléatoirement pour maintenir la couverture de zone tout en visant à économiser l'énergie globale du réseau. Cependant, avec cette technique, il est difficile d'être sûr que la zone d'intérêt est entièrement couverte. Dans [1, 112], les auteurs ont proposé des protocoles distribués basés sur le clustering pour maintenir la k-couverture de la zone. Cependant, l'overhead de communication généré par ces protocoles est très important.

Huang et Tseng [51] ont étudié le problème de la k-couverture d'une zone de cibles. Ils ont considéré que la zone de détection de chaque capteur est modélisée par un disque unitaire et que la zone des cibles est entièrement k-couverte si le périmètre de la zone de détection de chaque capteur est k-couvert. Cependant, la complexité dans le pire des cas de l'algorithme proposé en terme de temps d'exécution est de l'ordre $\theta\left(n^2 * log\, n\right)$ où n représente le nombre de capteurs

dans le réseau.

Dans [97], So et Ye ont utilisé le concept des diagrammes de Voronoï d'ordre k [84] pour élaborer un algorithme de vérification de la k-couverture. Ils ont montré que si tous les sommets des diagrammes de Voronoï sont couvets alors la zone des cibles est entièrement couverte. Cependant, la complexité de cet algorithme en terme de temps d'exécution est importante puisqu'elle est calculée en fonction du temps de construction des diagrammes de Voronoï qui est de l'ordre de $\theta(n * logn + n * k^2)$ [62] et du temps de vérification de la k-couverture.

Dans ces deux travaux que nous avons présentés, les auteurs ne proposent pas d'algorithmes distribués et n'abordent pas le problème de la k-couverture avec *k* aussi grand, pourtant la complexité des algorithmes proposés en termes temps d'exécution est si importante. En outre, les auteurs supposent que les zones de détection des capteurs suivent le modèle de disque unitaire, mais en réalité ces zones ont des formes irrégulières.

Dans [103], Tian et Georganas ont proposé une extension de l'algorithme présenté dans [102] pour traiter le problème de préservation de la k-couverture et la k-connexité dans les réseaux de capteurs. Cependant, cette extension nécessite un grand nombre de messages de contrôle et suppose toujours que $R_c \geq 2 * R_s$.

Zhou et al. [122] ont présenté un algorithme glouton et deux heuristiques pour résoudre le problème de couverture de zone par des ensembles connectés. L'algorithme glouton consiste à trouver un ensemble de capteurs appelé M pour assurer la couverture de la zone d'intérêt tel que le graphe de communication induit par M soit connecté. Cet algorithme s'exécute comme suit. Initialement, un capteur aléatoire c_a est placé dans M tel que la zone de détection de c_a coupe la zone d'intérêt R_Q. Ensuite, ce nœud doit sélectionner un capteur candidat et un chemin candidat pour assurer conjointement la couverture et la connexité. Un capteur *c* est dit capteur candidat si $c \notin M$ et il existe $m \in M$ tel que $d(c, m) < R_s(c) + R_s(m)$. Un chemin est dit chemin candidat s'il existe une séquence de capteurs $< p_0, p_1, p_2, ..., p_l >$ tel que p_0 est un capteur candidat, $p_i \notin M$ pour $i < l$, $p_l \in M$ et la séquence de capteurs forme un chemin de communication. Pour sélectionner un capteur candidat, le capteur initialement mis dans M, diffuse un message de recherche dans son voisinage à $2 \times r$ sauts tel que *r* est le nombre maximum de sauts séparant deux nœuds partageant une région de détection. Au début, cette recherche concerne le résultat d'une diffusion à deux sauts et après l'analyse des messages reçus par le capteur, un nouveau capteur est ajouté à l'ensemble M. Le processus de recherche des capteurs candidats et de che-

mins candidats se répète jusqu'à ce que la zone soit k-couverte. Cependant, la complexité de cet algorithme glouton est très élevée en terme de messages échangés entre les capteurs pour construire l'ensemble M. En outre, les messages échangés peuvent être corrompus et leurs tailles sont potentiellement larges. Pour remédier à cette limitation, Zou et al. ont proposé une version distribuée de cet algorithme glouton. Dans cette version, la taille des messages échangés entre les nœuds est petite et fixe. Aussi, chaque nœud doit collecter les informations se trouvant à $Max\,(t, r)$ sauts où *t* est une constante fixée à 2 pour dire qu'il faut rechercher le capteur candidat à partir de deux sauts et plus, et *r* est calculée en fonction de la densité. Par conséquent, *r* aura une valeur si élevée dans les réseaux denses. D'où, une complexité en termes de messages échangés qui reste toujours importante. De plus, cette solution ne peut pas être appliquée aux réseaux de capteurs car les capteurs disposent d'une mémoire très réduite pour stocker les informations de tous les voisins se trouvant à *r* sauts d'un nœud donné.

Dans [107], Wang et al. ont proposé un protocole distribué de configuration de couverture (CCP) capable de fournir différents degrés de couverture exigés par des applications. CCP est basé sur l'algorithme d'éligibilité de k-couverture. Cet algorithme est exécuté par chaque capteur pour décider s'il devrait devenir actif ou non (par exemple le nœud c dans la figure

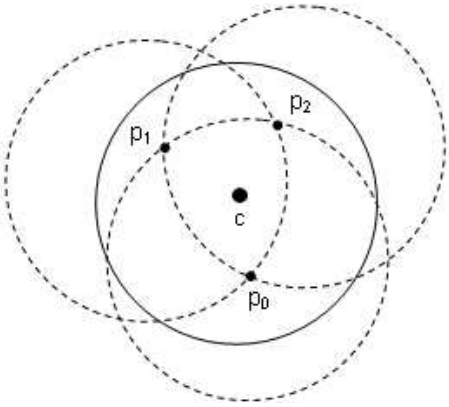


FIG. 6.2 – Exemple d'éligibilité de nœud

Pour l'évaluation de la k-couverture d'une zone de cibles, CCP se base sur le concept suivant : si les points d'intersection entre tous les disques unitaires sont k-couverts, alors la zone des cibles est entièrement k-couverte. Puis, dans [108], Wang et al. ont proposé une version améliorée de CCP dans le but d'économiser l'énergie. Cependant, CCP et sa version améliorée présentent plusieurs limites. La complexité dans le pire des cas de l'algorithme utilisé par CCP

et son extension est de l'ordre de $\theta\left(d^3\right)$ où *d* est le nombre maximal de voisins d'un capteur dans le réseau. En plus, CCP et son extension supposent que les capteurs connaissent leurs positions, et que le mécanisme d'ordonnancement d'activité des capteurs ne permet pas d'assurer que le nombre de capteurs mis en mode actif est minimal.

Dans [113], Yang et al. ont traité conjointement le problème de k-couverture (k-CS) et le problème de k-connecté couverture (k-CCS) en utilisant la technique de programmation d'entiers (IP) puis la programmation linéaire basée sur un algorithme d'approximation (LPA). Dans la formulation de ces problèmes, les auteurs ont impliqué les coefficients (a_{ij}) et les variables booléennes (x_i) qui explicitent respectivement la relation de couverture entre deux capteurs c_i et c_j, et l'appartenance d'un capteur c_i à l'ensemble C (ensemble des capteurs actifs). Les coefficients, les variables et le problème de programmation d'entiers (IP) sont définis comme suit : si c_i est couvert par c_j alors $a_{ij} = 1$ sinon $a_{ij} = 0$, et si c_i appartient à C alors $x_i = 1$ sinon $x_i = 0$. Puis, le IP représenté par l'équation (6.1) sera résolu.

$$\begin{cases} Min\left(x_1 + x_2 + ... + x_n\right) \\ \sum_{j=1}^{n} a_{ij} x_j \geq k \qquad \textit{pour} \quad i, j = 1, 2, ..., n \\ x_j \in \{0, 1\} \end{cases} \tag{6.1}$$

La contrainte $\sum_{j=1}^{n} a_{ij} x_j \geq k$ garantit que tout capteur c_i du réseau est couvert par au moins *k* capteurs de C. Par ailleurs, puisque IP est NP-difficile, les auteurs ont proposé un algorithme d'approximation basé sur la programmation linéaire (LPA). Dans LPA, Yang et al. ont supposé que $0 \leq x_j \leq 1$ et que la solution optimale de x_j^* soit calculée comme suit :

Si $x_j^* \geq \frac{1}{\rho}$ alors $\bar{x_j} = 1$ et $C = C \cup \{s_j\}$
Sinon $\bar{x_j} = 1$
avec $\rho = (\Delta + 1)$ où Δ est le degré maximal dans le réseau.

Cependant, la complexité de LPA est très importante car elle est dominée par la résolution de LP. La meilleure résolution de LPA est fournie par l'algorithme de YE [114] et sa complexité est de l'ordre de $\theta\left(n^3\right)$. De ce fait, le temps d'exécution de LPA est très important. Pour pallier à cette limitation, Yang et al. ont proposé une approche quasi locale appelée CKA (Cluster-based Algorithm). CKA est basée sur le concept de clustering qui devra être exécuté *k* fois. Les cluster-heads sélectionnés après chaque itération sont marqués et enlevés du réseau. Ces cluster-heads doivent être connectés par des nœuds gateways qui sont également marqués. Ainsi, pour

tout nœud marqué (cluster-head ou gateway), s'il n'a pas k voisins marqués, il désigne d'autres nœuds non marqués dans son voisinage pour être marqués de telle sorte que le nombre de voisins marqués dans son voisinage soit *k*. La complexité de la solution proposée reste toujours importante. Elle est de l'ordre de $\theta\left(k * log^3 n\right)$. Pour améliorer les performances de CKA en termes de complexité, Yang et al. ont proposé une solution locale basée sur l'information locale dans 2-voisinage, appelée PKA (Pruning-based k-CS/k-CCS Algorithm). Cette solution s'exécute comme suit. Initialement, à tout nœud *u* est attribuée une priorité $L\left(u\right)$, et il est représenté par un couple $\left(ID\left(u\right), L\left(u\right)\right)$ où $ID\left(u\right)$ est son identifiant. Ensuite, tout nœud *u* diffuse la liste de ses 1-voisins $\left(\mathrm{N}_1\left(u\right)\right)$ et construit son sous-ensemble $C\left(u\right) = \left\{v \,|v \in N_1\left(u\right) \wedge L\left(v\right) > L\left(u\right)\right\}$. Le nœud *u* ne sera pas marqué si $C\left(u\right)$ est connecté par des nœuds ayant une priorité élevée à celle de *u* (cette contrainte n'est pas prise en considération si la connectivité n'est pas traitée), et si pour tout voisin *w* de *u* $\left(w \in N_1\left(u\right)\right)$, il existe *k* nœuds distincts $\left(v_1, v_2, ..., v_k\right)$ dans $C\left(u\right)$ tel que $w \in N_1\left(v_i\right)$. Dans cette approche, les auteurs ont impliqué une priorité qui a un aspect abstrait dans le choix des nœuds qui doivent garantir la couverture de la zone d'intérêt.

Dans [22], Carle et Simplot-Ryl ont présenté un protocole basé sur l'ensemble dominant pour maintenir la couverture de surface, appelé ADS (Area Dominating Set Protocol). ADS est une version améliorée de DS et CDS qui assurent la couverture de nœud. Il consiste à sélectionner un nombre minimal de capteurs pour assurer simultanément la couverture globale de la zone des cibles et la connectivité du réseau de capteurs tout en conservant l'énergie déployée. Cette approche est basée sur le protocole de construction d'ensemble dominant connecté (CDS). Un CDS est un sous ensemble de nœuds connectés tels que chaque nœud du graphe est soit dominant ou voisin d'un nœud dominant .i.e. soit il appartient à CDS ou voisin d'un nœud du CDS. Il est généré par le processus de marquage présenté dans [109, 110] où chaque nœud ayant deux voisins non connectés est choisi comme dominant. Une règle *k* a été ajoutée à CDS pour générer un k-CDS [34] dans le but de minimiser le cardinal de CDS construit initialement. Cette règle consiste à enlever tout nœud de CDS qui est couvert par *k* autres dominants. La complexité de cette approche est de l'ordre de $\theta\left(d^2\right)$ où *d* est le nombre maximal de voisins d'un nœud dans le réseau. Par ailleurs, ADS utilise son propre algorithme de couverture de surface puisque CDS n'était pas conçu pour assurer la couverture de zone. Cet algorithme est exécuté comme suit : un nœud *u* établit l'ensemble de ses voisins et calcule la surface couverte $S\left(u\right)$ par chacun de ses voisins. Puis, le nœud *u* détermine le sous-graphe sous-jacent de ses 1-voisins. Si le sous-graphe est connexe et les nœuds du sous-graphe couvrent entièrement la zone de détection du nœud *u*, le nœud *u* opte pour le statut sommeil ; autrement, il choisit le statut actif. La complexité de

l'algorithme de couverture utilisé par ADS est de l'ordre de $\theta(d^3)$ parce que cet algorithme se base sur l'algorithme de construction de CDS dont la complexité est $\theta(d^2)$.

6.3 Contexte du problème de couverture

Les capteurs sont déployés généralement en grand nombre et leurs régions de détection sont superposées par les capteurs voisins (adjacents), ce qui mène à un grand nombre de capteurs superflus. En conséquence, nous pouvons mettre en mode actif un nombre minimal de capteurs capables de garantir une couverture totale de la zone d'intérêt et éteindre la radio des capteurs superflus. Ceci dans le but de réduire la consommation d'énergie globale du réseau et de prolonger la longévité des capteurs. Par ailleurs, dans les réseaux de capteurs, la détermination d'un nombre minimum de capteurs actifs se base sur le taux de couverture exigée et de son degré. On dit qu'un réseau de capteurs a un degré de couverture d'ordre *k* (k-couverture) si et seulement si chaque point dans la région de son déploiement est couvert par au moins *k* capteurs. Cependant, quand le taux de couverture de la région n'atteint pas 100% alors on parle de l'existence de points vides de détection. L'apparition de ces points vides peut être due aux défaillances ou à l'extinction de la radio de certains capteurs.

Nous considérons un ensemble de capteurs, $C = \{ c_1, c_2, ..., c_n \}$, déployés dans une zone d'intérêt S euclidienne bidimensionnelle. Chaque capteur c_i est placé dans S aux coordonnées (x_i, y_i) et a une portée de détection (surveillance) R_s i.e. qu'il peut surveiller n'importe quelle cible se trouvant à une distance inférieure ou égale à R_s.

6.3.1 Préliminaires

Pour faciliter la description de l'approche que nous avons proposée, nous présentons les définitions et les notations utilisées par notre approche et les contextes sur lesquelles se base notre approche :

- Définition 1 : Zone de surveillance

La zone de surveillance ou de détection d'un capteur c_i placé à (x_i, y_i) est représentée par la surface $S(c_i) = \{ X \in P \ | d(X, X_i) \leq R_s \}$ où $d(X, X_i)$ est la distance euclidienne séparant tout point du plan P du capteur c_i.

- Définition 2 : Couverture de zone

On dit qu'un ensemble de capteurs $C = \{ c_1, c_2, ..., c_n \}$ couvre entièrement une zone de cibles S si et seulement si $S \subseteq S(C)$ tel que $S(C) = S(c_1) \cup S(c_2) ... \cup S(c_n)$ (figure 6.3).

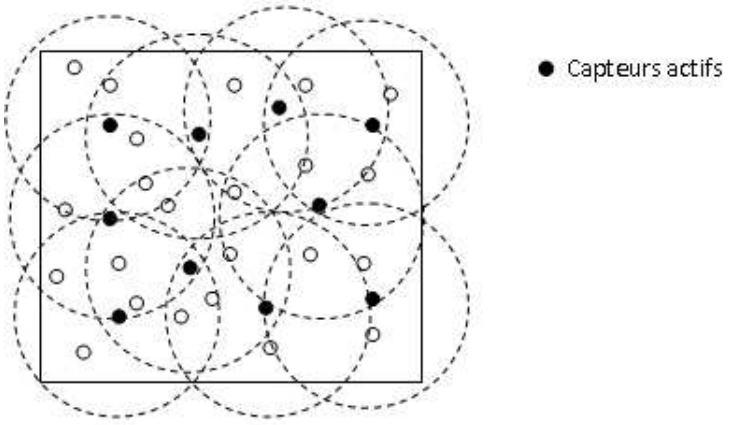


FIG. 6.3 – Couverture de zone d'intérêt

- Définition 3 : 1-couverture d'un point

Un point p dans S serait couvert par un capteur c_i s'il se trouve dans la portée de détection de c_i, notée $R_s(c_i)$:

$$p \in S \text{ est couvert par } c_i \Leftrightarrow d(p, c_i) \leq R_s(c_i) \quad \text{ou } p \in S(c_i)$$

On définit l'ensemble de capteurs qui couvrent un point p par :

$$\begin{aligned} Couvert(p) &= \{ c_i \in C \mid d(p, c_i) \leq R_s(c_i) \} \\ &= \{ c_i \in C \mid p \in S(c_i) \} \end{aligned}$$

Si l'ensemble $Couvert(p) \neq \phi$ ou $|Couvert(p)| \neq 0$ alors le point p est dit couvert sinon c'est un point vide dans la zone d'intérêt.

- Définition 4 : k-couverture d'un point

Un point *p* dans *S* est k-couvert s'il se trouve dans les portées de détection d'au moins *k* capteurs :

$$p \in S \text{ est k-couvert} \Leftrightarrow |Couvert(p)| \geq k$$

- Définition 5 : 1-couverture de zone

Une zone de déploiement de capteurs *S* est dite couverte si et seulement si tout point p appartenant à cette zone $(p \in S)$ est couvert par au moins un capteur de l'ensemble $C = \{c_1, , c_2 , ..., c_n\}$:

$$S \text{ est dite couverte} \Leftrightarrow \forall p \in S \; Couvert(p) \neq \phi$$
$$\text{ou } \forall p \in S \; |Couvert(p)| \neq 0$$

- Définition 6 : k-couverture de zone

Une zone de déploiement de capteurs *S* est dite k-couverte si et seulement si tout point *p* appartenant à cette zone est k-couvert.

$$S \text{ est k-couverte} \Leftrightarrow \forall p \in S \quad |Couvert(p)| \geq k$$

- Contexte 1.

Etant donné un nombre *k*, le problème de k-couverture est un problème de décision dont le but est de déterminer si tous les points dans *S* sont k-couverts ou non.

- Contexte 2.

On définit la k-couverture d'une zone d'intérêt où *k* représente le nombre minimal de capteurs que couvre tout point *p* de la zone *S*.

$$k = Min(|Couvert(p)| \;\; \forall p \in S)$$

6.4 Algorithme d'ordonnancement d'activité des capteurs (CSA)

Pour l'ordonnancement d'activité des capteurs, nous proposons un algorithme distribué appelé CSA (Cluster-based Scheduling Algorithm) et une version améliorée de cet algorithme appelée CSA_VS (Cluster-based Algorithm - Virtual Sensor). CSA sélectionne les capteurs qui vont passer en mode actif après l'élection des cluster-heads et la formation des clusters suivant l'algorithme d'auto organisation CSOS présenté dans le chapitre 3. Dans CSOS, nous avons traité le problème de la couverture de nœud alors que dans CSA, nous allons généraliser cet aspect et aborder le problème de couverture de zone. Par ailleurs, les capteurs qui assurent la couverture de la zone d'intérêt, seront choisis d'une manière distribuée et en fonction de leur poids par leurs cluster-heads correspondants. CSA s'exécute en deux phases. La première phase s'exécute selon le schéma algorithmique suivant :

- C_i : le cluster i,
- $N_2(CH, C_i)$: les 2-voisins du cluster-head CH qui appartiennent au cluster C_i,
- $Couvert(C_i)$: ensemble des capteurs du cluster C_i qui sont mis en mode actif pour maintenir la couverture de la zone d'intérêt S.
- Un capteur se met en mode actif selon son poids.

Pseudo-code de l'algorithme CSA

Pour chaque cluster-head (CH) **faire**

Début

$N_2^{'}(CH, C_i) = N_2(CH, C_i)$

$Couvert(C_i) = \{CH\}$

Fin Pour

/* Mettre en mode actif les 2-voisins isolés de CH et les 1-voisins de CH qui peuvent couvrir ces nœuds

Tant que $\exists v \mid v \in N_2^{'}(CH, C_i) \ \wedge \ \exists! u \in N_1(CH, C_i) \ | v \in N_1(u, C_i)$ **faire**

$Couvert(C_i) = Couvert(C_i) \cup \{u, v\}$

$N_2^{'}(CH, C_i) = N_2^{'}(CH, C_i)/\{v\}$

Fin Tant que

Tant que $N_2^{'}(CH, C_i) \neq \phi$ **faire**

- Choisir $u \in N_1(CH, C_i)$:

  $Weight(u) = Max(Weight(u_i) : \ u_i \in N_1(CH, C_i))$

- $Couvet(C_i) = Couvert(C_i) \cup \{u\}$
- Choisir $v \in N_1(u, C_i)$:

  $Weight(v) = Max(Weight(v_i) : \ v_i \in N_1(u, C_i) \ \wedge \ v_i \notin N_1(CH, C_i))$

- $Couvet(C_i) = Couvert(C_i) \cup \{v\}$

Pour tout $w \in N_1(u) \ \wedge \ w \notin N_1(v)$ **faire**

$Couvert(C_i) = Couvert(C_i) \cup \{w\}$

Fin Pour

- $N_2^{'}(CH, C_i) = N_2^{'}(CH, C_i)/N_1(u, C_i)$

Fin Tant que

Dans la deuxième phase, chaque capteur actif classe ses 1-voisins par ordre croissant de leurs poids. Ensuite, il vérifie le nombre de capteurs actifs dans sa zone de détection. Soit t le nombre de capteurs dans la zone de détection d'un capteur u. Si t est inférieur au degré de couverture k, alors u met en mode actif $(k - t)$ capteurs ayant le plus grand poids dans son 1-voisinage. Puis,

u met en mode idle $k/2$ capteurs non actifs qui ont le plus grand poids dans son 1-voisinage. Ces capteurs peuvent remplacer les capteurs actifs qui cessent de fonctionner avant l'écoulement de la période T_{actif} de leurs activités.

Pour illustrer le fonctionnement de CSA, nous l'exécutons sur l'exemple de la figure 6.4. Pour simplifier, nous supposons que les capteurs ont la même quantité d'énergie. De ce fait, le poids des nœuds peut être calculé seulement en fonction de la 2-densité (tableau 6.1). Pour la formation de clusters par CSOS, nous avons utilisé les paramètres suivants : $Thresh_{Upper} = 10$ et $Thresh_{Lower} = 5$ (figure 6.5).

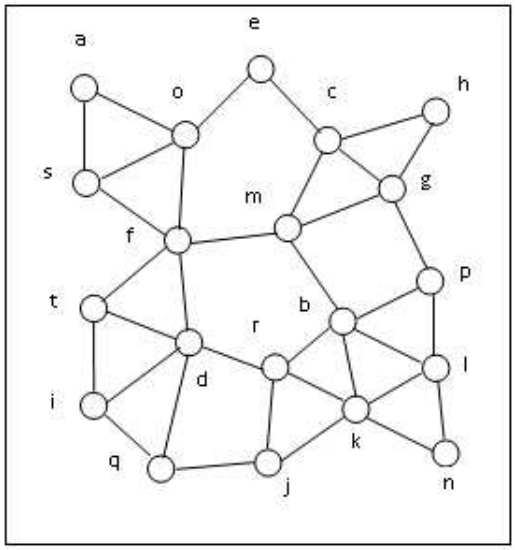
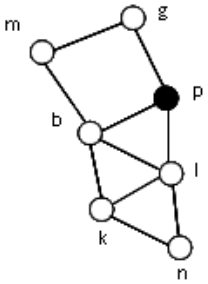


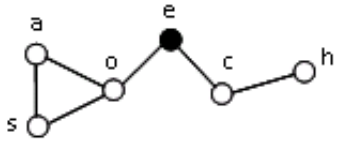
FIG. 6.4 – Exemple d'un réseau sans fil

TAB. 6.1 – Informations topologiques des nœuds du réseau

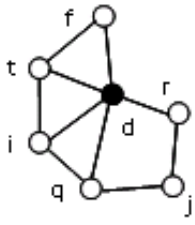
Nœud	2-degré	2-densité
a	5	1.400
b	11	1.727
c	9	1.444
d	11	1.727
e	8	1.625
f	13	1.615
g	8	1.500
h	5	1.400
i	6	1.667
j	8	1.750
k	9	1.667
l	8	1.750
m	14	1.714
n	6	1.833
o	8	1.500
p	9	1.778
q	7	1.714
r	12	1.833
s	7	1.429
t	8	1.500



(a) cluster #1



(b) cluster #2



(c) cluster #3

FIG. 6.5 – Formation de clusters

Après la formation de clusters par CSOS. CSA sélectionne les capteurs actifs qui assurent la 1-couverture de zone comme montre la figure 6.6.

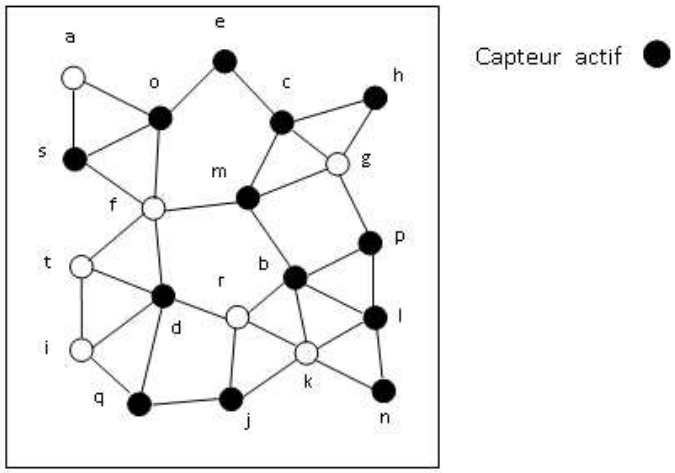


FIG. 6.6 – Sélection de capteurs actifs

6.5 Evaluation de la k-couverture de zone

Au premier regard, le problème de k-couverture semble être très difficile. Une première solution consiste à trouver toutes les régions partagées par un certain nombre de capteurs et vérifier que chaque région est couverte par au moins *k* capteurs. La vérification géométrique de toutes les régions couvertes par un nombre de capteurs est un travail difficile et complexe parce qu'il pourrait exister autant de régions partagées par plusieurs capteurs dont le nombre peut atteindre dans le pire des cas $\theta(n^2)$. En outre, il pourrait être difficile de déterminer ces régions.

Dans cette section, nous proposons deux solutions au problème de k-couverture. La première consiste à vérifier que le périmètre de chaque capteur est k-couvert et la deuxième est basée sur l'approche du capteur virtuel.

6.5.1 Couverture du périmètre de la zone de détection d'un capteur

Cette solution consiste à vérifier comment le périmètre de la zone de détection de chaque capteur est couvert au lieu de déterminer la couverture de chaque région de la zone d'intérêt.

Spécifiquement, elle essaie de déterminer si le périmètre de la zone de détection d'un capteur donné est suffisamment couvert et collecte les informations de couverture de tous les capteurs avant la prise de décision.

Avant de présenter cette solution, nous présentons les hypothèses suivantes et le lemme suivant sur lesquels se base l'algorithme proposé :

Hypothèse 1. Considérons deux capteurs quelconques c_i et c_j. Un point sur le périmètre du capteur c_i est couvert par le capteur c_j si ce point est dans la zone de détection de c_j.

Hypothèse 2. Considérons un capteur quelconque c_i. Nous disons que le périmètre de la zone de détection de c_i est k-couvert si tout point sur ce périmètre est couvert par au moins *k* capteurs en plus de c_i. De même, un segment $[0, 2\pi]$ du périmètre de c_i est k-couvert si tout point sur le segment est couvert par au moins *k* capteurs en plus de c_i.

Hypothèse 3. Considérons deux capteurs c_i et c_j dont les coordonnées sont respectivement (x_i, y_j) et (x_j, y_j) et dénotons par $d(c_i, c_j)$ (Eq 6.2) la distance euclidienne entre c_i et c_j. Si $d(c_i, c_j) > 2 * R_s$ où R_s est le rayon de détection d'un capteur (R_s est le même pour tous les capteurs) alors c_j ne couvre aucun point du périmètre de la zone de détection de c_i.

$$d(c_i, c_j) = \sqrt{|x_i - x_j|^2 + |y_i - y_j|^2} \tag{6.2}$$

Hypothèse 4. Le générateur de positions des capteurs que nous avons développé, ne génère pas la même position pour les capteurs du réseau.

Lemme. La zone d'intérêt *S* est entièrement k-couverte si et seulement si tout capteur dans le réseau a le périmètre de sa zone de détection qui est k-couvert.

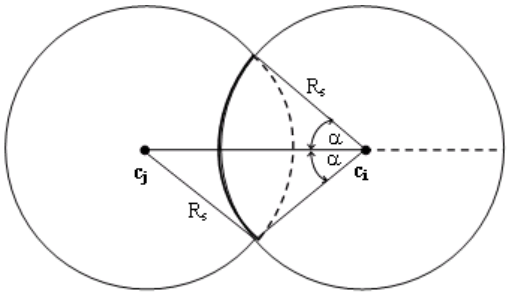


FIG. 6.7 – Périmètre couvert du capteur c_i par c_j

Comment se fait le calcul du périmètre couvert d'un capteur c_i par un capteur c_j ?

Supposons que le capteur c_j se trouve à gauche du capteur c_i comme c'est présenté dans la figure 6.7, le calcul du périmètre de la zone de détection de c_i couvert par c_j se fait comme suit :

- Calcul de l'angle de couverture $\alpha = cos^{-1}(\frac{d(c_i,c_j}{2R_s})$ ou $\alpha = Arccos(\frac{d(c_i,c_j}{2R_s})$
- Calcul de l'arc couvert par c_j : l'arc est compris entre l'angle $\pi-\alpha$ et $\pi+\alpha$ $([\pi-\alpha, \pi+\alpha])$.

Pour déterminer le périmètre couvert de c_i par les autres capteurs, nous procédons ainsi :

1. Pour tout capteur c_j tel que $d(c_i, c_j) \leq 2 * R_s$,
   - Calculer la distance euclidienne $d(c_i, c_j)$ séparant $c_i(x_i, y_i)$ et $c_j(x_j, y_j)$
   - Déterminer le demi angle α_j qui couvert un demi arc du paramètre couvert par c_j :
     $$\alpha_j = cos^{-1}(\frac{d(c_i,c_j)}{2R_s})$$
   - Calculer le périmètre couvert par c_j représenté entre les angles $\alpha_{j,g}$ et $\alpha_{j,d}$.
2. Pour tout voisins c_j de c_i tels que $d(c_i, c_j) \leq 2 * R_s$, placer l'arc couvert par c_j (représenté entre les angles $\alpha_{j,d}$ et $\alpha_{j,g}$) sur un segment $[0, 2\pi]$.
3. Si le segment $[0, 2\pi]$ est entièrement couvert, on dira que le périmètre de la zone de détection de c_i est couvert, sinon il existe certains points dans la zone de détection de ci qui ne sont pas couverts par d'autres capteurs, appelés points vides.

La complexité dans le pire des cas pour vérifier si le périmètre d'un capteur est k-couvert, est de l'ordre de $\theta(d\,log\,d)$ où d est le nombre de capteurs dont les zones de détection se chevauchent avec le capteur considéré. Donc, la complexité de l'algorithme proposé pour vérifier que les périmètres de tous les capteurs sont k-couverts, est $\theta(n * d\,log\,d)$, où d est le nombre maximum de capteurs qui peuvent partager une même zone de détection avec un autre capteur et n le nombre de capteurs déployés dans la zone d'intérêt *S*.

6.5.2 Capteur virtuel et discrétisation de la zone de déploiement

Notre solution consiste à discrétiser la zone d'intérêt S en plusieurs points en forme de matrice. Pour vérifier la k-couverture de chaque point dans la zone d'intérêt, nous supposons que chaque point est un capteur virtuel qui échange des beacons ou messages Hello avec son

voisinage pour connaître son degré et par conséquent le nombre de capteurs qui le couvrent. Ce nombre représente le degré de couverture du point c'est-à-dire sa k-couverture.

L'algorithme associé à cette approche s'exécute comme suit :

1. Discrétiser la zone d'intérêt S en forme de matrice, et donner un nombre aléatoire m représentant le nombre de capteurs virtuels pour vérifier le degré de couverture de la zone d'intérêt.
2. Générer aléatoirement un capteur virtuel : par génération de positions aléatoires suivant une loi de distribution uniforme telle que les capteurs virtuels générés sont uniformément distribués dans la zone d'intérêt S et leurs positions ne coïncident pas avec les capteurs qui existent déjà ou les capteurs virtuels déjà générés.
3. Calculer le degré de couverture de chaque capteur virtuel généré : pour cela, il suffit de calculer la distance euclidienne qui sépare ce capteur à tout capteur déployé dans la zone d'intérêt. Si cette distance est inférieure au rayon de détection R_s, incrémenter son degré de 1 en choisissant un 1-voisin en mode sommeil qui a le plus grand poids.
4. Calculer le degré de couverture de la zone d'intérêt : le degré virtuel de la zone est égal au degré minimal de l'ensemble des capteurs virtuels générés.

Formellement, soient $v_i,\ i = 1,\ ..., m$ les capteurs virtuels générés et $\delta_1(v_i)$ leurs degrés correspondants. La k-couverture de la zone d'intérêt S, notée $C_i(S)$ est égale au degré minimal de tous les capteurs virtuels. Si $C_i(S)$ est nulle $(C_i(S) = 0)$, alors il existe des points vides (points non couverts par les capteurs actifs) dans la zone d'intérêt et dans ce cas le taux de couverture de la zone est inférieur à 100%.

1. Pour tout $v_i(x_i, y_i)\ i = 1,\ ...,\ m$: capteurs virtuels, appelés points de références,
   Calculer $\delta_1(v_i) = |N_1(v_i)|$ avec $N_1(v_i)$ est l'ensemble de 1-voisins actifs de v_i.
2. Calculer la k-couverture $C_v(S)$ de la zone d'intérêt S,
   $C_v(S) = Min\{\delta_1(v_i)\ \ i = 1, ..., m\ \}$
3. Si $C_v(S) = 0$ alors il existe des points vides dans la zone S (points non couverts), sinon l'ordre de couverture de la zone S est $C_v(S)$.

En particulier, si nous voulons vérifier la 1-couverture de la zone d'intérêt, il suffit que chaque capteur virtuel généré ait au moins un voisin en état actif.

6.6 Calcul du taux de couverture

Pour calculer le taux de couverture $\theta(C_v(S))$ d'une zone S, nous générons aléatoirement un ensemble de points de références dans la zone S, soit $Ref(v_i)$ cet ensemble. Ces points de références représentent des capteurs virtuels. Puis, nous déterminons le nombre de capteurs virtuels ayant au moins un voisin en état actif $\{ v_i : \delta_1(v_i) > 0 \}$. De ce fait, le taux de couverture de la zone S est le rapport du nombre de capteurs virtuels ayant au moins un voisin en état actif et $Ref(v_i)$:

$$\theta(C_v(S)) = \frac{| \{ v_i : i = 1, ..., m / \delta_1(v_i) > 0 \} |}{| Ref(v_i) |} \tag{6.3}$$

La précision du taux de couverture $\theta(C_v(S))$ dépend du nombre de points de référence $Ref(v_i)$ et de leurs positions.

Cet algorithme nous permet d'améliorer les performances de CSA et de pallier ses limites. De ce fait, si nous remarquons qu'un capteur virtuel n'a aucun voisin en état actif, nous exécutons la partie maintenance de l'algorithme d'ordonnancement d'activité des capteurs CSA_VS pour améliorer le taux de couverture de la zone d'intérêt. Pour cela, nous choisissons le capteur qui a le plus grand poids parmi les voisins en mode idle du capteur virtuel et nous le mettons en mode actif. Nous pouvons de même généraliser ce procédé pour assurer la couverture multiple de la zone d'intérêt à un certain ordre *k*.

6.7 Evaluation des performances

Dans cette section, nous évaluons le taux de couverture de zone par CSA et CSA_VS pour deux degrés de couverture : 1-couverture et 2-couverture. Puis, nous comparons les performances de CSA_VS à celles des algorithmes LPA, PKA, et CKA présentés dans [113] en termes de nombre et de pourcentage de nœuds actifs.

Pour l'évaluation de la couverture de zone (taux de couverture et pourcentage de nœuds actifs), nous avons utilisé un simulateur écrit en C++, que nous avons développé pour éviter le bruit causé par les autres simulateurs.

6.7.1 Contexte d'évaluation de la simulation

Pour illustrer l'impact de la densité sur le nombre et le pourcentage de capteurs actifs, nous avons considéré plusieurs topologies de réseau. Chacune d'elle comprend n capteurs ayant la même portée de communication R_c et la même portée de détection R_s telle que $R_c = 2 * R_s$. Les capteurs sont placés aléatoirement dans une zone d'intérêt de superficie $100m \times 100m$ suivant une loi de distribution uniforme, et la station de base est placée en dehors de la zone d'intérêt. En outre, pour voir l'effet de la densité du lien sur les performances de CSA_VS, nous avons utilisé deux portées de détection distinctes $20m$ et $40m$. Pour chaque configuration, les simulations sont répétées 100 fois pour le calcul de la valeur moyenne de chaque critère de performance. Le tableau 6.2 résume les paramètres de simulation utilisés.

TAB. 6.2 – Paramètres de simulation

Paramètres	Valeurs
Superficie de la zone d'intérêt	$100m \times 100m$
Nombre de capteurs	100, 200, 300, 400, et 500
Rayon de détection (R_s)	20m, 40m

6.7.2 Analyse des résultats numériques

Dans cette section, nous présentons et nous analysons les performances de l'algorithme CSA et sa version améliorée CSA_VS.

Taux de couverture en fonction du nombre de capteurs déployés

Pour la détermination du taux de couverture de la zone d'intérêt par CSA et CSA_VS, nous avons généré aléatoirement 100 points de référence et nous avons vérifié s'ils sont couverts ou non. De ce fait, nous avons utilisé plusieurs topologies de réseau dont les densités sont distinctes pour illustrer l'impact de la densité sur le taux de couverture de la zone. En outre, nous avons utilisé deux portées de détection distinctes.

La figure 6.8 montre un exemple de déploiement aléatoire de 500 capteurs sur une zone d'intérêt de superficie $100m \times 100m$.

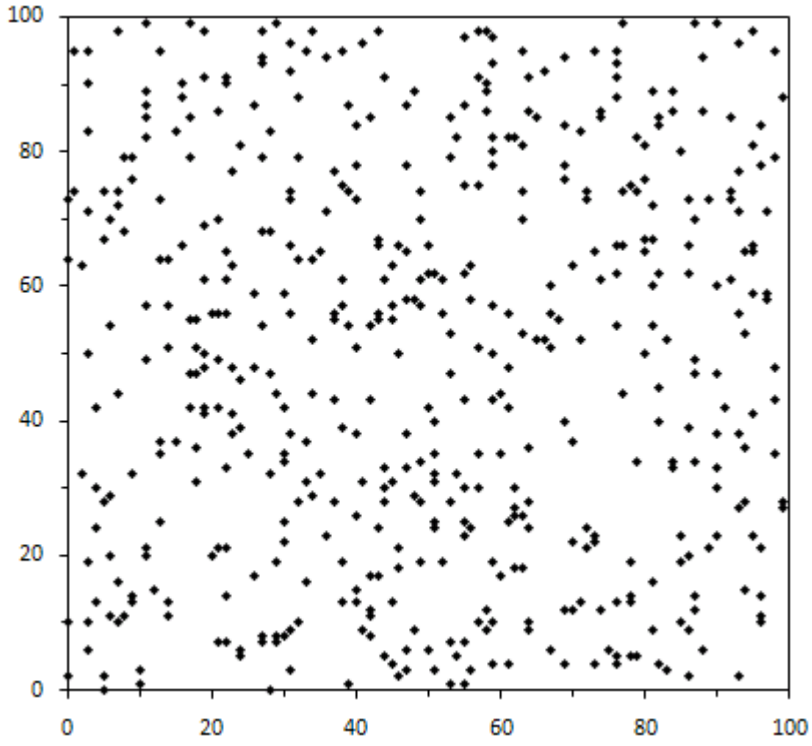
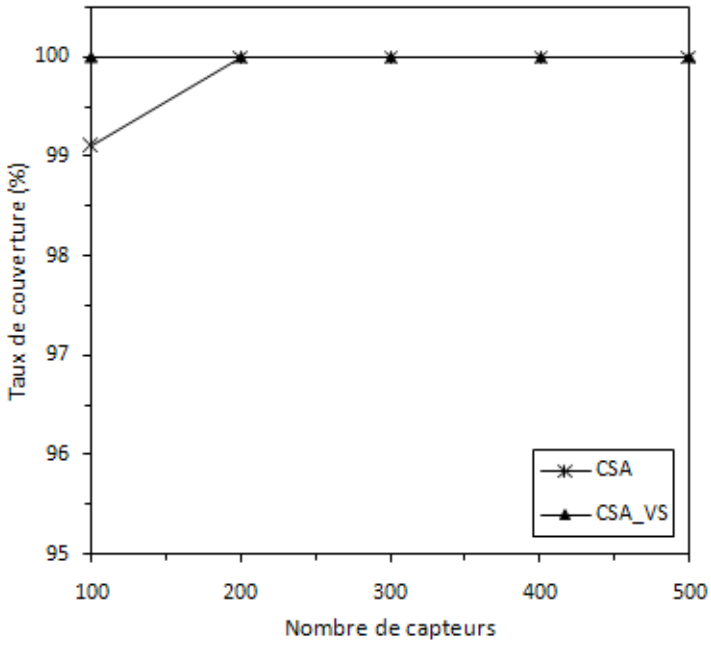
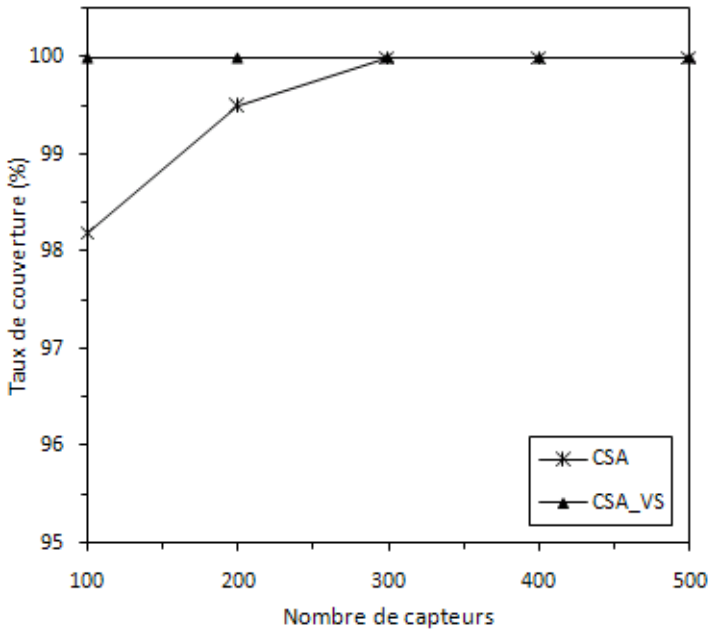


FIG. 6.8 – Exemple de déploiement aléatoire de 500 capteurs



(a) $(R_S = 20, k = 1)$



(b) $(R_S = 20, k = 2)$

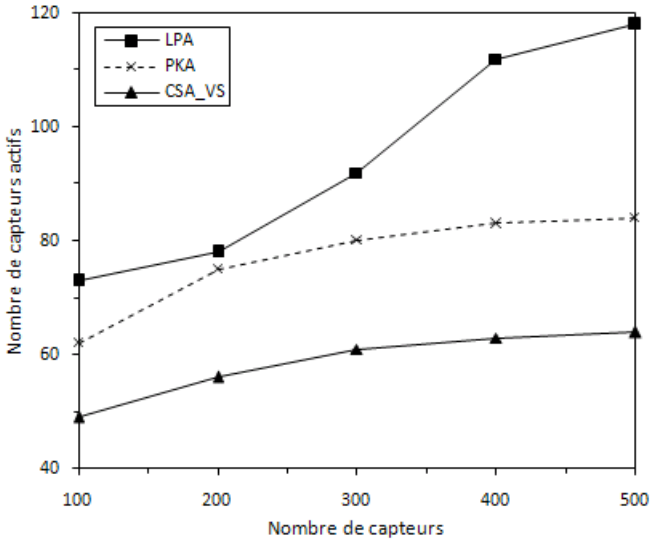
FIG. 6.9 – Taux de couverture

La figure 6.9(a) montre que le taux de la 1-couverture assuré par CSA est de l'ordre de $99,1\%$ quand le nombre de capteurs est 100 dans le cas où le rayon de détection est $20m$ et il passe à 100% (couverture totale de la zone d'intérêt) quand le nombre de capteurs est égal à 200. Cependant, le taux de la 2-couverture assuré par CSA est de l'ordre de $98,2\%$ quand le nombre de capteurs est 100 et passe à 100% quand le nombre capteurs dépasse 300. En outre, nous avons constaté que le taux de la 1-couverture et de la 2-couverture assurés par CSA_VS est toujours 100% pour les deux portées de surveillance $R_s = 20m$ et $R_s = 40m$.

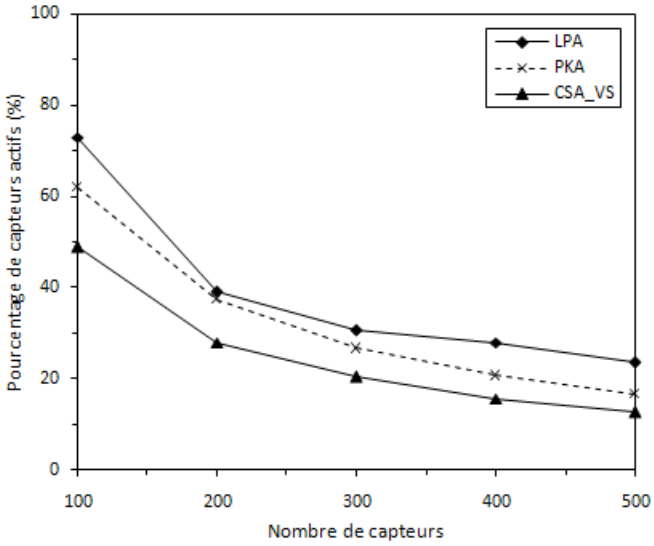
Evaluation de la couverture de zone

Pour l'évaluation du nombre de capteurs actifs en fonction de la portée de détection et du degré de couverture k, nous avons de même utilisé deux portées de détection distinctes $R_s = 20m$ et $R_s = 40m$ pour voir l'effet de la densité du lien sur les performances de CSA_VS et les degrés de couverture suivants : $k = 2,\ 3,$ et 4.

- Evaluation de la 2-couverture pour $R_s = 20\,m$



(a) Nombre de capteurs actifs



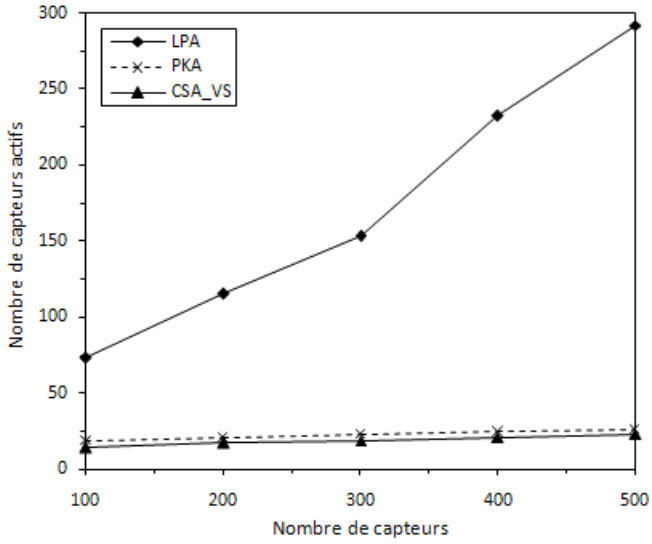
(b) Pourcentage de capteurs actifs

FIG. 6.10 – Evaluation de la 2-couverture pour $R_s = 20\,m$

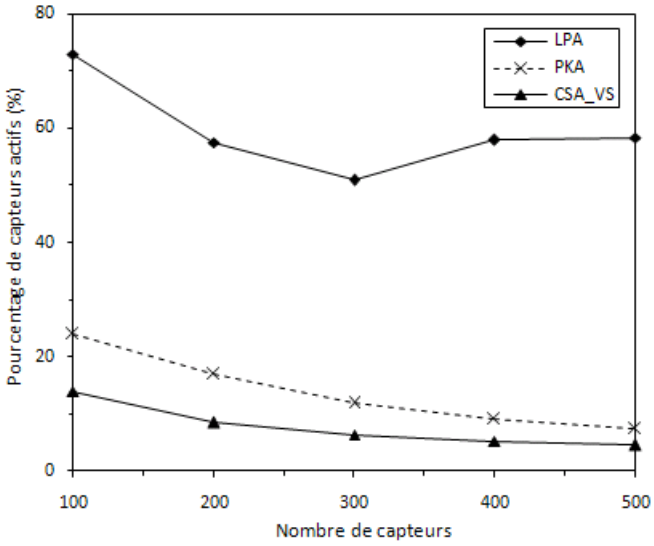
Les figures 6.10(a) et 6.10(b) comparent respectivement le nombre et le pourcentage de capteurs actifs pour assurer la 2-couverture totale de la zone d'intérêt quand la portée de détection est 20 m. Nous constatons que CSA_VS implique un nombre minimal de capteurs pour assurer la 2-couverture totale de la zone d'intérêt relativement à PKA et LPA. En outre, CSA_VS fournit de meilleurs résultats en terme de pourcentage de nœuds actifs. En effet, quand le nombre de

capteurs déployés augmente, le pourcentage de nœuds actifs diminue grandement. Par contre, LPA implique plus que 50% de capteurs pour la 2-couverture de la zone d'intérêt.

- Evaluation de la 2-couverture pour $R_s = 40\,m$



(a) Nombre de capteurs actifs



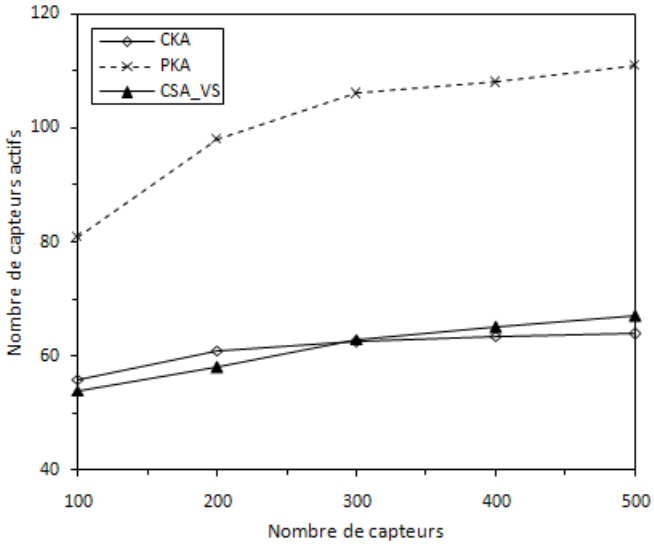
(b) Pourcentage de capteurs actifs

FIG. 6.11 – Evaluation de la 2-couverture pour $R_s = 40\,m$

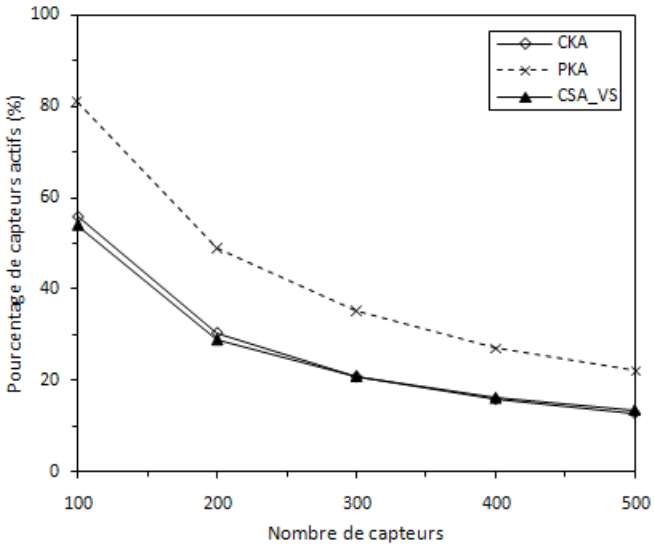
En outre, les figures 6.10(a), 6.10(b), 6.11(a) et 6.11(b) montrent l'effet de la densité de lien sur le nombre de capteurs actifs. Ainsi, quand la portée de détection des capteurs augmente le nombre de capteurs actifs diminue à l'exception de LPA qui est centralisé. Ces résultats montrent

également que l'aspect quasi local de CSA_VS a de bons effets sur ses performances. D'autre part, la forte densité des réseaux a un impact négatif sur les performances de LPA car quand la densité augmente le degré maximal d'un nœud augmente ce qui aura une influence sur le rapport $1/(1+\Delta)$ où Δ est le degré maximal et par conséquent le cardinal de l'ensemble C contenant les nœuds actifs augmente.

Evaluation de la 3-couverture pour $R_s = 20\,m$



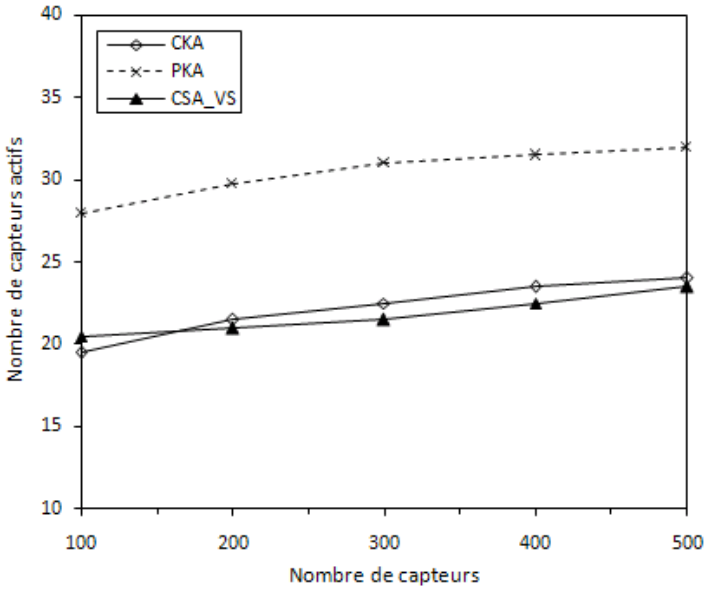
(a) Nombre de capteurs actifs



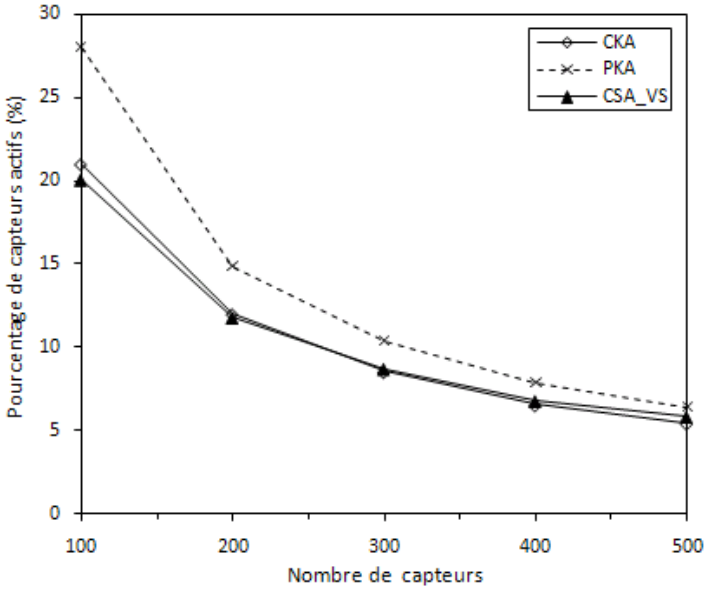
(b) Pourcentage de capteurs actifs

FIG. 6.12 – Evaluation de la 3-couverture pour $R_s = 20\,m$

Evaluation de la 3-couverture pour $R_s = 40\,m$



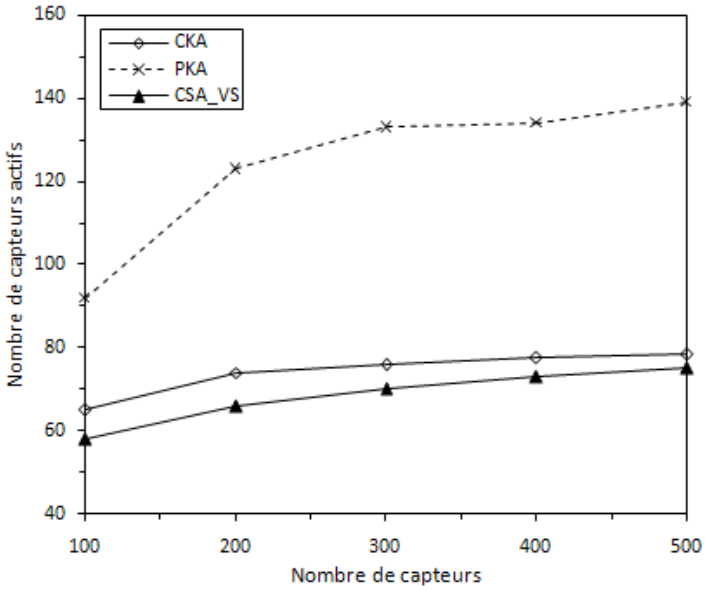
(a) Nombre de capteurs actifs



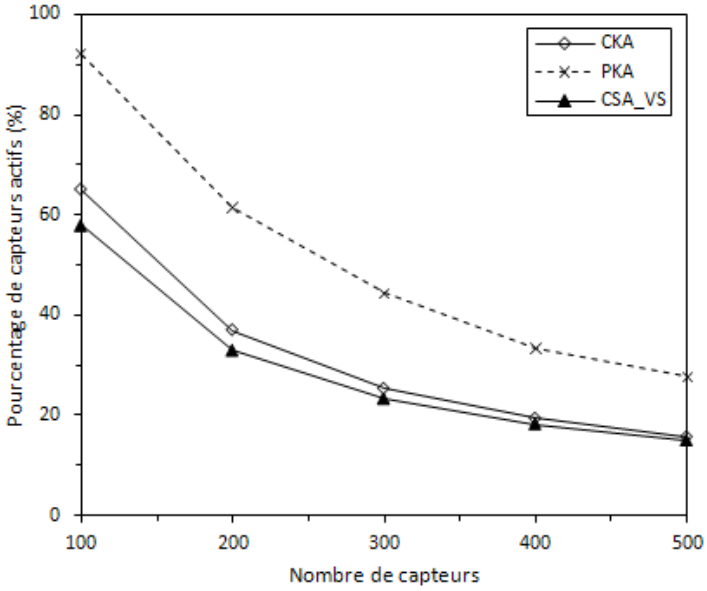
(b) Pourcentage de capteurs actifs

FIG. 6.13 – Evaluation de la 3-couverture pour $R_s = 40\,m$

Evaluation de la 4-couverture pour $R_s = 20\,m$



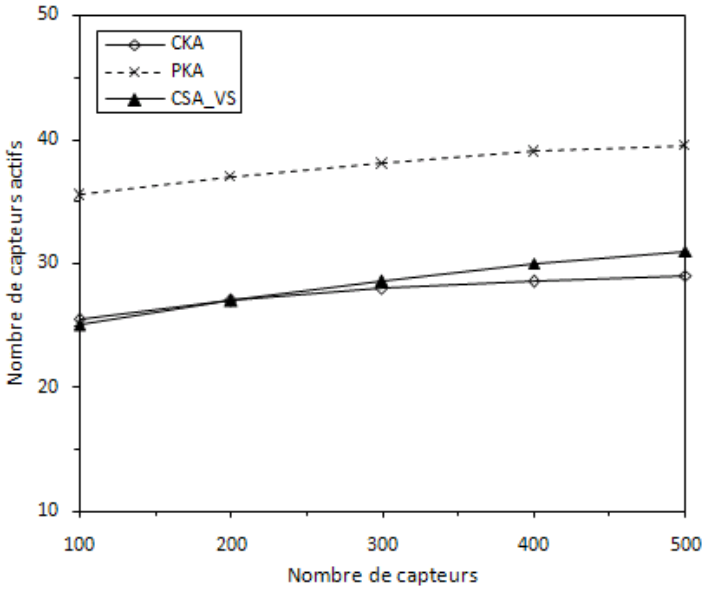
(a) Nombre de capteurs actifs



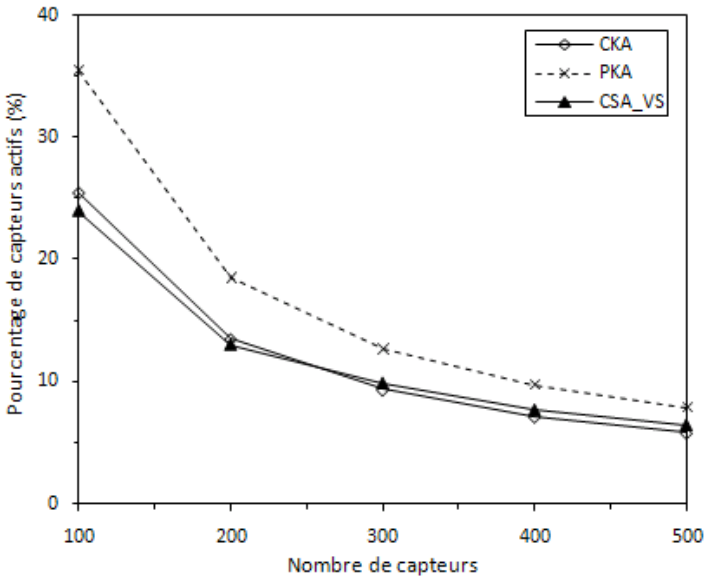
(b) Pourcentage de capteurs actifs

FIG. 6.14 – Evaluation de la 4-couverture pour $R_s = 20\,m$

Evaluation de la 4-couverture pour $R_s = 40\,m$



(a) Nombre de capteurs actifs



(b) Pourcentage de capteurs actifs

FIG. 6.15 – Evaluation de la 4-couverture pour $R_s = 40\,m$

Dans les figures 6.12, 6.13, 6.14 et 6.15, nous avons comparé CSA_VS à PKA et à CKA qui a le même aspect que CSA_VS. Les résultats montrent que CSA_VS fournit de meilleurs résultats que CKA pour un degré de couverture assez élevé en terme de nombre de nœuds actifs

quand le nombre de nœuds dans le réseau est inférieur à 300 alors que CKA dépasse légèrement CSA_SV quand le réseau devient dense. Cependant, les auteurs dans [113] n'ont pas vérifié que le taux de couverture d'un degré assez élevé ($k = 3$ ou $k = 4$) est 100%, contrairement à CSA_VS qui fournit la couverture totale de la zone d'intérêt avec les résultats présentés dans ces différentes figures.

Durée de vie du réseau de capteurs

Pour évaluer les performances de CSA_VS en termes de durée de vie, nous avons utilisé le contexte d'exécution de la simulation pour l'évaluation de PEAS [118]. Dans ce contexte, la consommation d'énergie se base sur les caractéristiques des capteurs de Berkeley où les consommations d'énergie d'un capteur dans une transmission, réception, idle et sommeil sont respectivement 60, 12, 12, et $0.03mW$, avec l'énergie initiale pour chaque capteur de 60 Joules. Cette quantité d'énergie lui permet de rester 5000 secondes en modes réception/idle. Nous avons considéré plusieurs topologies pour illustrer l'impact de la densité des réseaux sur leurs durées de vie et nous avons fait un ajustement de quelques paramètres du contexte d'exécution de PEAS pour la clarté de la simulation. De ce fait, nous avons considéré une zone d'intérêt d'une superficie $100m \times 100m$ au lieu de $50m \times 50m$ et la portée de détection de chaque capteur est 20m au lieu de 10m. Le tableau 6.3 présente les paramètres de simulation.

TAB. 6.3 – Paramètres de simulation

Paramètres	Valeurs
Superficie de la zone d'intérêt	$100m \times 100m$
Nombre de capteurs	150, 200, 300, 400, 500
Rayon de détection (R_s)	20m
Communication	12mW
Réception	6mW
Idle	6mW
Sommeil	0.03mW
Taille de paquet échangé	25 octets

Dans chaque topologie, les capteurs sont aléatoirement déployés sur une zone d'intérêt selon une loi de distribution uniforme et la station de base est placée en dehors de la zone d'intérêt. Au début de la simulation, tous les capteurs ont la même quantité d'énergie (60 Joules) et ils cessent

de fonctionner quand leurs batteries sont épuisées. La durée de vie du réseau est en fonction du taux de couverture de la zone des cibles. Ainsi, les simulations s'exécutent jusqu'à ce que le taux de couverture devienne inférieur à un certain seuil. Si ce taux devient inférieur à ce seuil, on dit que le réseau cesse d'accomplir ses missions. Nous avons fixé ce seuil à 90%. Les valeurs moyennes des paramètres de performances sont calculées après chaque période.

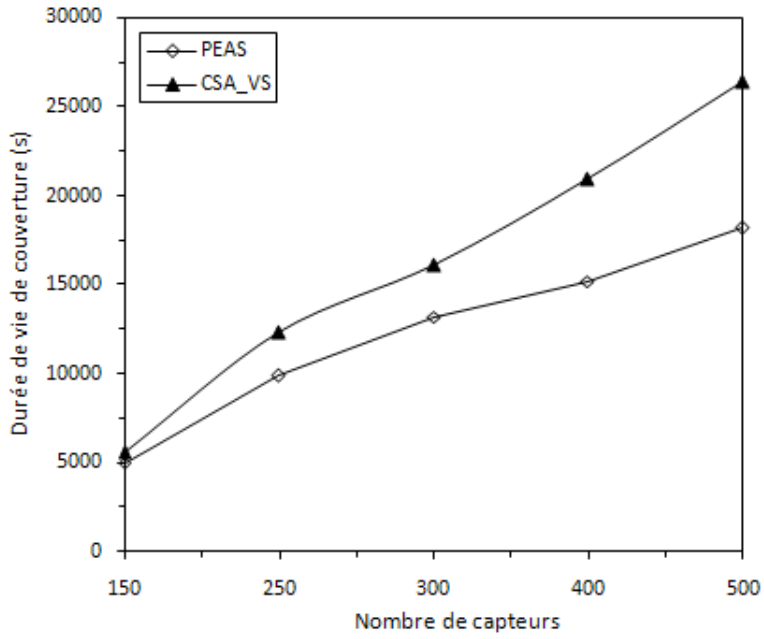


FIG. 6.16 – Comparaison de durée de vie

La figure 6.16 montre la relation entre la densité des nœuds et la durée de vie. Nous constatons sans surprise que la durée de vie augmente quand la densité des réseaux augmente. Cela permet aux capteurs de rester en mode sommeil plus de temps dans un réseau dense que dans un autre moins dense. En plus, l'aspect quasi local de CSA_VS et la sélection périodique des capteurs actifs en fonction de leurs capacités lui permettent de prolonger plus la durée de vie que PEAS.

6.8 Conclusion

Dans ce chapitre, nous avons abordé le problème de la k-couverture de zone car dans certaines applications, il est possible que certaines régions dites régions "sensibles" dans la zone

d'intérêt soient plus importantes que les autres et aient besoin d'être couvertes par plus de capteurs pour garantir la tolérance aux fautes et faire face aux mesures erronées collectées par les capteurs. Par exemple, dans le parc d'une maison de retraite ou d'une clinique, nous assurons la k-couverture des endroits sensibles et dangereux pour protéger les personnes dépendantes. Les solutions que nous avons proposées peuvent tester si un endroit est k-couvert ou non. Ainsi, un endroit sensible est k-couvert si le périmètre de la zone de détection de tout capteur placé dans cet endroit est k-couvert. Pour vérifier la k-couverture de cet endroit, il suffit d'appliquer l'algorithme "couverture de périmètre" ou "capteur virtuel" au niveau de cet endroit et vérifier que tout point du segment $[0, 2\pi]$ est au moins *k* fois généré par les capteurs de cet endroit ou que le capteur virtuel a au moins *k* capteurs actifs.

Le travail présenté dans ce chapitre a contribué à assurer la k-couverture totale de zone d'intérêt en impliquant un nombre minimal de capteurs. De ce fait, la consommation d'énergie sera minimisée et par conséquent la durée de vie des réseaux sera prolongée. Afin de montrer les avantages des algorithmes que nous avons présentés, nous avons comparé les résultats obtenus par ces algorithmes aux résultats fournis par d'autres algorithmes performants décrits dans la littérature. Nous montrons que CSA_VS implique un nombre inférieur de capteurs actifs pour la couverture de zone par rapport à LPA, PKA, et CKA et prolonge la durée de vie plus que PEAS. Ainsi, l'aspect quasi local de CSA_VS et la sélection périodique des capteurs actifs en fonction de la k-densité et de l'énergie restante des capteurs ont permis à CSA_VS de fournir de bons résultats en termes de pourcentage de nœuds actifs et de durée de vie.

Conclusion

Dans cette thèse, nous avons traité le problème de diffusion et de couverture de zone dans les réseaux de capteurs. Pour atteindre cet objectif, nous avons proposé trois protocoles. Le premier est un protocole de diffusion dans un environnement non idéaliste (modèle lognormal). Le deuxième concerne l'auto-organisation pour un acheminement moins coûteux de l'information d'un capteur à la station de base. Le troisième est un protocole d'ordonnancement d'activité de capteurs pour une k-couverture totale de la zone d'intérêt.

Dans ce travail, nous avons présenté d'abord un état de l'art sur les approches de diffusion de l'information dans les réseaux ad hoc et de capteurs. Dans cette étude, nous avons présenté et analysé les protocoles basés sur deux approches : le clustering et l'ensemble dominant connecté (CDS). Ensuite, nous avons proposé notre contribution pour améliorer les performances du protocole de diffusion MPR dans un environnement réaliste. Ainsi, pour pallier aux limites de ces protocoles, nous avons développé un nouvel algorithme de clustering CSOS pour une diffusion efficace dans les réseaux de capteurs sans fil. CSOS prend en considération les contraintes imposées par les réseaux de capteurs et il s'adapte à un type particulier de réseaux : les réseaux de capteurs mobiles. L'objectif de CSOS est de créer une topologie virtuelle stable. Cette topologie consiste à générer des 2-clusters stables dont la taille est homogène et d'éviter les réactions en chaine qui peuvent être déclenchées lorsqu'un nœud migre d'un cluster à un autre ou lorsqu'un cluster perd son cluster-head. En outre, CSOS s'exécute périodiquement pour ne pas épuiser les cluster-heads et implique une combinaison de métriques de stabilité pour élire les cluster-heads : k-densité, énergie restante et mobilité. Les résultats de simulations montrent l'apport de la stabilité sur les performances de CSOS. En effet, ces résultats ont montré que CSOS est plus robuste face à la mobilité des nœuds que d'autres algorithmes de clustering existants.

En outre, pour assurer la k-couverture d'une zone d'intérêt, nous avons proposé un troisième protocole d'ordonnancement d'activité de capteurs CSA_VS. Ce protocole implique un nombre minimal de capteurs actifs pour accomplir cette tâche par rapport à d'autres protocoles de cou-

verture de zone tels que PKA, LPA, CKA et PEAS. De ce fait, CSA_VS permet de conserver l'énergie et par conséquent maximiser la longévité du réseau. Ceci résulte de l'aspect quasi local de CSA_VS et de la sélection périodique des capteurs actifs en fonction de leurs capacités basées sur la k-densité et l'énergie restante. Les résultats de simulations ont bien illustré les avantages de CSA_VS en termes de pourcentage de capteurs actifs et durée de vie par rapport à d'autres protocoles de couverture existants.

Finalement, pour tester les performances de CSA_VS, nous avons développé une plateforme générique permettant la couverture d'un parc d'une maison de retraite ou une clinique. Pour cela, nous avons utilisé des capteurs MicaZ de Crossbow. Le choix de ce type de capteurs s'explique par leur popularité et leur facilité de programmation. Dans cette plateforme, les capteurs sont déployés dans les endroits sensibles du parc et ils remontent l'information au centre de contrôle lorsqu'un événement critique se produit.

Perspectives du travail de thèse

Les réseaux de capteurs constituent un axe de recherche très fertile et ont de nombreuses perspectives d'application dans des domaines très variés : domotique, surveillance industrielle et environnementale, etc. Il reste encore de nombreux problèmes à résoudre dans ce domaine afin de pouvoir les utiliser dans les conditions réelles. En outre, chaque application a ses propres contraintes. De ce fait, la conception d'un réseau de capteurs est une tâche très difficile parce qu'elle devra combiner les contraintes propres aux systèmes distribués et aux systèmes embarqués.

Dans cette thèse, nous avons utilisé un modèle de couche physique idéale (modèle de disque unitaire) pour le développement de CSOS et CSA_VS. Dans ce modèle, les communications sont considérées comme toujours fiables : pas d'interférences et pas de collisions, et qu'un nœud destinataire v reçoit toujours un message sans erreur d'un nœud émetteur u quand il se trouve à sa portée $(d(u, v) \leq R_c)$. Cependant, les communications utilisant les liens radio ne sont pas toujours fiables et les messages échangés entre les nœuds ne sont pas toujours reçus sans erreur à cause de plusieurs facteurs qui perturbent ces communications tels que les obstacles. Dans ce contexte, les solutions que nous avons présentées, peuvent être toujours améliorées afin d'être adaptées à un environnement non idéal.

En outre, parmi les résultats de recherche que nous avons présentés dans cette thèse, la plupart d'entre eux sont obtenus par des simulations. De ce fait, un des travaux futurs est d'implémenter les fonctionnalités qui manquent dans le kit utilisé pour le développement de l'application générique afin d'enrichir et améliorer les résultats d'expérimentation obtenus. Ensuite, il faudra mettre en place une application qui implique plusieurs types de capteurs pour assurer une surveillance permanente et un suivi médical à distance des personnes dépendantes et qui soit capable de remonter une alerte à plus faible latence si un évènement critique se produit.

Annexe A : Déploiement d'un réseau de capteurs pour la surveillance des personnes dépendantes

Dans cette thèse, nous avons abordé différentes problématiques des réseaux de capteurs telles que l'auto-organisation, la diffusion, la conservation d'énergie et la couverture de zone. Comme le contexte de la thèse est d'appliquer un réseau de capteurs pour l'exécution de services dans l'environnement des personnes dépendantes, nous allons analyser le rôle d'un réseau de capteurs dans la surveillance d'un parc d'une maison de retraite ou d'une clinique et présenter une plateforme d'expérimentation d'un réseau de capteurs que nous déployons au sein du laboratoire d'informatique de l'université de Franche-Comté. Nous commençons par la présentation de l'architecture de la surveillance d'une zone d'intérêt. Ensuite, nous analysons le rôle d'un réseau de capteurs dans la surveillance d'une zone d'intérêt. Puis, nous présentons la plateforme d'expérimentation d'un réseau de capteurs MicaZ de Crossbow et les outils logiciels pour remonter les informations à un centre de contrôle.

A.1 Architecture de la plateforme

La plateforme que nous avons développée, consiste à déployer un ensemble de capteurs dans les endroits sensibles d'un parc d'une maison de retraite ou d'une clinique (figure A.1) et sur ses occupants (personnes dépendantes). Ces capteurs dont la tâche est d'effectuer un monitoring permanent du parc et des personnes dépendantes, communiquent entre eux en fonction de l'état du parc et des mouvements des personnes dépendantes, et préviennent tout personnel distant (centre de contrôle) pour qu'il intervienne dans les meilleurs délais lorsqu'un événement pertinent se produit. Dans notre contexte, l'événement pertinent s'agit de la température lorsqu'elle dépasse une certaine valeur seuil (T_α). En outre, pour assurer la couverture globale du parc tout en pré-

voyant la tolérance aux fautes en cas de panne ou d'épuisement de l'énergie de certains capteurs, il est nécessaire de garantir une couverture superposée de toute la surface du parc.

Dans la plateforme proposée, chaque capteur mesure périodiquement la température de son environnement proche et envoie une alerte à la station de base qui est connectée à un ordinateur domestique lorsque la valeur de la température collectée dépasse la valeur seuil T_α. L'ordinateur domestique remonte à son tour l'alarme à un centre de contrôle distant via une passerelle résidentielle (connexion Internet). En outre, la plateforme permet aussi de repérer l'endroit dans lequel l'événement pertinent s'est produit. Par ailleurs, l'application que nous avons mise en place, peut servir d'application générique. En effet, dans la plupart des cas, chaque capteur mesure plusieurs grandeurs physiques.

Dans [64, 65], nous avons proposé une architecture logicielle basée sur Bluetooth [126] pour interconnecter les appareils domestiques dans un habitat. Cette interconnexion peut nous permettre l'accès et l'utilisation des services offerts par ces appareils domestiques. Cependant, la mise en place de cette architecture nécessite l'existence de dispositifs dotés de la technologie Bluetooth.

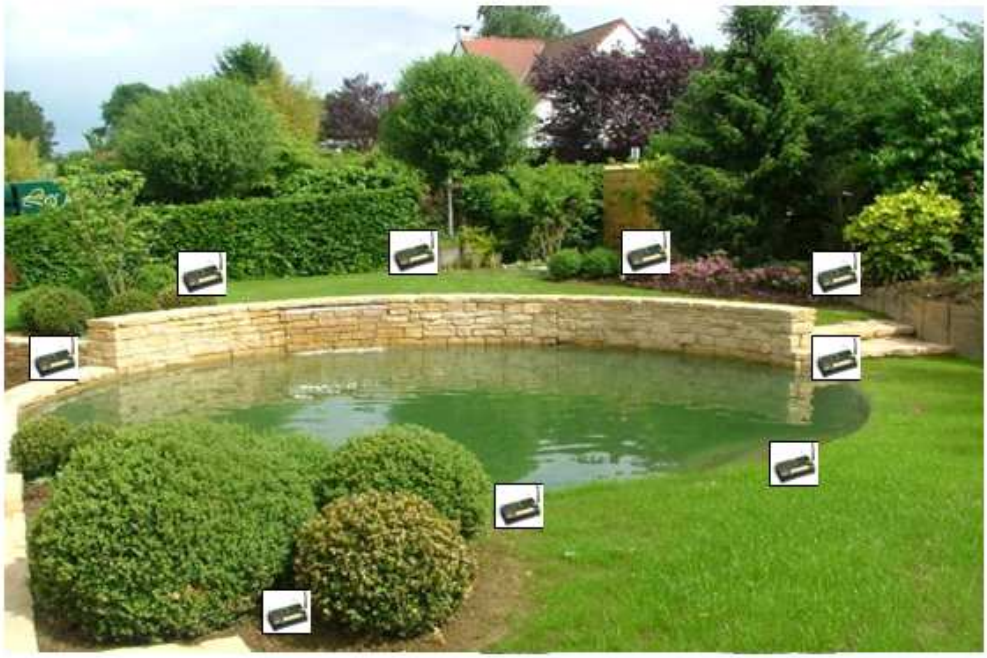


Fig A.1- Zone de déploiement

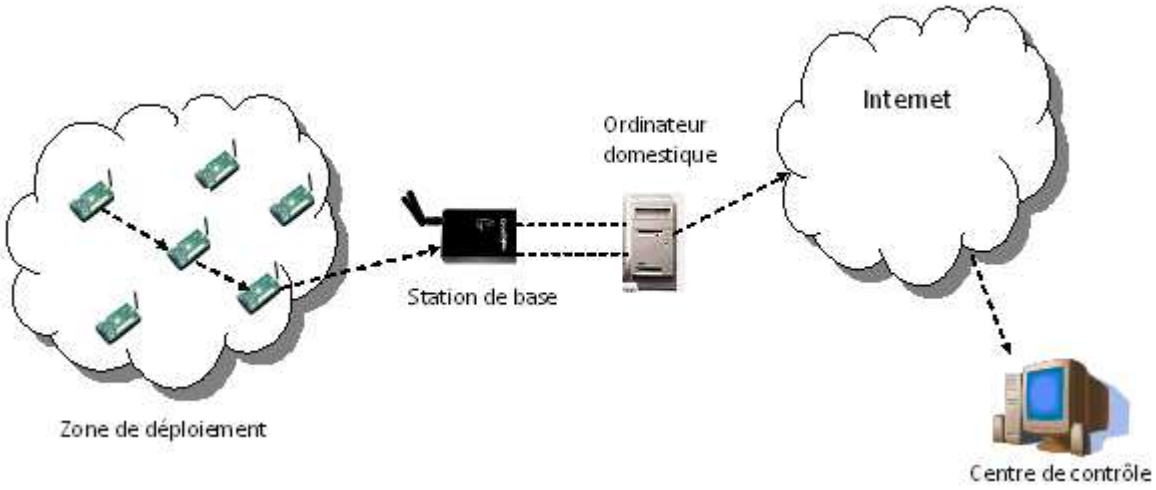


Fig A.2- Architecture de la plateforme

Au début, nous avons assigné à chaque capteur un identifiant $Node_{Id}$. Cet identifiant représente l'endroit où se trouve le capteur. Puis, nous avons placé ces capteurs dans les différents endroits du parc. La figure A.2 résume l'architecture de la plateforme.

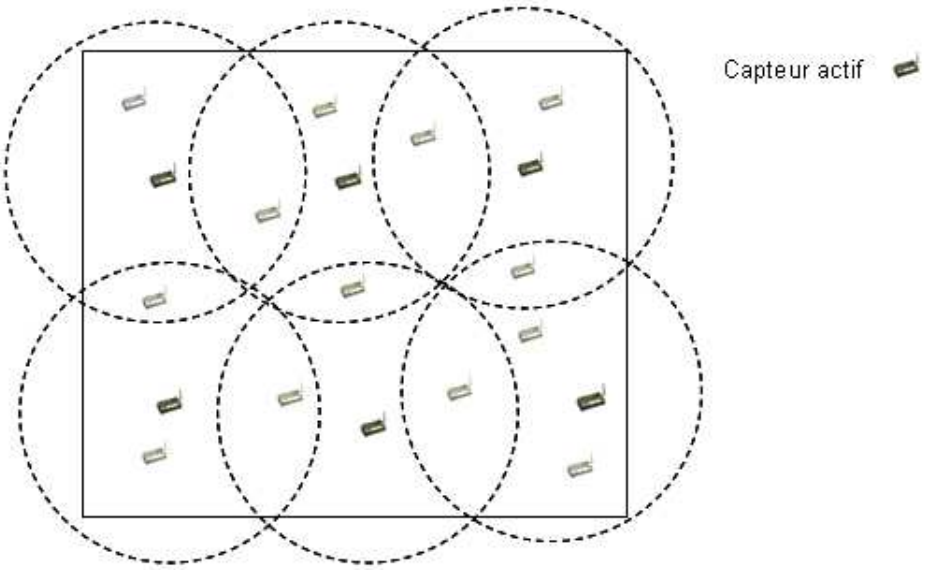


Fig A.3- Couverture de zone par les capteurs

Par ailleurs, la contribution proposée pour concevoir cette plateforme permet de mettre en place de nouveaux mécanismes de collaboration entre matériel et logiciel en utilisant des méthodes de conception conjointe pour la mise en place de la plateforme. Cette plateforme est

orientée monitoring et événement (Event/driven). En effet, elle permet d'une part aux capteurs déployés d'envoyer périodiquement les informations collectées à la station de base, et d'autre part de faire remonter l'alerte au centre de contrôle lorsqu'un événement pertinent survient. Elle permet également la couverture totale de la zone de déploiement comme le montre la figure A.3.

A.2 Outils matériels et logiciels utilisés

Pour la mise en place de la plateforme, nous avons utilisé les outils matériels et logiciels suivants :

A.2.1 Matériel utilisé

Le kit de réseau de capteurs utilisé pour la mise en place de la plateforme proposée, contient : 6 capteurs MicaZ et une station de base MIB520.

Capteur MicaZ

Un capteur MicaZ se compose de deux modules comme le montre la figure A.4 : une carte MPR2600 et une carte MTS400.

La carte MPR 2600 contient un microcontrôleur et un transceiver :

- Le microcontrôleur contient une mémoire flash de 128kB, une mémoire vive de 4kB et un processeur de 4MHz. Il réalise le calcul et le traitement des informations des capteurs.
- Le transceiver est un module de communication permettant des transmissions à fréquence radio de 2.4 GHz allant jusqu'à 250kbps entre les capteurs et des capteurs vers la station de base.

La carte MTS400 (carte des senseurs) capte des mesures environnementales : température, accélération, pression, luminosité et l'humidité, les transforme en valeurs numériques, puis elle les communique à la carte MPR 2600 (figure A.5).

Une nouvelle génération de carte MTS420 est également fournie chez Crossbow avec une fonctionnalité supplémentaire. Elle fonctionne comme une carte MTS400 et elle offre également la position géographique du capteur.

Fig A.4- Capteur MicaZ

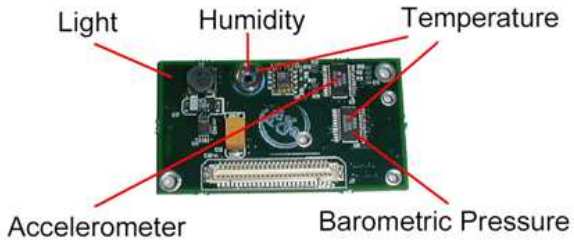


Fig A.5- Carte MTS400

Station de base

La station de base contient deux composants comme montre la figure A.6 : un pour récupérer les mesures transmises sans fil par les capteurs et l'autre pour envoyer les données à l'ordinateur :

- Le premier composant est une carte MPR2600 qui récupère les données transmises par les capteurs via des ondes radio.
- Le deuxième composant est une carte MIB520 qui est dotée d'une liaison filaire vers l'ordinateur via un câble USB. Cette carte reçoit des données de la carte MPR2600 et les transmet à l'ordinateur via le câble USB. Elle sert également à injecter des programmes sur les capteurs MicaZ afin de faire des tâches bien précises.

Fig A.6- Station de base

A.2.2 Technologies sans-fil

Bluetooth [126] et Zigbee [127] sont les standards les plus aptes à être exploités dans les réseaux de capteurs sans-fil. Le succès de la technologie Bluetooth l'a amené à être utilisée dans le cadre des LAN sans fil. Malheureusement, cette technologie présente deux inconvénients majeurs : elle consomme une grande quantité d'énergie pour assurer les communications entre les nœuds et elle ne permet pas de connecter plus de 80 nœuds (10 piconets). Ainsi, elle ne peut pas être utilisée par des capteurs qui sont généralement déployés en grand nombre dans des zones hostiles, et qui présentent une autonomie d'énergie (batteries non rechargeables) et qui devraient idéalement fonctionner pour une longue durée. Par contre, le standard Zigbee offre des caractéristiques qui répondent aux besoins des réseaux de capteurs puisqu'il consomme moins d'énergie que Bluetooth. Le petit débit qu'il offre, n'est pas vraiment un handicap pour un réseau de capteurs puisque la taille des paquets échangés n'est pas vraiment importante.

A.2.3 Outils logiciels

La réalisation de la plateforme d'expérimentation nécessite le choix d'un système d'exploitation, d'un émulateur de capteurs et d'un langage de programmation permettant d'implémenter des applications sur les capteurs.

Le système d'exploitation : TinyOs

TinyOs [128] est un système d'exploitation orienté événement (event-driven) conçu pour les réseaux de capteurs. Il est implémenté entièrement en Nesc et il respecte une architecture basée sur une association de composants, permettant de réduire la taille du code nécessaire à sa mise en place. TinyOs occupe un espace mémoire très faible dans sa distribution minimale (512 octets). Par conséquent, il s'adapte aux capteurs pourvus de ressources mémoires très limitées. Pour autant, la bibliothèque de composants de TinyOs est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données.

Une application s'exécutant sur TinyOs est constituée d'une sélection de composants systèmes et de composants développés spécifiquement pour l'application. Parmi les principaux composants systèmes, on trouve l'ordonnanceur TinyOs. Cet élément est responsable de la gestion des tâches et des évènements du système. Son choix déterminera le fonctionnement global du système et le dotera de propriétés précises telles que la capacité à fonctionner en évènementiel.

Le langage de programmation NesC

NesC [129] est un langage orienté composant, conçu pour incarner les concepts de structuration et d'exécution du système TinyOs. Il propose une architecture basée sur des composants, permettant de réduire la taille mémoire du système TinyOs et de ses applications. Un composant peut être un concept abstrait ou un élément matériel tel que LEDs, timer, ou ADC et il peut être réutilisé dans différentes applications. Ces applications sont des ensembles de composants associés entre eux dans un but précis.

En NesC, les composants présentent des similarités avec des objets. Les états sont encapsulés et on peut y accéder par des interfaces. L'implémentation des composants est réalisée par la déclaration des tâches, des commandes ou des événements. En outre, l'ensemble des composants et leurs interactions sont fixés à la compilation ce qui permet d'optimiser l'application pour une exécution plus performante.

NesC permet de déclarer deux types de fichiers : les configurations et les modules. Le fichier configuration est la définition du ou des composants qui seront utilisés par l'application déployée sur le capteur. Par exemple, l'application présentée par la figure utilise le composant LedsC, ce qui permet de manipuler les leds du capteur. Les modules constituent les briques élémentaires du code et implémentent une ou plusieurs interfaces. Les interfaces spécifient un ensemble de fonctions à mettre en application par le fournisseur de l'interface (commandes) et par l'utilisateur de l'interface (événements). Ceci permet à une interface simple de représenter une interaction complexe entre les composants (par exemple, enregistrement d'un événement pertinent, suivi d'un rappel de service quand cet événement se produit). En spécifiant des interfaces, un composant ne peut pas appeler la commande d'envoi à moins qu'il fournisse une exécution de l'événement sendDone. En outre, les interfaces permettent de lier statiquement les composants entre eux. Ceci augmente l'efficacité d'exécution, favorise la robustesse de l'application, et permet une meilleure analyse statique du programme de l'application.

Serial Forwarder

Le programme Serial Forwarder est équipé de TinyOs et est utilisé pour lire un paquet de données à partir d'un port série et l'expédier via une connexion Internet. Ainsi, une fois que le programme Serial Forwarder est exécuté, il écoute les connexions client sur un port TCP donné et expédie des messages TinyOs via ce port, et vice versa. En outre, les applications peuvent être reliées à Serial Forwarder qui agit en tant que passerelle entre le port série et le réseau radio.

MIG (Message Interface Generator)

TinyOs fournit des outils pour produire automatiquement des objets message des descriptions de paquet. Ainsi, au lieu d'analyser les formats de paquet manuellement, l'outil de MIG établit une interface Java à la structure de message. En effet, pour une séquence d'octets donnée, le générateur de code MIG devra analyser automatiquement chacun des champs des paquets et fournit un ensemble d'accédants et de mutators standard pour visualiser les paquets reçus ou produire des nouveaux paquets.

A.3 DataMoteMonitor : Envoi d'une alerte

L'application que nous avons développée est une application Java, nommée "DataMoteMonitor". Elle utilise le protocole Serial Forwarder pour recevoir les données envoyées par les capteurs par le biais du protocole TCP. Ces données collectées sont stockées dans une base de données Mysql en utilisant javaMysqlConnector API. En outre, pour tester DataMoteMonitor, nous avons utilisé l'application Mote nommée "XSensorMST400" qui envoie les mesures suivantes : le niveau de la batterie, la température et l'humidité.

Le développement de DataMoteMonitor s'articule autour des modules suivants :

A.3.1 Le protocole Serial Forwarder

La communication entre l'application client et Serial Forwarder se fait de la manière suivante : Serial Forwarder envoie des paquets aux réseaux de capteurs à travers une connexion TCP en utilisant un protocole spécifique. Ce protocole pourvoit pour l'échange un identifiant quand la connexion est établie, ceci est dans le but est d'assurer que les deux côtés connaissent le format du paquet qui devra être transmis via ce protocole. Ainsi, après l'ouverture de la connexion TCP, le client envoie une trame d'initialisation et attend la réception d'une autre trame de la part de Serial Forwarder. Ensuite, après cette phase d'échange de versions de format de paquet, la version utilisée sera la minimale des deux versions échangées entre les deux. Si la version utilisée est égale à 33, alors le client envoie la trame d'identification de sa plateforme. Puis, il attend la réception de la trame d'identification de la plateforme. Cette trame est dont la taille est de quatre octets. Par contre, si la version utilisée est égale à 32, il n'y a pas d'échange de trame d'identification de plateforme entre les deux et le client attend la réception des trames de données et envoie d'autres si nécessaire.

Le format de la trame utilisé par le protocole Serial Forwarder :

- **Trame d'initialisation**

  La trame d'initialisation contient deux champs :

  – SFP Cookie : valeur utilisée pour identifier le protocole Serial Forwarder. Elle est presque toujours égale à 84.

  – SFP Version : elle prend deux valeurs pour connaître la version de la plateforme utilisée :

    • 32 : aucune identification de la plateforme ne sera effectuée

    • 33 : l'identification de la plateforme doit s'effectuer

- **Trame d'identification de la plateforme**

L'identifiant de la plateforme est utilisé pour spécifier le format du payload qui sera contenu dans les champs des paquets de données. DataMoteMonitor utilise l'identifiant de la plateforme associé aux capteurs MicaZ. Voici, les différents identifiants associés aux différents capteurs fabriquées par différentes firmes.

1 : TOSMsg est le format utilisé par les capteurs Mica, Mica2, Mica2dot.

2 : IEEE 802.15.4 est le format utilisé par la famille des capteurs Telos.

3 : TOSMsg modifié est le format utilisé par les capteurs MicaZ.

4 : est le format utilisé par les capteurs eyes.

- **Trame de données**

La trame de données contient les valeurs collectées par les capteurs. La taille maximale d'un paquet de données est de 255 octets. Le champ Length indique le nombre d'octets dans la zone correspondante (figure A.7). Ainsi, une fois que la communication commence, les trames générées par XSensorMTS400 sont encapsulées dans la structure TinyOs. La figure A.8 montre la structure d'un paquet TinyOs. La taille du paquet de données est mentionnée dans le $5^{\text{ième}}$ octet. Le premier et le deuxième octets indiquent l'adresse de diffusion (FFFF : Broadcast). Le troisième et le quatrième octets représentent le type et le groupe du paquet (AM Type et AM Group).

Fig A.7- Trame de données

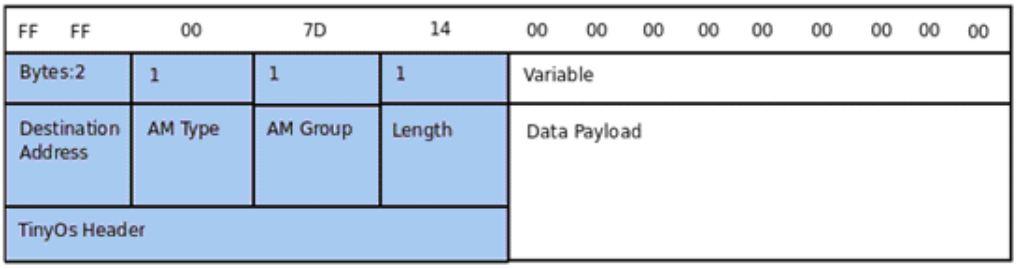


Fig A.8- Trame TinyOs

- **Trame XSensorMTS400**

Les trames XSensorMTS400 sont encapsulées dans des trames TinyOs. Un en-tête de XsensorMTS400 est inséré dans des trames TinyOs entre l'en-tête et les données (figure A.9).

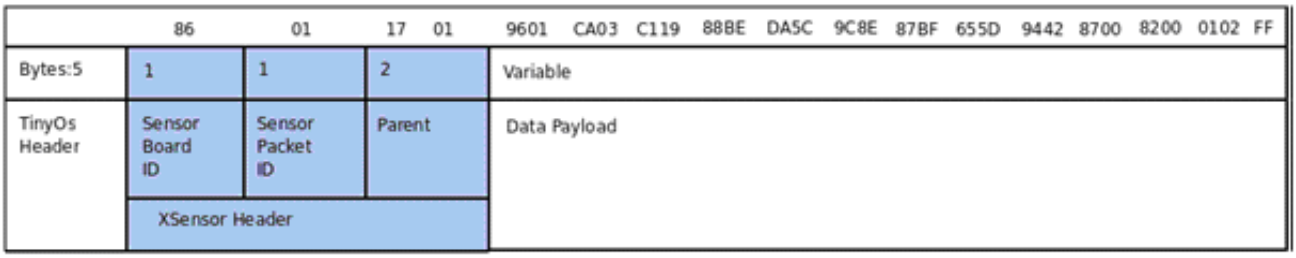


Fig A.9- Trame XSensorMTS400

Le premier octet de l'en-tête de XSensorMTS400 spécifie le type de carte utilisée. Dans cet exemple, sa valeur est 86, ce qui correspond aux cartes de type MTS400/420. L'octet suivant représente l'identifiant (ID) du capteur (01). Le troisième et le quatrième octets représentent le parent c'est à dire le capteur qui a envoyée cette trame à ce capteur. Les données contenues dans la trame sont :

– 96 01 : le voltage de la batterie
– CA 03 : l'humidité

– C1 19 : la température
– 94 42 : la pression
– 87 00 : la luminosité
– 01 02 : la position X
– FF : la position Y

En outre, les données collectées par les capteurs, sont exprimées en bits et leurs positions sont indexées relativement au premier bit de l'en-tête de la trame XSensorMTS400 (offset). Par exemple, la donnée qui retourne la température est indexée au $64^{\text{ième}}$ bit (offset 64), et la donnée qui retourne la tension à $32^{\text{ième}}$ bit de l'en-tête de la trame XSensorMTS400. En outre, les valeurs contenues dans un paquet de données sont des valeurs brutes, ce qui nécessite de les transformer en valeurs réelles.

Les formules (1), (2) et (3) permettent de transformer respectivement la tension, l'humidité et la température en valeurs réelles.

$$\textit{Voltage\_de\_Batterie} = \frac{1252352}{valeur + 1000} \tag{1}$$

$$\textit{Humidité} = valeur \times \frac{1}{11.5} + 34 \tag{2}$$

$$\textit{Temperature} = -38.4 + (0.0098 \times valeur) \tag{3}$$

A.3.2 Extraction des champs d'un paquet XSensorMTS400

Pour avoir des données utilisables d'un capteur, la spécification de paquet doit être établie pour analyser les données de Serial Forwarder : température, humidité et la tension de la batterie. Le fichier de spécification du paquet XSensorMTS400 est fourni avec le programme XSensorMTS400 en langage C. Ainsi, le MIG peut convertir les spécifications en classe Java pour être utilisées par DataMoteMonitor. La commande suivante est exécutée pour créer des spécifications Java :

mig java -target=micaz -java-classname=XSensorMTS400 sensorboard.h sensorboard -o

où :

– "target=micaz" : ce paramètre précise le type de capteur

- "java-classname=XSensorMTS400" est le nom de la classe Java générée
- "sensorboard.h" est le programme de spécification du capteur
- Sensorboard est la fonction principale dans le fichier de spécification

La classe message produite par les outils source de TinyOs fournit les méthodes pour extraire les valeurs brutes à partir des paquets transmis par le programme XSensorMTS400 et les méthodes pour convertir ces valeurs brutes en valeurs réelles telles que :

1. la méthode CalculTemperature() qui permet de convertir les valeurs brutes en valeurs réelles en utilisant la formule 6.3.

2. la méthode getUIntElement() est la méthode la plus utilisée. Elle permet d'extraire des octets d'un paquet de données à partir d'une position donnée et les convertir en valeur décimale. Après cette opération, les méthodes de conversion sont appelées pour calculer les valeurs réelles.

3. la méthode getTemperature() retourne la valeur correspondante. La méthode getUIntElement() est appelée pour obtenir les données de la trame, puis la méthode de calcul sera exécutée.

A.3.3 Différents phases du processus d'exécution

Une fois que le protocole Serial Forwarder est au courant des spécifications de XSensorMTS400 et DataMoteMonitor, la phase de codage est initiée. Cette phase est composée de trois parties :

- Communication réseau : développement de sockets, exécution du protocole Serial Forwarder et structuration des paquets.
- DataMonitoring : collecte de données par les capteurs et envoie des alertes quand des événements pertinents surviennent.
- Base de données : développement de l'interface objet de la base de données.

Communication réseau

Les communications réseau dans DataMoteMonitor sont basées sur des sockets Java. Le protocole Serial Forwarder utilisé dans le code source de TinyOs a été réécrit dans DataMoteMonitor et le modèle d'événement a été utilisé pour créer le mécanisme d'événement pour superviser les paquets reçus.

Le mécanisme des communications réseau dans DataMoteMonitor est présenté par le diagramme des classes du package Network :

- La classe NetworkManager : Le Thread de la classe Network nécessite d'être exécuté en arrière plan d'une manière continue pour recevoir sans interruption les paquets envoyés par le réseau de capteurs déployé. La classe principale du package Network instancie la classe de socket SFSource et implémente toutes les méthodes nécessaires pour la conception du modèle. Elle stocke également l'objet FPacketListener et appelle la méthode onPacketReceived() quand des paquets sont reçus.
- La classe SFProtocol : fournit les méthodes pour recouvrir des données pour être conformes au protocole Serial Forwarder.
- La classe SFPacketListener : fournit la définition de onPacketReceived où l'action à réaliser devra être inscrite.
- La classe SFPacketEvent : symbolise un événement créé quand un paquet est reçu. Celui-ci est stocké dans SFPacketEvent pour être utilisé par la méthode onPacketReceived.
- La classe NtConfig : c'est la classe qui configure le réseau. Elle est composée de l'adresse IP de Serial Forwarder et le numéro de port.

Quand NetworkManager est instancié, le SFSource est instancié aussi, et une socket est créée pour relier DataMoteMonitor à Serial Forwarder. Chaque fois qu'un paquet est reçu, la méthode onPacketReceived est appelée pour informer DataMoteMonitor d'un paquet reçu.

A.4 Conclusion

Dans DataMoteMonitor, chaque capteur mesure les grandeurs suivantes : le voltage de la batterie, l'humidité et la température de sa zone de couverture. Puis, il les envoie à une station de base connectée à un ordinateur domestique. L'ordinateur domestique compare les valeurs reçues aux valeurs seuils. Si une des grandeurs collectées dépasse la valeur seuil alors il fait remonter une alarme à un centre de contrôle distant. Dans notre contexte, une alarme est remontée au centre de contrôle lorsque la température dépasse la valeur seuil T_α, mais tout autre traitement peut être envisagé.

Par ailleurs, DataMoteMonitor est une application générique. En effet, dans la plupart des cas, chaque capteur mesure plusieurs grandeurs et si une des grandeurs collectées dépasse la

valeur seuil alors il fait remonter une alarme au centre de contrôle par l'intermédiaire de la station de base via une connexion Internet. En outre, l'alarme remontée au centre de contrôle peut prendre plusieurs formes : Message, SMS, Mail.

La zone de couverture peut être si vaste et le nombre de capteurs déployés peut être si important, par suite l'application du protocole de couverture de zone CSA_VS que nous avons proposée dans le sixième chapitre, peut assurer la surveillance totale de la zone d'intérêt pour une longue durée.

Bibliographie personnelle

1. M. Lehsaini, H. Guyennet, and M. Feham. CES : Cluster-based Energy-efficient Scheme for Mobile Wireless Sensor Networks. Chapter in the book Wireless Sensor and Actor Networks II (Boston Springer), vol.264, pp.13-24, Mai 2008.

2. M. Lehsaini, H. Guyennet, and M. Feham. Cluster-based Self-Organization Scheme for Mobile Wireless Sensor Networks. International Review on Computers and Software (IRECOS), March 2008.

3. M. Lehsaini, H. Guyennet, and M. Feham. MPR-based broadcasting in ad hoc and sensor networks with a realistic environment. International Journal of Computer Science and Network Security (IJCSNS), vol.7, no.10, pp.82-89, October 2007.

4. M. Lehsaini, H. Guyennet, and M. Feham. An Efficient Cluster-based Self-organization Algorithm for Wireless Sensor Networks. International Journal Sensor Networks (Inderscience), To appear.

5. M. Lehsaini, H. Guyennet and M. Feham. α-coverage Scheme for Wireless Sensor Networks. In Proceedings of the fourth (IEEE & ACM) International Conference on Wireless and Mobile Communications (ICWMC'08), pp.91-96, Athens, Greece, July 2008.

6. M. Lehsaini, H. Guyennet, and M. Feham. Cluster-based Energy-efficient Scheme for Mobile Wireless Sensor Networks. In Proceedings of the IFIP Conference on Wireless Sensor and Actor Networks, Ottawa, Ontario, Canada, July 2008.

7. M. Lehsaini, H. Guyennet, and M. Feham. A Novel Cluster-based Self-organization Algorithm for Wireless Sensor Networks. In Proceedings of the IEEE International Symposium on Collaborative Technologies and Systems (CTS08), pp.19-26, California, USA, Mai 2008.

8. M. Lehsaini et M. Feham. Conception et implémentation d'un réseau domestique dans l'habitat. Conférence Internationale MCSEAI'06, Université Ibn Zohr Agadir, Maroc, Décembre 2006.

9. M. Lehsaini et M. Feham. Conception d'une architecture basée sur Bluetooth pour les réseaux à domicile. Conférence Internationale sur l'Informatique et ses Applications CIIA06, Université de Saida, Algérie, Mai 2006.

Bibliographie

[1] Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In Proceedings of International Symposium on Information Processing in Sensor Networks, pp.424-432, New York, NY, USA, 2004. ACM Press.

[2] C. Adjih, P. Jaquet, and L. Viennot. Computing Connected Dominated Sets with Multipoint Relays. International Journal Ad Hoc & Sensor Wireless Networks, vol.1, no.1-2, pp.27-39, March 2005.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks : a survey. Computer Networks (Elsevier), vol.38, no.4, pp.393-422, March 2002.

[4] J.N. Al-Karaki and A.E. Kamal. On the Correlated Data Gathering Problem in Wireless Sensor Networks. In Proceedings of The Ninth IEEE Symposium on Computers and Communications (ISCC'04), vol.1, pp.226-231, Alexandria, Egypt, July 2004.

[5] J.N. Al-Karaki, R. Ul-Mustafa, and A.E. Kamal. Data Aggregation in Wireless Sensor Networks - Exact and Approximate Algorithms. In Proceedings of IEEE Workshop on High Performance Switching and Routing (HPSR'04), pp.241-245, Phoenix, Arizona, USA, April 2004.

[6] K.M. Alzoubi, P.J Wan, and O. Frieder. New distributed algorithm for connected dominating set in wireless ad hoc networks. In Proceedings of the 35^{th} Annual Hawaii International Conference on System Sciences (HICSS'02), pp.3849-3855, Maui, Hawaii, January 2002.

[7] K.M. Alzoubi, P.J. Wan, and O. Frieder. Distributed heuristics for connected dominating set in wireless ad hoc networks. Journal Of Communication and Networks, vol.4, no.1, pp.22-29, March 2002.

[8] A.D. Amis and R. Prakash. Load-Balancing Clusters in Wireless Ad Hoc Networks. In Proceedings 3^{rd} IEEE Symposium on Application-Specific Systems and Software Engineering Technology, pp.25-32, Los Alamitos, CA, March 2000.

[9] A.D. Amis, R. Prakash, T.H.P Vuong, and D.T. Huynh. Max-Min D-Cluster Formation in Wireless Ad Hoc Networks. In Proceedings of IEEE Conference on Computer Communications (INFOCOM'00), vol.1, pp.32-41, Anchorage, Alaska, March 2000.

[10] B. An and S. Papavassiliou. A mobility-based clustering approach to support mobility management and multicast routing in mobile ad hoc wireless networks. International Journal of Network Management, vol.11, no.6, pp.387-395, November/December 2001.

[11] K. Arisha, M. Youssef, and M. Younis. Energy-Aware TDMA-Based MAC for Sensor Networks. In Proceedings of the IEEE Workshop Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT 2002), New York, USA, May 2002.

[12] D.J. Baker and A. Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. IEEE Transactions on Communications, vol.29, no.11, pp.1694-1701, November 1981.

[13] S. Banerjee and S. Khuller. A clustering scheme for hierarchical control in multihop wireless networks. In Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), pp.1028-1037, Anchorage, Alaska, USA, April 2001.

[14] S. Basagni. Distributed clustering for ad hoc networks. In Proceedings of Fourth International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN'99), pp.310-315, Perth/Fremantle, WA, Australia, June 1999.

[15] S. Basagni. Distributed and mobility-adaptive clustering for multimedia support in multihop wireless networks. In Proceeding of Vehicular Technology Conference (VTF'99), vol.2, pp.889-893, September 1999.

[16] P. Basu, N. Khan, and T.D.C. Little. A mobility Based Metric for Clustering in Mobile and Ad Hoc Networks. In Proceedings of International Conference of Distributed Computing Systems Workshop (ICDCSW'01), pp.413-418, Phoenix, Arizona, USA, April 2001.

[17] M. J. Brown. Users Guide Developed for the JBREWS Project. Technical Report LA-UR-99-4676. Los Alamos National Laboratory of California University. 1999.

[18] N. Bulusu, J. Heidemann, and D. Estrin. GPS-Less Low Cost Outdoor Localization for Very Small Devices. IEEE Personal Communications Magazine, vol.7, no.5, pp.28-34, October 2000.

[19] Y. Cai, M. Li, W. Shu, and M.Y. Wu. Acos : A precise energy-aware coverage control protocol for wireless sensor networks. International Journal Ad Hoc Sensor Wireless Networks, vol.3, no.1, pp.77-98, 2007.

[20] M. Cardei, D. MacCallum, X. Cheng, M. Min, X. Jia, D. Li, and D.Z. Du. Wireless Sensor Networks with Energy Efficient Organization. Journal of Interconnection Networks, vol.3, no. 3-4, pp.213-229, 2002.

[21] M. Cardei, X. Cheng, and D.Z. Du. Connected Domination in Ad Hoc Wireless Networks. In Proceedings of Sixth International Conference on Computer Science and Informatics, North Carolina, USA, Mar. 2002.

[22] J. Carle and D. Simplot-Ryl. Energy efficient area monitoring for sensor networks. IEEE Computer Society Magazine, vol.37, no.2, pp.40-46, February 2004.

[23] C.C Chang, H.K Wu, W. Liu, and M. Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. In Proceedings of IEEE Singapore International Conference on Networks (SICON'97), pp.197-212, Singapore, April 1997.

[24] M. Chatterjee, S. K. Das, and D. Turgut. WCA : A weighted clustering algorithm for mobile ad hoc networks. Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks), vol.5, no.2, pp.193-204, April 2002.

[25] M. Chatterjee, S. K. Das, and D. Turgut. A weight based distributed clustering algorithm for mobile ad hoc networks. In Proceedings of the International Conference on High Performance Computing, IEEE & ACM (sponsors), pp.511-521, Bangalore, India, December 2000.

[26] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. : an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. ACM Wireless Networks Journal, vol.8, no.5, pp.481-494, September 2002.

[27] G. Chen and I. Stojmenović. Clustering and routing in mobile wireless networks. Technical Report TR-99-05, University of Ottawa, June 1999.

[28] G. Chen, F.G. Nocetti, J.S. Gonzalez, I. Stojmenović. Connectivity based k-hop clustering in wireless networks. In Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02), pp.2450- 2459, Maui, Hawaii, January 2002.

[29] X. Cheng, M. Ding, D.H. Du, and X. Jia. Virtual backbone construction in multihop ad hoc wireless networks. International Journal of Wireless Communications and Mobile Computing, vol.6, no.2, pp.183-190, March 2006.

[30] X. Cheng, X. Huang, D. Li, W. Wu, and D.Z. Du. Polynomial-Time Approximation Scheme for Minimum Connected Dominating Set in Ad Hoc Wireless Networks. Networks International Journal. vol.42, no.4, pp.202-208, September 2003.

[31] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit Disk Graphs. Discrete Mathematics, vol.86, pp.165-177, 1990.

[32] C.Y. Chong and S.P. Kumar. Sensor Networks : Evolution, Opportunities, and Challenges. In Proceedings of the IEEE, vol.91, no.8, pp. 1247-1256, 2003.

[33] F. Dai and J. Wu. Distributed Dominant Pruning in Ad Hoc Networks. In Proceedings of the IEEE 2003 International Conference on Communications (ICC'2003), vol.1, pp.353-357, May 2003.

[34] F. Dai and J. Wu. On Constructing k-Connected k-Dominating Set in Wireless Ad Hoc and Sensor Networks. Journal of Parallel and Distributed Computing, vol.66 , no.7, pp.947-958, July 2006.

[35] S. Das, C. Perkins, and E. Royer. Ad hoc On Demand Distance Vector Routing (AODV). Internet Draft, IETF MANET Working Group, November 2001.

[36] I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks : a survey. IEEE Communications Magazine, vol.4, no.4, pp.115-121, April 2006.

[37] L. Doherty, K.S.J. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In Proceedings of the IEEE INFOCOM, vol.3, pp.1655-1663, Alaska, 2001.

[38] C. Enz, A. El-Hoiydi, J.D. Decotignie, and V. Pereis. WiseNET : An ultralow-power wireless sensor network solution. IEEE Computer Society Magazine, vol.37, no.8, pp.62-70, August 2004.

[39] A. Ephremides, J. E. Wieselthier, and D.J. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. In Proceedings of the IEEE, vol.75, no.1, pp.56-73, January 1987.

[40] Q. Fang, F. Zhao, and L. Guibas. Lightweight Sensing and Communication Protocols for Target Enumeration and Aggregation. In Proceedings of the 4^{th} ACM International Symposium on Mobile Ad hoc Networking and Computing (MOBIHOC 2003), pp.165-176, Annapolis, Maryland, June 2003.

[41] L. Feeney. An energy-consumption model for performance analysis of routing protocols for mobile ad hoc networks. ACM Journal of Mobile Networks and Applications, vol.3, no.6, pp.239-249, 2001.

[42] Y. Fernandess and D. Malkhi. K-clustering in wireless ad hoc networks. In Proceedings of the second ACM International Workshop on Principles of Mobile Computing, pp.31-37, Toulouse, France, October 2002. ACM Press.

[43] M. Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. ACM/Baltzer Journal of Wireless Networks, vol.1, no.3, pp.255-265, 1995.

[44] T.B. Gosnell, J.M. Hall, C.L. Ham, D.A. Knapp, Z.M. Koenig, S.J. Luke, B.A. Pohl, A. Schach von Wittenau, and J.K. Wolford. Gamma-Ray Identification of Nuclear Weapon Materials. Technical Report DE97053424. Lawrence Livermore National Lab., Livermore, CA (USA). February 1997.

[45] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. Algorithmica, vol.20, no.4, pp.374-387, April 1998.

[46] C. Gui and P. Mohapatra. Power conservation and quality of surveillance in target tracking sensor networks. In Proceedings of ACM Mobile Computing and networking (Mobicom), pp.129-143, Philadelphia, PA, USA, 2004.

[47] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Micro Sensor Networks. In IEEE Proceedings of the Hawaii International Conference on System Sciences (HICSS '00), 2000.

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

[49] G. Hoblos, M. Staroswiecki, and A. Aitouche. Optimal design of fault tolerant sensor networks. In proceedings of the 2000 IEEE International Conference on Control Applications, pp.467-472, 2000.

[50] C.F. Hsin and M. Liu. Network coverage using low duty-cycled sensors : random coordinated sleep algorithms. In Proceedings of the 3^{rd} International Symposium on Information Processing in Sensor Networks (IPSN), pp.433-442, Berkeley, CA, USA, 2004.

[51] Y. Huang and Y. Tseng. The coverage problem in a wireless sensor network. In Proceedings of the ACM International Conference on Wireless Sensor Networks and Applications, San Diego, CA, September 2003.

[52] F. Ingelrest and D. Simplot-Ryl. Maximizing the probability of delivery of multipoint relay broadcast protocol in wireless ad hoc networks with a realistic physical layer. Mobile Ad-hoc and Sensor Networks, Lecture Notes in Computer Science (Springer), vol.4325 pp.143-154, November 2006.

[53] P. Jacquet, P. Mühlethaler, T. Clausen, A. Qayyum, L. Viennot, and A. Laouiti. Optimized link state routing protocol for ad hoc networks. In Proceedings of the IEEE International Muti-topic Conference (INMIC'01), pp.62-68, Lahore, Pakistan, December 2001.

[54] J.G. Jetcheva, Y. Hu, D. Johnson, and D. Maltz. The Dynamic Source Routing Protocol for Mobile AdŰhoc Networks (DSR). Internet Draft, IETF MANET working group, November 2001.

[55] J. Jiang and W. Dou. A coverage preserving density control algorithm for wireless sensor networks. In Proceedings of 3^{rd} International Conference on AD-HOC Networks and Wireless (ADHOC-NOW), pp.42-45, Vancouver, BC, Canada, 2004.

[56] M. Jiang, J. Li, and Y.C Tay. Cluster-Based Routing Protocol(CBRP). INTERNET-DRAFT, draft-ietf-manet-cbrp-spec-01.txt, August 1999.

[57] P. Johnson and D.C Andrews. Remote continuous monitoring in the home. Journal of Telemedicine and Telecare, vol.2, no.2, pp.107-113, June 1996.

[58] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell. Fault Tolerance Techniques for Wireless Ad Hoc Sensor Networks. In proceedings of IEEE Sensors, vol.2, pp.1491-1496, 2002.

[59] P. Krishna, N.H. Vaidya, M. Chatterjee, and D.K. Pradhan. A Cluster-based Approach for Routing in Dynamic Networks. ACM SIGCOMM Computer Commnications Review, vol.27, no.2, pp.49-64, April 1997.

[60] J. Kurvilla, A. Nayak, and I. Stojmenović. Hop count optimal position based packet routing algorithms for ad hoc wireless networks with a realistic physical layer. In proceedings of IEEE international conference on Mobile Ad hoc and Sensor Systems (MASS'04), pp. 398-405, Fort Lauderbale, USA, October 2004.

[61] B.J. Kwar, N.O.Song, and L.Miller. On the scalability of ad hoc networks. In Proceedings of the IEEE Communication letters, 2004.

[62] D.T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. IEEE Transactions on Computers, vol.C-31, pp.478-487, 1982.

[63] G. Lee, J. Kong , M. Lee, and O. Byeon. A Cluster-based Energy Aware Routing Protocol for Sensor Networks. In Proceedings of the International Conference On Parallel and Distributed Computing and Systems, Phoenix, AZ, USA, 2005.

[64] M. Lehsaini et M. Feham. Conception et implémentation d'un réseau domestique dans l'habitat. Conférence Internationale MCSEAI'06, Université Ibn Zohr Agadir (Maroc), Décembre 2006.

[65] M. Lehsaini et Feham. Conception d'une architecture basée sur Bluetooth pour les réseaux à domicile. Conférence Internationale sur l'Informatique et ses Applications CIIA06, Université de Saida (Algérie), Mai 2006.

[66] M. Lehsaini, H. Guyennet, and M. Feham. MPR-based broadcasting in ad hoc and sensor networks with a realistic environment. International Journal of Computer Science and Network Security (IJCSNS), vol.7, no.10, pp.82-89, October 2007.

[67] M. Lehsaini, H. Guyennet, M. Feham .A novel cluster-based self-organization algorithm for wireless sensor networks. In Proceedings of the IEEE International Symposium on Collaborative Technologies and Systems (CTS 2008), pp.19-26, Irvine (California), USA, May 2008.

[68] M. Lehsaini, H. Guyennet and M. Feham. α-coverage Scheme for Wireless Sensor Networks. In Proceedings of the fourth (IEEE & ACM) International Conference on Wireless and Mobile Communications (ICWMC'08), pp.91-96, Athens, Greece, July 2008.

[69] M. Lehsaini, H. Guyennet, and M. Feham. Cluster-based Self-Organization Scheme for Mobile Wireless Sensor Networks. International Review on Computers and Software (IRECOS), March 2008.

[70] M. Lehsaini, H. Guyennet, and M. Feham. CES : Cluster-based Energy-efficient Scheme for Mobile Wireless Sensor Networks. Chapter in the book Wireless Sensor and Actor Networks II (Boston Springer), vol.264, pp.13-24, Mai 2008.

[71] M. Lehsaini, H. Guyennet, and M. Feham. Cluster-based Energy-efficient Scheme for Mobile Wireless Sensor Networks. In Proceedings of the IFIP Conference on Wireless Sensor and Actor Networks, Ottawa, Ontario, Canada, July 2008.

[72] N. Li and J.C. Hou. FLSS : a fault-tolerant topology control algorithm for wireless networks. In Proceedings of MobiCom, pp.275-286, Sep./Oct. 2004.

[73] X.Y. Li, P.J. Wan, Y. Wang, and C.W. Yi. Fault tolerant deployment and topology control in wireless networks. In Proceedings of MobiHoc, pp.117-128, June 2003.

[74] H.C. Lin, and Y.H. Chu. A clustering technique for large multihop mobile wireless networks. In proceedings of the IEEE Vehicular Technology Conference, vol.2, pp.1545-1549, May 2000.

[75] C.R. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. IEEE Journal of Selected Areas in Communications, vol.15, no.7, pp.1265-1275, September 1997.

[76] P. Lin, C. Qiao, and X. Wang. Medium access control with a dynamic duty cycle for sensor networks. In IEEE Wireless Communications and Networking Conference (WCNC'04), vol.3, pp.1534-1539, March 2004.

[77] S. Lindsey and C.S. Raghavendra. PEGASIS : Power-Efficient Gathering in Sensor Information Systems. IEEE Transaction on Parallel and Distributed Systems, vol.13, no.9, pp.924-935, September 2002.

[78] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks. In Proceedings IPDPSŠ04, pp.224-231, April 2004.

[79] A. Manjeshwar and D.P. Agrawal. TEEN : A Protocol for Enhanced Efficiency in Wireless Sensor Networks. 1^{st} International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, San Francisco, CA, April 2001.

[80] M.V. Marathe, H. Breu, H.B. Hunt III, S.S Ravi, and D.J. Rosenkrantz. Simple heuristics for unit disk graphs. Networks, vol.25, pp.59-68, December 1995.

[81] V. Mhatre , and C. Rosenberg. Design guidelines for wireless sensor networks : communication, clustering and aggregation. Ad Hoc Networks Journal, vol.2, no.1, pp.45-63, 2004.

[82] S.D. Muruganathan, D.C.F. Ma, R.I. Bhasin, and A.O. Fapojuwo. A Centralized Energy-Efficient Routing Protocol for Wireless Sensor Networks. IEEE Communication Magazine, vol.43, no.3, pp.S8-S13, 2005.

[83] W. Navidi and T. Camp. Stationary distributions for the random waypoint mobility model. IEEE Transactions on Mobile Computing, vo.3, no.1, 2004.

[84] A. Okabe, B. Boots, and K. Sugihara. Spatial Tessellations : Concepts and Applications of Voronoi diagrams. 2^{nd} Edition. Wiley, Chichester. pp.671, 2000.

[85] E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis, and V.Z. Groza. Sensor-based information appliances. IEEE Instrumentation Measurement Magazine, vol.3, no.4, pp.31-35, December 2000.

[86] J. Polastre, J. Hill, and D.E. Culler. Versatile low power media access for wireless sensor networks. In Proceedings of the 2^{nd} International Conference on Embedded Networked Sensor Systems (SenSys'04), pp.95-107, Baltimore, MD, USA, November 2004. ACM press.

[87] J. Polastre, R. Szewczyk, and D. Culler. Telos : Enabling Ultra-Low Power Wireless Research. In proceedings of the 4^{th} International Conference on Information Processing in

Sensor Networks : Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN'05), pp.364-369, April 2005

[88] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In Proceedings of the 35^{th} Annual Hawaii International Conference on System Sciences (HICSS'02), pp.3866-3875, Big Island, Hawaii, USA, January 2002.

[89] L.Quin and T.Kunz. On-demand routing in MANETs : The impact of a realistic physical layer model. In proceedings of the International Conference on Ad hoc, Mobile, and Wireless Networks, pp.37-48, Montreal, Canada, 2003.

[90] V. Raghunathan, C. Schurgers, S. Park, and M.B. Srivastava. Energy-Aware Wireless Microsensor Networks. IEEE Signal Processing Magazine, vol.19, pp.40-50, 2002.

[91] V. Rodoplu and T. H.Y. Meng. Minimum energy mobile wireless networks. IEEE Journal on Selected Areas in Communications, vol.17, no.8, pp.1333-1344, August 1999.

[92] A. Safwat and H. Hassanein. Infrastructure-based routing in wireless mobile ad hoc networks. International Journal of Computer Communications, vol.25, no.3, pp.210-224, 2002.

[93] C.Santivanez, B.McDonald, I.Stavrakakis, and R.Ramanathan. On the scalability of ad hoc routing protocols. In Proceedings of Infocom, 2002.

[94] A. Savvides, C.C. Han, and M. Srivastava. Dynamic fine-grained localization in Ad-Hoc networks of sensors. In Proceedings of the 7^{th} ACM Annual International Conference on Mobile Computing and Networking (MobiCom), pp.166-179, July 2001.

[95] J.P. Sheu, C.H. Yu, and S.C. Tu. A distributed protocol for query execution in sensor networks. In IEEE Wireless Communications and Networking Conference (WCNC), vol.3, pp.1824-1829, New Orleans, LA, USA, 2005.

[96] S. Slijepcevic and M. Potkonjak. Power Efficient Organization of Wireless Sensor Networks. In proceedings of the IEEE International Conference on Communications, vol.2, pp.472-476, 2001.

[97] A.M.C. So and Y. Ye. On Solving Coverage Problems in a Wireless Sensor Network Using Voronoi Diagrams. In Workshop on Internet and Network Economics (WINE 2005), p.584-593. Hong Kong, December 2005.

[98] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor networks. Transactions on IEEE Personal Communications/Wireless Communications, vol.7, no.5, pp.16-27, October 2000.

[99] K. Sohrabi and G.J. Pottie. Performance of a novel self-organization protocol for wireless ad-hoc sensor networks. In Proceeding of the 50^{th} IEEE Vehicular Technology Conference (VTC'99), vol.2, pp.1222-1226, Amsterdam, September 1999.

[100] I. Stojmenović and M.Seddigh. Broadcasting algorithms in wireless networks. In proceedings of the international conference on Advances in Infrastructure for Electronic Busniss, Science, and Education on the Intenet SSGRR, L'Aquila, Italy, July/Aug. 2000.

[101] I. Stojmenović, M. Seddigh, and J. Zunic. Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks. IEEE Transactions on Parallel and Distributed Systems, vol.13, no.1, pp.14-25, January 2002.

[102] D. Tian and N. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In proceedings of ACM international conference on Wireless Sensor Networks and Applications (WSNA), pp.32-41, Atlanta, GA, USA, 2002.

[103] D. Tian and N.D. Georganas. Connectivity maintenance and coverage preservation in wireless sensor networks. AdHoc Networks Journal (Elsevier Science), pp.744-761, 2005.

[104] T. Van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In proceedings of the 1^{st} ACM International Conference on Embedded Networked Sensor Systems (SenSys'03), pp.171-180, Los Angeles, California, USA, November 2003.

[105] P.J Wan, K.M. Alzoubi, and O. Frieder. Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks. Review of Mobile Networks and Applications (Springer : Kluwer Academic Publishers), vol.9, no.2, pp.141Ű149, Avril 2004.

[106] A. Wang, S.H. Cho, C.G. Sodini, and A.P. Chandrakasan. Energy-Efficient Modulation and MAC for Asymmetric Microsensor Systems. In Proceedings of ISLPED 2001, Huntington Beach, CA, August 2001.

[107] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In proceedings of the 1^{st} ACM international conference on Embedded networked sensor systems, pp.28-39, Los Angeles, California, USA, 2003.

[108] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. ACM Transactions on Sensor Networks (TOSN), vol.1, no.1, pp.36-72, August 2005.

[109] J. Wu. Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links. IEEE Transactions on Parallel and Distributed Systems, vol.13, no.9, pp.866-881, September 2002.

[110] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In Proceedings of the 3^{rd} International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DialM'99), pp.7-14, Seattle,Washington, USA, August 1999.

[111] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed Energy Conservation for Adhoc Routing. In Proceedings of the 7^{th} Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'01), pp.70-84, Rome, Italy, July 2001.

[112] S. Yang, F. Dai, M. Cardei, and J. Wu. On multiple point coverage in wireless sensor networks. In proceedings of IEEE Mobile Ad hoc and Sensor Systems (MASS), 2005.

[113] S. Yang, F. Dai, M. Cardei, and J. Wu. On Connected Multiple Point Coverage in Wireless Sensor Networks. International Journal of Wireless Information Networks, vol.13, no.4, pp.286-301, October 2006.

[114] Y. Ye. o(n3l) potential reduction algorithm for linear programming. Mathematical Programming, vol.50, no.2, pp.239-258, 1991.

[115] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In Proceedings of the 21^{st} International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02), vol.3, pp.1567-1576, New York, NY, USA, June 2002.

[116] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. Transactions on IEEE/ACM Networking, vol.12, pp.493-506, June 2004.

[117] M. Ye, C. Li, G. Chen, and J. Wu. EECS : an energy efficient cluster scheme in wireless sensor networks. In proceedings of the IEEE International Workshop on Strategies for Energy Efficiency in Ad Hoc and Sensor Networks, Phoenix, Arizona, 2005.

[118] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. Peas : A robust energy conserving protocol for long-lived sensor networks. In Proceedings of International Conference on Distributed Computing Systems (ICDCS), pp. 28-37, Rhode Island, USA, 2003.

[119] O. Younes and S. Fahmy. HEED : A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. IEEE Transactions on Mobile Computing, vol.3, no.4, pp.366-379, October-December 2004.

[120] J.Y. Yu and P.H.J. Chong. 3hBAC : a novel non-overlapping clustering algorithm for mobile ad hoc networks. In IEEE Pacific Rim Conference on Communications, Computers and signal Processing (PACRIM'03), vol.1, pp.318-321, Victoria, Canada, August 2003.

[121] H. Zhang and J. Hou. Maintaining sensing coverage and connectivity in large sensor networks. International Journal Wireless Ad Hoc and Sensor Networks, vol.1, no.1-2, pp.89-123, 2005.

[122] Z. Zhou, S. Das, and H. Gupta. Connected k-coverage problem in sensor networks. In proceedings of 13^{th} International Conference on Computer Communications and Networks(ICCCN), pp.373-378, Chicago, IL, USA, 2004.

[123] MIT_μAMPS LEACH NS2 Extensions, www.ece.rochester.edu/research/wcng/code/index.html.

[124] http ://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf

[125] The Network Simulator ns-2, http ://www.isi.edu/nsnam/index.php.

[126] Bluetooth SIG, "Specification of the Bluetooth system, Core, Version 1.1", Bluetooth SIG Homepage : www.Bluetooth.com

[127] http ://www.ieee802.org/15/pub/TG4.html

[128] www.tinyos.net/tinyos-1.x/doc/

[129] http ://www.tinyos.net/tinyos-1.x/doc/nesc/ref.pdf

Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique

Au cours de ces dernières années, la technologie des réseaux sans fil n'a cessé de croître grâce aux développements technologiques dans divers domaines liés à la micro-électronique et à l'informatique. La mise en réseaux des capteurs sans fil permet d'étendre la couverture de la zone et la diffusion pour couvrir un territoire plus important, obtenir des résultats plus rapidement, plus fiables et permettre la tolérance aux fautes.

Au niveau de la diffusion, nous avons proposé un algorithme pour la diffusion dans un environnement réaliste basé sur l'approche MPR (Multi Points Relais) puis un algorithme d'auto-organisation appelé CSOS basé sur le concept de clustering. Ce dernier permet d'organiser les capteurs en 2-clusters homogènes en taille et de confier aux cluster-heads la responsabilité de communiquer et de transmettre les données collectées par les capteurs à la station de base. L'élection des cluster-heads se fait périodiquement selon leur poids qui est fonction de leur capacité à supporter cette tâche. Le poids d'un nœud est calculé grâce aux métriques suivantes : k-densité, énergie restante et mobilité. Nous avons impliqué ces facteurs dans le calcul des poids des capteurs afin de générer des clusters stables.

Au niveau de la couverture de zone, nous avons proposé un algorithme pour assurer la couverture totale de la zone d'intérêt à des degrés différents. Ce protocole se base sur le même concept que CSOS mais il implique plus de capteurs actifs pour la couverture de zone. En effet, le nombre de capteurs actifs augmente en fonction du degré de couverture de zone appelé k-couverture ou couverture multiple.

Pour tester les performances de nos contributions, nous avons réalisé plusieurs simulations et nous avons comparé les résultats obtenus relativement aux résultats d'autres protocoles. Afin de tester notre protocole de couverture de zone, nous avons développé une application de surveillance dans un parc d'une maison de retraite pour les personnes dépendantes.

Mots clés : Auto-organisation, Diffusion, k-couverture, Ordonnancement d'activité, RdC.