N° D'ORDRE : 05/2013-M / INF

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie « Houari Boumediene »

Faculté d'Electronique et d'Informatique

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTERE

En : INFORMATIQUE

Spécialité : Informatique Mobile

Par : **LALAMA Amor**

Sujet

L'EXCLUSION MUTUELLE DANS LES RESEAUX DE CAPTEURS ET ACTIONNEURS

Soutenu publiquement le **13 Mars 2013,** devant le jury composé de :

Mr. BENCHAIBA Mahfoud	Maitre de conférence/A à l'USTHB	Président
Mr. DERHAB Abdelouahid	Maitre de recherche/A au CERIST	Directeur de Mémoire
Mme. MOUSSAOUI Samira	Maitre de conférence/A à l'USTHB	Examinateur
Mr. ZAFOUNE Youcef	Maitre de conférence/A à l'USTHB	Examinateur

Remerciements

Je remercie tout d'abord ALLAH, le tout puissant, pour nous avoir donné la santé et le courage pour finir ce modeste travail.

Je tiens à remercier mon directeur de thèse Mr DERHAB Abdelouahid pour la confiance qu'il m'a témoignée en ma proposant ce sujet. J'aimerais lui adresser mes vifs remerciements pour sa disponibilité durant toute la période du projet, ses encouragements, son aide généreuse et surtout sa patience. Les discussions scientifiques qu'il a su générer, ses remarques et ses suggestions m'ont permis de finaliser ce travail. Je souhaite lui transmettre ma reconnaissance et ma plus profonde gratitude.

Je remercie les honorables membres de jury qui ont pris la peine de lire et d'évaluer ce mémoire.

Je tien à remercie aussi l'ensemble des enseignants de l'USTHB, sans exception, ainsi tous les employés qui ont rendu plus confortable notre formation au sein de l'université.

Je remercie toute l'équipe de laboratoire IA de l'EMP ainsi tous les membres du centre informatique de l'ENPEI.

Un grand merci à tous les membres de ma famille et spécialement mes parents qui m'ont soutenu au long de mes études

Nos remerciements vont aussi à tous ceux et celles qui ont participé de près ou de loin à l'élaboration du présent travail, à tous nos amis et collègues pour leur soutien moral tout au long de la préparation de ce travail, et qui ont fait en sorte, par leur amour, leur affection et leur soutien, que je puisse avoir les meilleures conditions possibles pour continuer mes études et aller de l'avant. Qu'ils trouvent ici ma gratitude et mon amour pour eux.

Je remercie chaleureusement ma femme et mes enfants pour leur confiance qu'ils m'accorder, leur amour, leur soutien, encouragement et leurs patiences.

Dédicace

A mes parents ;
A ma femme ;
A mes enfants ;
A toute la famille ;
A tous ceux qui me sont chers.

Sommaire

Liste des figures

Liste des algorithmes

Introduction générale

Un réseau de capteurs sans fil, ou Wireless Sensor Network (WSN), est un ensemble d'entités déployées de façon à couvrir un territoire donné. Ils sont capables d'opérer en toute autonomie afin de collecter, traiter, et envoyer des données relatives à leur environnement (par exemple la température, l'humidité, la pression… etc.) vers une station puits. Les capteurs disposent de faibles capacités énergétiques, communiquent entre eux par des liaisons sans fil. Ils sont utilisés dans plusieurs domaines d'application à savoir : les applications militaires, les applications de surveillance de phénomènes environnementaux et urbains, et les applications médicales.

Dans les réseaux de capteurs et actionneurs, ou Wireless Sensor and Actor Networks (WSANs), en plus des nœuds de capteurs, une classe spéciale de nœuds appelés actionneurs est ajoutée au réseau. Ces actionneurs sont capables d'agir sur l'environnement, ils sont plus riches que les capteurs en matière de capacités énergétiques, et peuvent être mobiles dans certains cas. Pour certains types d'applications comme les applications de temps réel, les actionneurs doivent répondre rapidement aux données envoyées par les capteurs. Par exemple, en cas d'incendie, une action devra être commencée le plus tôt possible. En outre, pour générer des actions correctes, les données envoyées des capteurs aux actionneurs doivent rester valides au moment de l'action. Par exemple, quand un capteur détecte la présence d'un intrus dans une région, cet intrus doit être dans la même région quand l'actionneur commence l'action. Les actionneurs sont équipés d'interfaces de communication sans fil, ce qui fait qu'ils sont capables de recevoir les commandes nécessaires soit à partir du nœud SINK (architecture semi-automatisée), soit à partir des capteurs et décident de prendre les mesures appropriées sur l'environnement (architecture automatisée).

Dans ce travail, nous nous intéressons au problème d'exclusion mutuelle dans les réseaux WSANs. Contrairement aux systèmes distribués [7] et aux réseaux mobiles ad-hoc [17, 19,20] où l'exclusion mutuelle consiste à garantir un accès atomique à une ressource critique partagée entre plusieurs processus, un certain nombre d'applications dans les réseaux WSANs poussent les chercheurs à considérer un autre type d'exclusion mutuelle. Si plusieurs actionneurs sont supposés couvrir une région donnée, il est nécessaire que la région d'action de chaque actionneur ne chevauche pas avec une autre afin d'éliminer les actions redondantes (la même action risque d'être exécutée plusieurs fois sur la même région). Cependant, il est parfois impossible d'assurer une couverture totale de la région d'évènement avec des régions d'action disjointes. Par exemple, considérons un système d'arrosage automatique, avec des capteurs d'humidité. Les arroseurs

(actionneurs) sont quand les capteurs détectent que l'humidité d'une région donnée est en dessous d'un certain seuil. Comme la région d'action de chaque actionneur est circulaire, le chevauchement entre ces régions est inévitable. Ici, on préfère l'activation d'un sous-ensemble d'arroseurs qui minimise les chevauchements entre les différentes régions d'action afin de minimiser la quantité d'eau consommée. Le problème d'exclusion mutuelle dans les réseaux WSANs sera défini comme suit : « ***Étant donné un ensemble d'actionneurs dans une région d'évènement, quel est le sous-ensemble minimal d'actionneurs qui couvre toute la région d'évènement tel qu'il y 'a un minimum de chevauchements entre les régions d'action,*** **ou minimise le coût d'action [2] ? »**

Sur la base de cette définition, nous constatons que ce problème consiste à trouver un ensemble d'actionneurs de couverture optimale qui assure une couverture totale de la région d'évènement sous quelques contraintes, chevauchement minimal, consommation d'énergie minimum, un délai de réponse, et donc nous utilisons dans ce travail l'appellation de ***« couverture avec contraintes dans les WSANs»*** au lieu de l'appellation d'exclusion mutuelle.

Les solutions à ce problème adoptent deux approches : une centralisée et l'autre distribuée. Dans l'approche centralisée, le nœud puits a besoin d'avoir une connaissance totale de la topologie du réseau pour calculer l'ensemble d'actionneurs nécessaire pour couvrir la région d'évènement Cette approche présente beaucoup d'inconvénients : (1) la panne du nœud puits a paralyser tout le système, (2) les messages échangés entre le nœud puits et les nœuds de la région d'évènement peuvent traverser un long chemin, ce qui augmente le délai pour déclencher une action ainsi que le nombre de messages générés, (3) le nœud puits doit calculer l'ensemble de couverture pour chaque évènement qui se produit dans le réseau, et (4) le nœud puits doit recalculer l'ensemble de couverture chaque fois que la taille de la région d'évènement change. Dans l'approche distribuée, l'ensemble de couverture est calculé par des actionneurs proches de la zone d'événement, sans le besoin de faire intervenir le SINK. Cette approche règle le problème de longs délais de latence et le nombre important de messages générés. Cependant, elle est moine optimale que l'approche centralisée sur le plan de coût de couverture.

Comme ni l'approche centralisée ni l'approche distribuée ne donnent des performances optimales en termes de toutes les métriques, nous proposons dans ce travail une approche semi-distribuée où on exécute un algorithme centralisé quand l'évènement est créé pour la première fois, et un algorithme distribué quand la taille de l'évènement se réduit.

Ce mémoire est organisé comme suit : dans le premier chapitre, nous donnons des généralités sur les réseaux de capteurs et actionneurs sans fil.

Dans le second chapitre, nous définissons la problématique d'exclusion mutuelle ainsi que les différents types d'exclusion mutuelle déjà définis dans la littérature et nous proposons notre nouvelle terminologie qui est la couverture avec contraintes dans les WSANs. Dans le troisième chapitre, nous présentons un état de l'art sur les travaux existants de couverture avec contraintes.

Dans le quatrième chapitre, nous proposons une architecture semi-distribuée pour résoudre le problème de couverture.

Dans le cinquième chapitre, nous comparons notre architecture avec les deux approches distribuée et centralisée proposées dans [2] et l'approche distribuée proposée dans [3] en termes de coût de communication et temps de réponse. En termes de couverture, nous la comparons avec le solution centralisée de Vedantham et al [2], ainsi que la solution centralisée de Derhab et Zair [3].Et enfin, nous finirons par une conclusion

Chapitre 1

Généralités sur les réseaux de capteurs et d'actionneurs sans fil

I.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 quatre éléments : (i) un sous-système de traitement composé d'un microprocesseur ou d'un microcontrôleur, (ii) un sous-système de communication composé d'une radio de courte portée (iii) un sous-système de captage qui relie le capteur avec le monde physique et mesure des grandeurs physiques comme la température, l'humidité, l'ensoleillement.etc, et (iv) un sous-système d'alimentation en énergie qui loge la pile. Comme la portée de communication des capteurs est limitée, ils coopèrent entre eux pour former une infrastructure de communication appelée réseau de capteurs sans fil ou Wireless Sensor Network (WSN). . Dans ce type de réseau, les capteurs échangent des informations, par exemple, sur l'environnement pour construire une vue globale de la région contrôlée. Les données collectées par les capteurs sont acheminées vers un « point de collecte» , appelé nœud de puits ou Sink. Le besoin d'interaction intelligente avec l'environnement a conduit à l'émergence d'une autre classe de réseaux capable d'effectuer la capture des grandeurs physiques (c.-à-d. surveillance) et agir sur l'environnement (c.-à-d. contrôle), appelée : « *Réseaux de capteurs et d'actionneurs* »., ou Wireless Sensor and actuator network (WSAN).

Ces réseaux peuvent être une partie intégrante des systèmes de surveillance tels que le champ de bataille, le contrôle du microclimat dans les bâtiments, l'énergie nucléaire, la détection des attaques biologiques et chimiques [21], la domotique [22] et surveillance de l'environnement et d'autres domaines.

Un réseau de capteur et actionneurs sans fil est constitué des nœuds capteurs et des nœuds actionneurs connectés entre eux par des liaisons sans fil pour capter et agir respectivement sur l'environnement et effecteur une action précise. Les capteurs sont des appareils à faible coût, faible énergie et une capacité limitée de capture, de calcul et de communication. Les actionneurs sont des nœuds riches équipés des ressources énergétiques, calculs et communications plus élevées.

En outre, le nombre de nœuds de capteurs déployés dans une zone cible peut être de l'ordre de centaines ou des milliers, où un tel déploiement dense n'est généralement pas nécessaire pour les nœuds actionneurs puisque les acteurs ont des capacités et peut agir sur de grandes surfaces.

Dans ce chapitre, nous détaillerons les principaux concepts liés aux réseaux de capteurs et actionneurs sans fil. En premier lieu, nous définirons l'architecture structurelle des WSAN, toute en détaillant les deux composants : capteur et actionneur. Ensuite nous décrivons l'architecture fonctionnelle de ce type de réseaux, puis nous donnons une description de sa plie protocolaire et nous finissons par leurs applications courantes.

I.2 Architecture physique d'un WSANs

Un WSAN est principalement composé des nœuds capteurs et des nœuds actionneurs. Il peut être vu comme un réseau Ad-Hoc composé de dispositifs hétérogènes, les actionneurs et les capteurs, différents en termes de capacités. Comme illustre la figure 1, un WSAN est composé d'un grand nombre de nœuds capteur et de quelques nœuds actionneurs, reliés par des moyens sans fil. Ces dispositifs coopèrent entre eux pour fournir une détection distribuée et exécuter des tâches spécifiques. Les capteurs sont déployés aléatoirement dans le champ d'intérêt à cause la leur densité. Par contre les actionneurs sont déployés d'une manière déterministe d'une façon à couvrir tout le champ d'intérêt. En plus de ça, les actionneurs peuvent être mobiles dans certain application. Un autre nœud appelé ***SINK*** ou station de base est responsable de la collecte de données. Il peut communiquer les données collectées à l'utilisateur final à travers un réseau de communication, éventuellement l'Internet ou un satellite. L'utilisateur peut à son tour utiliser la station de base comme passerelle, afin de transmettre ses requêtes aux nœuds de ce réseau.

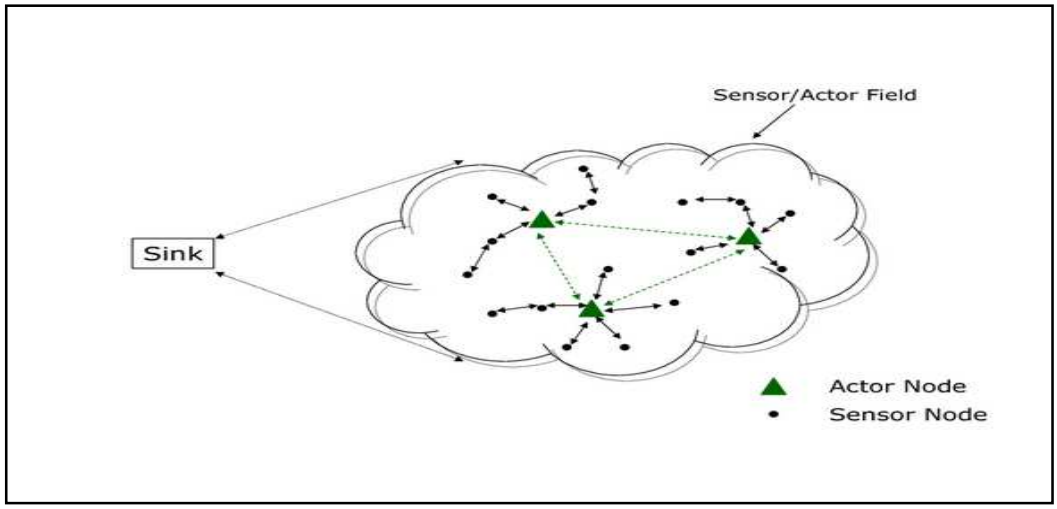


Figure 1: Architecture physique d'un WSANs

I.2.1 Architecture d'un nœud capteur

Un nœud capteur est composé de quatre composants de base [1], comme représentée dans la figure 2, une unité d'acquisition, une unité de traitement, une unité de communication et une source d'énergie.

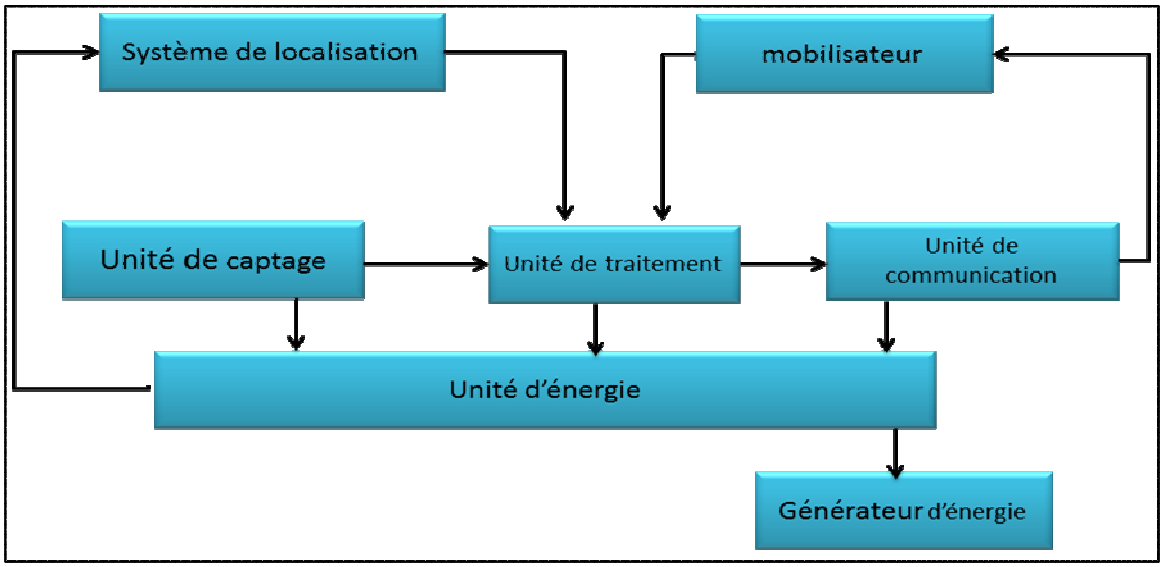


Figure 2 : Les composants d'un capteur

I.2.2 L'unité d'acquisition des données ou de captage

Elle se compose de deux sous unités : unité de captage et un convertisseur analogique numérique(CAN). Le capteur permet de mesurer des informations environnementales : température, humidité, pression, accélération, sons, image, vidéo etc. Puis produit des signaux analogiques qui sont convertis par un convertisseur analogique numérique pour pouvoir être traitées par l'unité de traitement. Il existe deux approches pour construire un capteur, la première approche intègre l'unité de captage sur le capteur par contre dans l'autre approche, l'unité de captage est connecté connectable au capteur à travers des bus d'extension, ce qui permet à l'utilisateur de choisir le type de l'unité de captage selon l'application.

I.2.2.1 L'unité de traitement des données

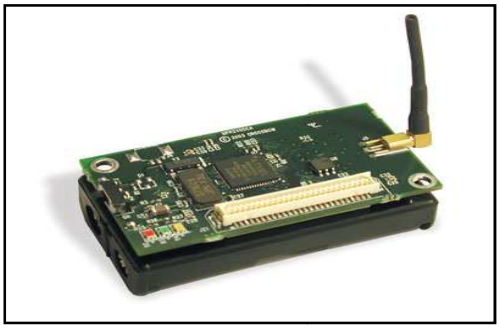
L'unité de traitement comprend un processeur avec une petite unité de stockage, une RAM pour les données et une ROM pour les programmes et souvent une mémoire flash. Cette unité fonctionne à l'aide d'un système d'exploitation spécialement conçu pour les micro-capteurs (TinyOS par exemple). Elle est chargée de gérer des procédures qui permettent à un nœud capteur de collaborer avec les autres nœuds du réseau. Elle peut aussi analyser les données captées pour alléger la tache de la station de base.

I.2.2.2 L'unité de transmission de données

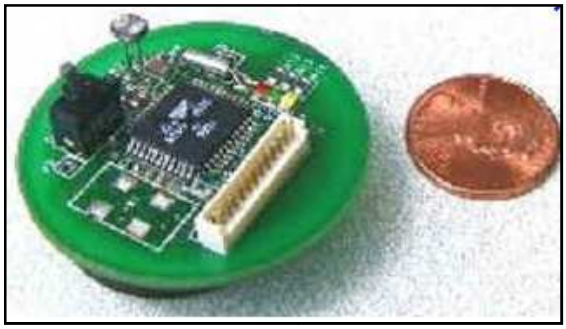
Cette unité est responsable d'effectuer toutes les émissions et réceptions des données sur un medium sans fil. Les composants utilisés pour réaliser la transmission sont des composants classiques, les unités de transmission de type radiofréquence (RF) sont préférables pour les WSN parce que les paquets transportés sont de petites tailles avec un bas débit. Ainsi on retrouve les mêmes problèmes que dans tous les réseaux sans fil : la quantité d'énergie nécessaire à la transmission augmente avec la distance. Pour les réseaux sans fil classiques (LAN, GSM) la consommation d'énergie est de l'ordre de plusieurs centaines de milliwatts alors que pour les réseaux de capteurs, le système de transmission possède une portée de quelques dizaines de mètres. Pour acheminer les données collectées au SINK, le réseau utilise un routage multi sauts.

I.2.2.3 L'unité d'énergie ou de puissance

Les capteurs sont de petits composants alimentés avec une batterie ou avec des piles. Pour qu'un réseau de capteurs reste autonome pendant une durée de quelques mois à quelques années sans intervention humaine, la consommation d'énergie devient le problème fondamental. Celle-ci n'est pas un grand problème pour les réseaux sans fil traditionnel, car on peut toujours recharger les batteries des dispositifs sans fil comme les téléphones portables ou les ordinateurs portables. Mais, dans un RCSF, il est difficile (parfois impossible dans certaines applications) de changer la batterie. Cette unité peut aussi contenir des systèmes de rechargement d'énergie à partir de l'environnement observé telles que les cellules solaires, afin d'étendre la durée de vie totale du réseau. La figure 3 montre quelques capteurs fabriqués par UC Berkeley.



(a) Capteur Mica



(b) Capteur Wec

Figure 3 : Exemple de capteurs

I.2.3 Architecture d'un actionneur

Dans les WSAN, le rôle des capteurs est de collecter les informations à partir du champ d'intérêt, par contre le rôle des actionneurs est de prendre la décision selon les informations collectées par les capteurs et effectuer les actions nécessaires. Les actionneurs sont des nœuds

riches en capacité de calcul et de communication. Leur déploiement est plus déterministe en le comparant avec les capteurs où le nombre de capteurs est plus grand de 10 à 100 fois par rapport aux actionneurs.

En général, en plus de l'unité de transmission, l'unité de traitement et l'unité d'énergie comme dans les capteurs mais plus puissant, un actionneur contient une unité de décision et unité d'action (figure4).

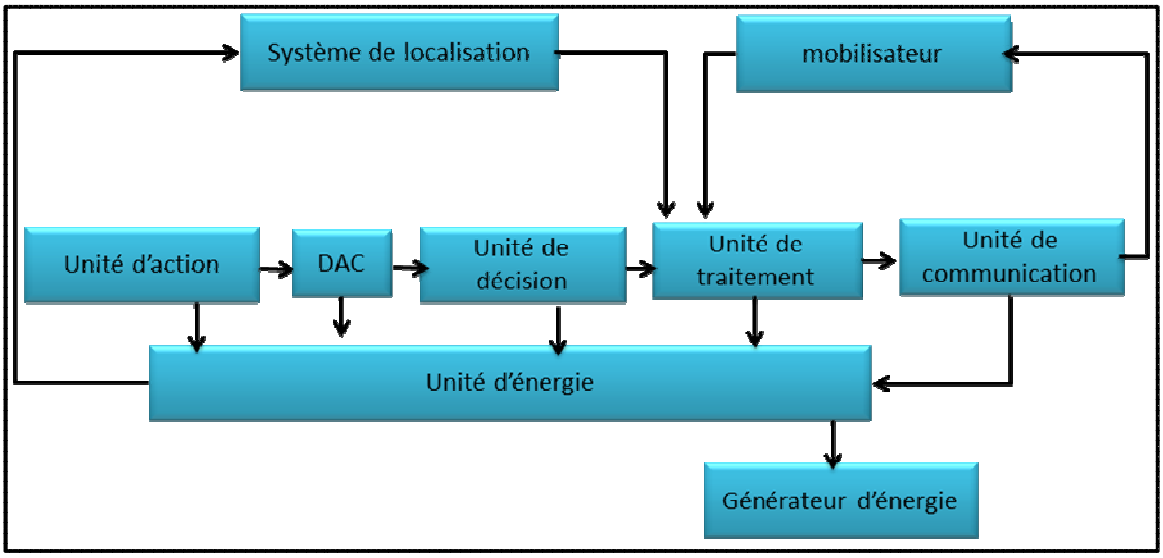


Figure 4 : Les composantes d'un actionneur.

I.2.3.1 Unité de décision

L'unité de décision (ou contrôleur) fonctionne comme une entité qui prend lectures du capteur comme entrée et génère des commandes d'action en sortie. Elle permet de spécifier l'action approprie et leur type à l'unité d'action.

I.2.3.2 Unité d'action

Cette unité contient un convertisseur Numérique/Analogique pour convertir les commande d'action transférées depuis l'unité de décision et le transmet vers l'actionneur.

Dans certaines applications, des nœuds intégré Capteur/Actionneur peuvent remplacer les nœuds actionneurs, l'actionneur. Comme un nœud intégré peut faire les deux taches : la capture, et l'action en même temps. Il contient, en plus des composantes présentées dans la figure 4, l'unité de capture et un convertisseur ADC. Un exemple des applications qui intègre les capteurs avec les actionneurs est les robots [1].

Les actionneurs peuvent être de différents types, par exemple les arroseurs d'incendie (figure 5-a), arroseurs d'irrigation (figure 5-b). Ils peuvent aussi être mobile (figures 5-c et 5-d).

Figure 5 : Exemple d'actionneurs : (a) arroseur d'incendie, (b) arroseur d'irrigation (c) hélicoptère de trace aérien, (d) mule robotique de bataille.

I.3 Architecture fonctionnelle des WSANs

Un WSAN est composé généralement d'un groupe de nœuds de capteurs, qui sont utilisés pour recueillir des informations de l'environnement et des nœuds actionneurs qui sont utilisés pour modifier le comportement de l'environnement. Ces nœuds reliés entre eux par des liaisons sans fils. Les nœuds capteurs capte les informations de l'environnement et rapporte l'état de l'environnement et les actionneurs réagissent sur l'environnement pour changer une grandeur spécifie par les applications.

Quand un capteur détecte un événement, traite les informations captées et, soit les transmet vers les actionneurs pour initier les actions appropries, soit vers une station de base pour les traiter et envoi des commandes d'action vers les actionneurs. Nous appelons la première architecture *une architecture automatisée* et la deuxième architecture une architecture *semi- automatisée* comme illustre la figure 6.

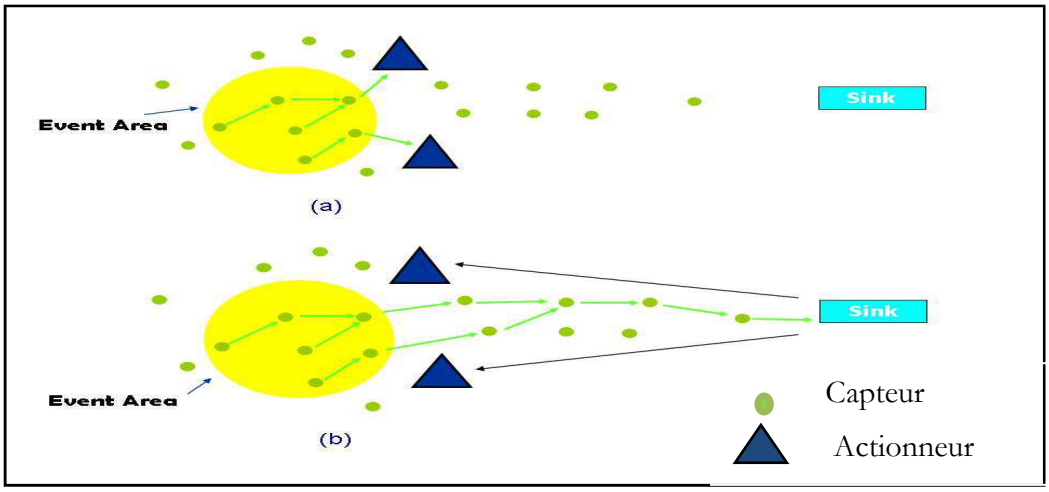


Figure 6 : Les architectures d'un WSAN : (a) automatique, (b) Semi-automatique

I.3.1 Architecture automatique

Dans cette architecture, les données sont récoltées par les capteurs et transmises directement vers les actionneurs, qui coordonnent d'une manière efficace pour exécuter la tache spécifique sans collaboration de la station de base (figure 6-a). Cette architecture est recommandée dans les applications temps réel, où la réaction rapide des actionneurs est une exigence critique. Une autre caractéristique est la consommation optimale et efficace d'énergie dans le réseau, tant que les capteurs ne s'impliquent pas au cours de transmission de données vers les actionneurs, il en résulte une augmentation de la durée de vie du réseau. Cependant, comme résultat de la communication directe entre le capteur et l'actionneur, cette architecture nécessite une implémentation d'un mécanisme efficace de coordination et de communication entre ces appareils.

I.3.2 Architecture semi-automatique

Dans cette architecture, les capteurs transmettent les informations vers la station de base (SINK), qui procède à la collecte des données et détermine quels actionneurs doivent exécuter l'action spécifie, elle se termine par la transmission de la commande d'action vers les actionneurs corresponds (figure 6-b). Cette architecture est similaire à celle des réseaux de capteurs, elle exige seulement une communication capteur-SINK et SINK-actionneur. Une des inconvénients de cette architecture est la centralisation de la décision qui nécessite une durée plus longue pour exécuter une action par les actionneurs, elle n'est pas recommandée dans les applications temps-réel.

I.4 Pile protocolaire

Pour faire face aux défis de coordination capteur/actionneur et actionneur/actionneur, la pile protocolaire des nœuds capteur et actionneurs constituée de trois plans [1] : plans de gestion, plans de communication et plan de coordination comme illustre la figure.7.

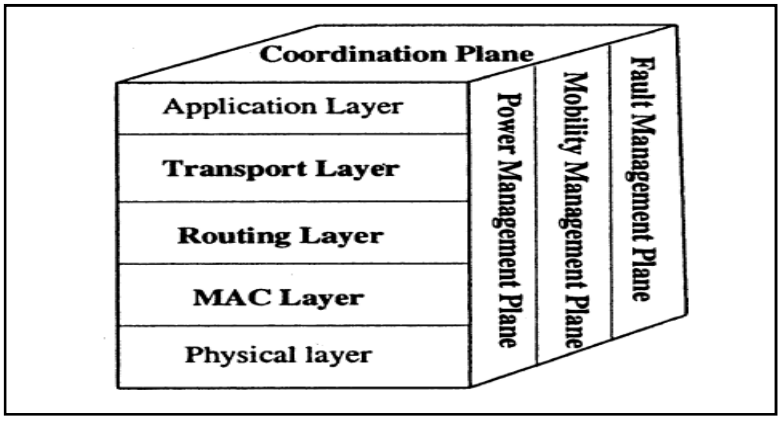


Figure 7 : La pile protocolaire des WSANs

Les données reçues par un nœud au niveau du plan de communication devraient être soumises au plan de coordination qui décide comment le nœud doit agir sur les données.

En outre, le plan de coordination fournit les moyens aux nœuds à être modélisés comme une entité sociale, c.-à-d. en termes des techniques de coordination et de négociation qu'ils possèdent.

Le plan de gestion est responsable de la surveillance et du contrôle d'un nœud capteur/acteur de sorte qu'il fonctionne correctement. Il peut également fournir les informations nécessaires pour la couche de coordination pour prendre des décisions.

I.4.1 Plan de gestion

Les fonctions exercées par la couche de gestion peuvent être classés en trois volets :

I.4.1.1 Gestion d'énergie

Il gère la façon comment un nœud utilise ses ressources énergétique. Par exemple, quand le niveau d'énergie d'un nœud est faible, ce plan informe le plan de coordination de sorte que ce nœud ne participera pas aux activités de capture, routage et action.

I.4.1.2 Gestionnaire de mobilité

Détecte et enregistre les mouvements de nœuds de sorte que la connectivité réseau est toujours maintenue.

I.4.1.3 Gestion d'erreur

Il se réfère à la détection et la résolution des problèmes de nœuds. Par exemple, lorsque la sensibilité de l'unité de détection ou de l'exactitude de l'unité d'actionnement se dégrade, le plan de gestion d'erreurs informe le plan de coordination de cette situation.

I.4.2 Plan de coordination

Ce plan est chargé de déterminer comment un nœud se comporte selon les données reçues à partir du plan de communication et du plan de gestion.

Après la capture d'un événement, les capteurs communiquent leurs lectures entre eux. Au niveau de chaque nœud capteur, ces données échangées sont transférées au plan de coordination pour prendre décisions. De cette façon, les capteurs sont capables de coordonner entre eux dans une tache de capture à haut niveau. En plus, la coordination capteur-capteur peut être exigée pour déterminer les nœuds qui ne transmettent pas les données (dû à l'énergie faible ou le protocole MAC appliqué), pour effectuer le routage multi-saut et l'agrégation des données et le plus important pour sélectionner l'actionneur(s) à qui les données capturées vont être transmises.

L'existence du plan de coordination peut être plus critique pour les actionneurs que pour les capteurs, car les actionneurs ont besoin de collaborer entre eux dont le but est d'effectuer des actions appropriées. Quand un événement se produit, le but commun de tous les actionneurs est de fournir les actions demandées pour cet événement, c'est-à-dire des capacités de négociation et de coordination sophistiquées sont nécessaires dans les WSANs pour assurer un comportement cohérent par l'ensemble des actionneurs. Ces capacités requises d'un actionneur sont définies dans le plan de coordination. Spécifiquement, quelle couche dans la coordination actionneur-actionneur est responsable de prendre les décisions, sur quels actionneurs vont réagir, sur quelle partie de la région de l'événement et est-ce que ces actionneurs agissent en même temps ou séquentiellement et dans quel ordre ?

I.4.3 Le plan de communication

Le plan de communication reçoit les commandes du plan de coordination, selon cette information, il établit les liens entre les nœuds en utilisant les protocoles de communication. Spécifiquement, le plan de communication traite : la construction des canaux physiques, l'accès du nœud au support de transmission (MAC), la sélection des chemins de routage et le transport des paquets d'un nœud à un autre.

Ce plan se compose de cinq couches : couche application, couche transport, couche liaison, couche de routage, couche MAC et couche physique. Dans ce qui va suivre, on va voir les exigences et les défis des couches : transport, MAC et celle du routage, et aussi l'intégration de

cross layer entre ces couches pour les communications capteur-actionneur et actionneur-actionneur.

I.4.3.1 La couche transport

En plus de la fiabilité conventionnelle, les nouveaux protocoles de transport doivent également supporter les exigences de temps-réel dans WSANs. Plusieurs protocoles de la couche de transport ont été développées pour les réseaux ad hoc et les réseaux de capteurs sans fil au cours des dernières années. Cependant, à notre connaissance, il n'existe pas des protocoles de transport qui traitent à la fois la fiabilité et temps-réel pour WSANs à ce jour. Par exemple, lorsque le protocole de transport pour la communication capteur-actionneur détecte une faible fiabilité, le protocole de transport pour la communication actionneur-actionneur règle le trafic entre les actionneurs afin que l'actionneur récepteur aussi puisse informer les actionneurs voisins de cette situation le plutôt possible.

Puisque les communications capteur-acteur et actionneur-actionneur se produisent consécutivement dans les WSANs, un protocole de transport unifié est nécessaire, ce qui fonctionne bien pour les deux cas.

I.4.3.2 La couche de routage

Dans les WSANs, quand les capteurs détectent un événement, il n'y a aucun actionneur spécifique à qui le message va être envoyé. Cette incertitude est due à l'existence de plusieurs actionneurs ce qui cause des défis dans les solutions de routage.

Premièrement, sélectionner un actionneur est un défi pour un capteur source.

Ensuite, les données sources doivent être routées vers l'actionneur sélectionné dans une énergie efficace. Tant que les données sources sont transmises à travers les capteurs de relais vers un nœud actionneur, elles peuvent agrégées ou acheminées pour achever une grande efficacité. En plus de déterminer la route sélectionnée et les données délivrées, le protocole de routage doit supporter les communications en temps réel en considérant les différentes dates limites dues aux différents intervalles de validité.

En outre, le protocole de routage doit aussi considérer le problème de priorité et doit fournir les données avec les moindres de délais bornés pour aboutir à l'actionneur à temps.

Dans les dernières années, un effort considérable de recherches sur les problèmes de routage dans les WSNs [2]. Parmi les protocoles réalisés concernant le routage dans les WSANs, nous citons : DSR, AODV, OLSR [23], qui sont adaptés pour la communication actionneur-actionneur que pour les réseaux ad-hoc, pour la communication capteurs-actionneur nous citons SEAD [24], et SPEED [25].

I.4.3.3 La couche MAC

Pour transmettre effectivement (sans collisions) l'information d'un événement à partir de plusieurs capteurs aux actionneurs, il y a un besoin d'un protocole MAC.

En plus, dans quelques applications (de robots distribuées par exemple), les actionneurs peuvent être mobiles. Quand ils se déplacent, ils quittent les régions de transmission de quelques capteurs et entrent dans les autres régions de capteurs.

Parmi les solutions proposées, nous citons: TRACE [26], PBP, ainsi qu'un protocole développé dans [27] qui est sans collision, il réduit le délai et fournit des garanties de temps réel, et garde l'énergie en éliminant les collisions.

Pour la communication actionneur-actionneur, les protocoles MAC existants et développés pour les réseaux ad-hoc ne peuvent pas être utilisés directement. Ils doivent supporter le trafic en temps réel.

I.4.3.4 Cross-layering

Les protocoles actuels des WSANs et WSNs sont basés largement sur une approche en couches. Or, la sous-efficacité et l'inflexibilité de ce paradigme cause des performances pauvres pour WSANs, due aux contraintes de faible consommation d'énergie et la faible latence. Pour cela, au lieu d'avoir des couches individuelles, on a besoin d'un cross-layering où les couches sont intégrées entre elles (elles communiquent entre elles pour effectuer un intérêt commun).

Dans WSANs, parmi les facteurs principaux qui causent une faible fiabilité est la congestion des réseaux, la couche MAC réagit localement en faisant un retour exponentiel [23], tandis que la couche transport réagit en retardant les transmissions des capteurs. Cependant ces deux couches travaillent indépendamment l'une de l'autre, ce qui cause de l'inefficacité due à la duplication de fonctions. Par l'approche cross-layering chaque protocole partage ses données avec les autres protocoles ce qui évite ces inefficacités. Par exemple, dans les WSANs quand la congestion est grande, premièrement la couche MAC réagit sur cette congestion. Si cette réponse n'est pas suffisante, la couche MAC informe la couche de routage sur cette congestion, cette dernière informe le plan de coordination sur cette situation. Alors le plan de coordination et la couche de routage permettent le reroutage vers un autre nœud actionneur approprié, et si cet actionneur et les routes n'existent pas, le protocole de transport peut être utilisé pour figer le trafic des transmissions.

Un autre exemple de cross-layering dans WSANs est celui de l'optimisation de la taille des paquets transmis du capteur à l'actionneur, dans ce cas les couches de routage, MAC, et physique doivent fonctionner ensemble.

En plus des interactions entre les couches transport, MAC, physique et de routage, dans les WSANs, il est nécessaire d'avoir une interdépendance entre la couche application et ces couches. Bien que le réseau fournisse la meilleure QoS aux applications, cette QoS varie avec le temps autant que les conditions des canaux et la topologie du réseau changent. Pour cela, la couche application doit s'adapter à la QoS offerte.

Les idées citées précédemment sont encore valides pour la communication actionneur-actionneur. Cependant, les actionneurs peuvent être mobiles, les caractéristiques des liaisons et la topologie du réseau changent rapidement. En cas d'une liaison perturbée qui est causée par quelque chose difficile à corriger dans la couche physique, c'est-à-dire une grande mobilité des nœuds, il est mieux pour la couche physique d'interagir avec les couches supérieures [28]. Par exemple, dans WSANs, les actionneurs effectuent des communications unicast au lieu du broadcast pour éviter que les capteurs reçoivent des messages inutiles. Donc, dans l'approche cross-layering, chaque couche de la pile protocolaire ne répond pas seulement aux variations locales, mais aussi répond aux informations venantes des autres couches [28,29].

I.5 Domaines d'applications des WSANs

Les réseaux de capteurs et d'actionneurs sont constitués de plusieurs types de capteurs : sismiques, magnétiques, thermiques, visuelle, infrarouges, acoustiques et radars, qui sont capable de gérer une très grande variété de conditions ambiantes qui inclus les suivants : température, humidité, mouvement de véhicule, condition d'éclairage, la pression, le bruit, la présence ou l'absence de quelques objets, la pression mécanique sur les objets attachés et les caractéristiques courantes tel que : la vitesse, la direction, et la taille d'un objet. Les nœuds capteurs peuvent être utilisés pour la capture continue, la détection d'évènement, l'identification d'évènement, et le contrôle locale des actionneurs. Le concept de micro-capture et la connexion sans fil de ces nœuds promettent de nouveaux domaines d'applications. Nous classifions les applications en : militaire, environnement, santé, maison et autres applications commerciales. Il est possible d'enrichir cette classification avec d'autres catégories comme : l'exploration d'espace, traitement chimique, les opérations de secours en cas de catastrophes [21].

I.5.1 Applications militaires

Comme dans le cas de plusieurs technologies, le domaine militaire a été un moteur initial pour le développement des réseaux de capteurs et d'actionneurs. Le déploiement rapide, le coût réduit, l'auto-organisation et la tolérance aux pannes des WSANs sont des caractéristiques qui rendent ce type de réseaux un outil appréciable dans un tel domaine. Certaines de ces applications peuvent s'automatiser avec l'utilisation d'actionneurs. Par exemple, la détection des mines qui

peut être effectuée par des actionneurs, et ainsi éviter les pertes humaines causées par ces mines. Ces mêmes mines anti-personnel peuvent être jugées trop dangereuses et moins efficaces dans un avenir proche, et pourraient être remplacées par un réseau de capteurs et d'actionneurs capables non seulement de détecter l'ennemi et de le capturer, mais aussi, de le tracer et de le poursuivre.

I.5.2 Applications à la sécurité

Les structures d'avions, navires, automobiles, métros,… etc. pourraient être suivies en temps réel par les WSANs, de même que les réseaux de circulation ou de distribution de l'énergie. Les altérations de structure d'un bâtiment, d'une route, d'un quai, d'une voie ferrée, d'un pont ou d'un barrage hydroélectrique (suite à un séisme ou au vieillissement) pourraient être détectées par des capteurs préalablement intégrés dans les murs ou dans le béton, sans alimentation électrique ni connexions filaires.

Certains capteurs ne s'activant que périodiquement peuvent fonctionner durant des années, voire des décennies. Un WSAN de mouvements peut constituer un système d'alarme distribué qui servira à détecter les intrusions sur un large secteur. Déconnecter le système ne serait plus aussi simple, puisqu'il n'existe pas de point critique.

La surveillance de routes ou voies ferrées pour prévenir des accidents avec des animaux (roadkill) ou des êtres humains ou entre plusieurs véhicules est une des applications envisagées des WSANs. Selon leurs promoteurs, ces réseaux pourraient diminuer certaines failles de systèmes de sécurité et mécanismes de sécurisation, tout en diminuant leur coût.

D'autres craignent aussi des dérives sécuritaires ou totalitaires si l'usage de ces réseaux n'est pas assujetti à des garanties éthiques sérieuses.

I.5.3 Applications environnementales

Le contrôle environnemental représente une catégorie très importante. On peut retenir deux sous-classes d'application :

I.5.3.1 Contrôle environnemental d'intérieur

Un bon nombre de micro-capteurs peut trouver sa place dans le contrôle de l'environnement d'intérieur tel que les capteurs : de températures, de lumières, de pollution et d'humidité. Ces derniers peuvent servir à informer des actionneurs comme l'extincteur d'incendie ou le contrôleur de la climatisation. L'application des WSANs peut être très bénéfique surtout si on sait qu'une grande partie de l'énergie est gaspillée, faute d'éclairer ou de climatiser un endroit vide ou avec une fenêtre ouverte. Des études ont montré que le contrôle de l'éclairage et de la climatisation dans les grands buildings peut diminuer la consommation d'énergie par un facteur

de deux quatrillion de BTU et économisera ainsi 55 milliards de dollars par an, mais aussi diminuer l'émission de gaz carbonique dans l'air par 35 million de tonnes.

Les systèmes de détection d'incendie sont courants et même obligatoires dans les grands buildings. L'application des WSANs peut non seulement coopérer pour éteindre un incendie plus rapidement et plus efficacement, mais aussi coordonner l'évacuation des occupants pris au piège en trouvant le meilleur chemin non obstrué par l'incendie ou le tremblement de terre et ainsi sauver leurs vies.

D'autre part, les applications de sécurité des WSANs qui peuvent détecter une fuite de gaz ou une intrusion, l'actionneur peut couper l'arrivée du gaz et ventiler la pièce, ou dans l'autre cas fermer les sortie et éjecter un gaz tranquillisant.

I.5.3.2 Applications à l'agriculture

Le déploiement d'un WSANs dans l'agriculture peut permettre d'augmenter la production et améliorer la qualité des produits. La capture des informations sur l'humidité, la pluie, la température de l'air et du sol, qui peuvent être transmises à des actionneurs qui réagissent pour assurer des conditions idéales pour le type de produit en exploitation.

Les WSANs peuvent aussi se révéler très économiques en gérant les systèmes d'irrigation, surtout que dans le domaine de l'agriculture, l'eau est une ressource très importante.

I.5.4 Applications commerciales

Des nœuds capteurs pourraient améliorer le processus de stockage et de livraison (pour garantir la chaine du froid en particulier). Le réseau ainsi formé, pourra être utilisé pour connaître la position, l'état et la direction d'un paquet ou d'une cargaison. Un client attendant un paquet peut alors avoir un avis de livraison en temps réel et connaître la position du paquet.

Des entreprises manufacturières, via des WSANs pourraient suivre le procédé de production à partir des matières premières jusqu'au produit final livré. Grâce aux WSANs, les entreprises pourraient offrir une meilleure qualité de service tout en réduisant leurs coûts. Les produits enfin de vie pourraient être mieux démontés et recyclés ou réutilisés si les micro-capteurs en garantissent le bon état.

Dans les immeubles, le système domotique de chauffage et climatisation, d'éclairage ou de distribution d'eau pourrait optimiser son efficience grâce à des micro-capteurs présents dans des tuiles aux plancher en passant par les murs, huisseries et meubles. Les systèmes ne fonctionneraient que là où il faut, quand il faut et à la juste mesure.

Utilisée à grande échelle, une telle application permettrait de réduire la demande mondiale en énergie et indirectement les émissions de gaz à effet de serre.

Rien qu'aux États-Unis, cette économie est estimée à 55 milliards de dollars par an, avec une diminution de 35 millions de tonnes des émissions de carbone dans l'air. Le monde économique pourrait ainsi diminuer ses impacts environnementaux sur le climat.

I.5.5 Applications médicales et vétérinaire

La médecine peut aussi profiter de l'application des WSANs. Les capteurs peuvent capter des paramètres tel le rythme cardiaque et le taux de sucre dans le sang et transmettre ces informations à des actionneurs qui interviendront avant que la situation ne devienne critique.

La surveillance des enfants est aussi une application possible où les capteurs peuvent s'embarquer dans des jouets, ce qui permettra de surveiller le comportement des enfants, prévenir et réagir en conséquence.

I.5.6 Applications de robotique

Beaucoup d'applications de robotique utilisent les WSANs, c'est un domaine d'application naturel où les robots eux-mêmes peuvent être des actionneurs. La robotique représente une réelle automatisation des WSANs. Nous pouvons imaginer toutes sortes de robots capables de réagir dans l'environnement en se basant sur des informations captées par des capteurs distribués ou même embarqués sur le robot lui-même.

L'utilisation des WSANs poursuit un vieux rêve de l'humanité, un environnement intelligent qui réagit automatiquement aux phénomènes physiques observés. Cette technologie répond à un enjeu économique très important, d'abord par son coût de déploiement très faible et son gain très important en énergie et autres ressources critiques.

I.6 Conclusion

Les réseaux de capteurs et actionneurs sans fil présentent un intérêt considérable et une nouvelle étape dans l'évolution des technologies de l'information et de la communication. Cette nouvelle technologie suscite un intérêt croissant vu la diversité de ces applications : militaire, contrôle environnemental, support logistique, santé, …etc.

Dans ce premier chapitre, nous avons présenté les réseaux de capteurs et actionneurs sans fil, WSANs, leurs architectures physique ainsi l'architecture fonctionnelle, la pile protocolaire et leurs diverses applications.

Dans les WSANs, deux architectures de communication sont possibles : semi-automatisée et automatisée. Chaque architecture présente des avantages et des inconvénients. La rapidité, l'exactitude, la minimisation des ressources, et d'autres facteurs peuvent influencer sur le choix d'une architecture sur l'autre.

La migration de ce nouveau type de réseau dans la vie courante a introduit de nouveaux défis qui doivent être traités, parmi ces défis, on s'intéresse dans notre mémoire au problème de minimisation de ressources consommées par les actionneurs ; ce problème sera traité sous le nom de couverture avec contraintes dans les WSANs.

Chapitre 2

L'exclusion mutuelle dans les réseaux de capteurs et actionneurs sans fil (WSAN)

II.1 Introduction

Dans le but d'éliminer l'intervention de l'être humain dans certaines applications critiques du réseau de capteurs sans fil comme par exemple l'extinction de feu (anti-incendie), des nœuds actionneurs, pour réagir sur l'environnement, sont introduits dans ce type de réseaux. Cette introduction a permis l'apparition des réseaux de capteurs et actionneurs sans fil (WSANs). Avec cette apparition, des défis et des axes de recherche sont apparus. Parmi ces défis nous traitons dans ce travail le problème d'utilisation rationnelle des ressources de ces actionneurs. Ce défi est défini comme suit : « quel est le sous-ensemble optimal d'actionneurs qui couvrent entièrement la région d'événement et produisent un chevauchement optimale entre les actionneurs dans la région d'action?». Pour éviter ces chevauchements, il doit avoir des champs d'action des actionneurs ***mutuellement exclusifs***. Dans certaines applications, le champ d'action des actionneurs est de la forme d'un cercle, donc il est impossible d'éviter les chevauchements entre les actionneurs lorsque deux actionneurs ou plus sont activés pour couvrir la totalité d'une région d'évènement. Ces chevauchements vont produire des consommations supplémentaires des ressources et des effets indésirables sur la région d'évènement. Par exemple, dans une application d'arrosage automatique dans le domaine agriculture, quand les capteurs détectent un taux d'humidité inférieur au seuil défini par l'application, les actionneurs doivent être activés. Dans ce cas, il est préférable d'activer un sous-ensemble optimal des arroseurs pour couvrir toute la région d'évènement de tel sorte que les ressources consommées (l'eau) soit minimales et pas des effets indésirables sur la région d'évènement (salinité du sol par exemple) avec une énergie réduite.

Les solutions de l'exclusion mutuelle dans ce stade doit assurer les points suivants : (i) une couverture totale de la région d'évènement par des actionneurs mutuellement exclusifs, (ii) une consommation minimale des ressources par les actionneurs, et (iii) une consommation minimale d'énergie par les deux types des nœuds du réseau (capteurs et actionneurs), elle dépend du nombre de messages circulant dans le réseau pour assurer l'exclusion mutuelle.

Dans ce chapitre, nous allons définir le problème d'exclusion mutuelle dans un premier lieu. Dans un deuxième lieu nous décrivons une classification des algorithmes d'exclusion

mutuelle selon le type de réseau toute en se concentrant sur l'exclusion mutuelle dans les réseaux de capteurs et actionneurs sans fils(WSANs)

II.2 Le problème d'exclusion mutuelle

Le problème d'allocation en exclusivité d'une seule ressource (Exclusion Mutuelle) est un paradigme des problèmes de compétition dans les systèmes. Il s'agit d'un problème de compétition dont l'énoncé est très simple : une entité, appelée ressource non partageable ou critique, ne peut être octroyée à un instant donné qu'à un seul processus parmi N processus du système. Autrement dit, dans un tel problème, c'est l'aspect de concurrence qui domine lorsque plusieurs processus tentent en même temps d'accéder à une même ressource.

En effet, l'utilisation de la ressource critique simultanément par un ensemble de processus peut créer une situation d'incohérence.

Dans les WSANs, le problème de l'exclusion mutuelle se réfère à fournir des régions d'action mutuellement exclusives pour couvrir une région d'évènement dont le but de minimiser les chevauchements entres les régions d'action des actionneurs activés afin d'optimiser les ressources consommées.

II.3 Classification des algorithmes d'exclusion mutuelle

Selon les types de réseaux, nous pouvons classifier les algorithmes d'exclusion mutuelle en quatre types : l'exclusion mutuelle dans les réseaux statiques, l'exclusion mutuelle dans les réseaux cellulaires, l'exclusion mutuelle dans les réseaux mobiles Ad-hoc et l'exclusion mutuelle dans les WSANs.

II.3.1 L'exclusion mutuelle dans les réseaux statique

Dans ce type de réseaux, où l'infrastructure est fixe et stable, les algorithmes d'exclusion mutuelle sont divisés en quatre (04) catégories : basés sur le principe de consensus (permission), basés sur des arbitres, basés sur des jetons (diffusion) et les algorithmes hybrides.

Les algorithmes basés sur les consensus sont fondus sur la notion de permission qui consiste à accorder à un processus demandeur une réponse favorable d'accès à la section critique par tous les processus du système. Dans cette catégorie, on trouve des algorithmes statiques et des algorithmes dynamiques. Parmi les algorithmes statiques, on trouve l'algorithme de Lamport [7] qui nécessite $3\times(n-1)$ messages, et Ricart et Agrawala [8] qui est une amélioration de Lamport, il nécessite $2\times(n-1)$. Dans les algorithmes dynamiques, on trouve Carvalho et Roucairol [9], qui nécessite de (0 à $2*(N-1)$). Les algorithmes basés sur des arbitres sont basés sur une méthode dite des plans projectifs finis. Ils sont dits à consensus partiel ou à permission d'arbitres : ceci

signifie qu'un processus ayant reçu plusieurs demandes, n'accorde son autorisation qu'à l'une d'entre elles à un moment donné ; les autres sont mises en attente. Parmi ces algorithme nous trouvons celle de Maekawa[10]. Ce dernier nécessite $3\times(\sqrt{n}-1)$.

Les algorithmes basés sur le jeton sont fondées sur le mécanisme du jeton. Ce mécanisme, simple dans son principe et très efficace, est utilisé notamment dans de nombreux problèmes inhérents au domaine des systèmes distribués et à celui des réseaux locaux de communication.

Dans le contexte de l'exclusion mutuelle, le jeton est chargé de contrôler l'accès d'un processus parmi N à une ressource critique. En outre, l'unicité du jeton assure d'emblée l'exclusion mutuelle. Parmi les algorithmes de cette catégorie, nous citons celle de Suzuki et Kasami[11] avec *n* messages pour entrer à la section critique.

La dernière catégorie dite hybride, quant à elle, dérive des deux approches précédentes où les processus sont structurés sous forme de groupes ayant des processus en commun. Parmi ces algorithmes, on trouve l'algorithme de Chang et al[12]. Il existe aussi autres algorithmes qui utilisent des structures logiques, comme par exemple celle de Raymond [13]. Ce dernier utilise (log(N) message pour entrer dans la section critique.

II.3.2 L'exclusion mutuelle dans les réseaux cellulaires

Badrinath [14] présente un algorithme d'exclusion mutuelle qui s'adapte à un environnement composé de stations de base et des unités mobiles.

L'idée de cet algorithme est de faire un découplage entre la mobilité des sites et la construction des algorithmes pour des environnements mobiles. Pour cela, les algorithmes distribués sont structurés de manière à ce que la majeure partie des communications et des calculs induit par l'algorithme soit prise en charge par le réseau statique. Il suggère que la structure de données qui englobe l'état de l'algorithme réside dans les sites fixes, alors que le rôle des unités mobiles se résume à initier l'exécution de l'algorithme et à recevoir le résultat de l'exécution.

Une autre solution est présentée par Singhal [15] où l'exécution de l'algorithme se fait seulement entre les nœuds demandant l'accès à la section critique.

II.3.3 L'exclusion mutuelle dans les réseaux Mobile Ad-hoc

Le problème de l'exclusion mutuelle dans les réseaux Ad Hoc est plus compliqué parce que ces derniers n'utilisent aucune infrastructure fixe, d'où la nécessité de concevoir des algorithmes qui s'adaptent à ce nouvel environnement caractérisé par ses fréquentes déconnexions, changement de topologie et de partitionnement, et qui prend en compte les caractéristiques physiques des unités mobiles (vitesse CPU, bande passante limitée, espace de

stockage, source d'énergie limitée). Ils doivent être aussi efficaces, performants et tolérants aux pannes.

Parmi les travaux d'exclusion mutuelle dans les réseaux Ad-hoc, nous citons le travail de WALTER et al [16]. Ces derniers ont proposé un algorithme d'exclusion mutuelle qui induit un graphe direct acyclique sur le réseau. Ce graphe modifie dynamiquement la structure logique pour s'adapter au changement physique de topologie dans les réseaux ad-hoc. Autres travaux dans la littérature dans les réseaux ad-hoc ont été effectués et décrits dans [17, 18, 19, 20].

II.3.4 L'exclusion mutuelle dans WSANs

La définition classique de l'exclusion mutuelle classique telle qu'elle était donnée dans la littérature, où la nécessité d'un accès unique à la section critique par les processus utilisant les mêmes ressources de cette section, ne s'adapte pas au problème de couverture dans les WSANs. Dans ce problème, et si nous appliquons la même définition, l'exclusion mutuelle doit assurer que chaque point de la région d'évènement soit couvert par un et un seul actionneur à la fois. Ce cas ne peut être vérifié dans les WSANs car les champs d'actions des actionneurs sont circulaires, donc le chevauchement entre les actionneurs est inévitable. Par conséquent des effets indésirables sont produits. Selon la nature de l'application, l'action sur une région d'évènement peut avoir plusieurs conséquences, par exemple: (i) une utilisation inefficace des ressources des actionneurs, par exemple : le gaspillage des produit utilisés dans les systèmes d'extinction de feu (eau, gaz inerte...) dans une application anti-incendie, (ii) une sur-utilisation de l'eau peut influer sur les plantes et cause leur mort et augmente aussi la salinité du sol dans une application d'irrigation automatique, (iii) des opérations incorrectes : par exemple dans une application automatique d'alarme anti-incendie, où il y'a une signature unique relative à la fréquence et la tonalité avec laquelle l'alarme se déclenche. Dans ce cas, lorsqu'un feu est détecté, un sous-ensemble minimum des vibreurs non-chevauchés doit être activés, de tel sorte que la signature ne soit pas brouillés et le feu est détecté et (iii) situations catastrophiques, par exemple dans une application automatique anti-intrusions, où on utilise un gaz poison pour paralyser l'intrus, dans ce cas, une dose rend l'intrus invalide mais deux dose peut le tuer.

Pour remédier à ces effets indésirables et utiliser les ressources de réseau d'une manière optimale, Vedantham et al [2] ont défini l'exclusion mutuelle dans les WSANs comme suit : ***Étant donné un ensemble d'actionneurs dans une région d'événement, quel est le sous-ensemble minimum d'actionneurs qui couvre toute la région d'événement de tel sorte qu'ils existent un minimum de chevauchements entre les régions d'action*** *?*

Dans la définition précédente de l'exclusion mutuelle, la définition de chevauchement minimal dans la région d'action peut avoir différentes significations soumises aux exigences d'application.

La figure 8 décrit un ensemble consistant de notations utilisées par Vedantham et al [2], tandis que la figure 9 illustre les différentes régions identifiées dans la notation picturale.

Variables	Description
R	La région d'évènement
$a_1 \ldots a_n$	Ensemble d'actionneurs dans la région *R*
M	Ensemble minimal d'actionneurs pour couvrir *R*
R_M	Région couverte par l'ensemble *M* d'actionneurs
A_i	La région couverte par a_i dans *R*
r_i	La nouvelle région couverte par chaque actionneur a_i
o_i	Chevauchement entre a_i et les chevauchements existants dans R_M
n_i	Le chevauchement entre a_i et R_M
VM_i	La fonction objective de l'actionneur a_i

Figure 8: Notations d'exclusion mutuelle utilisées par Vedantham et al.

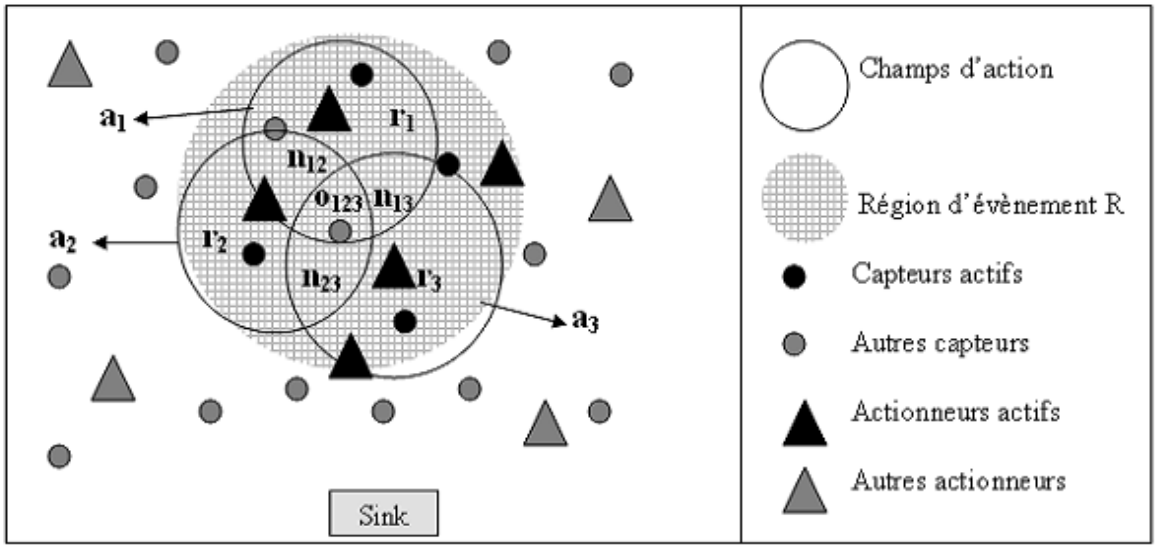


Figure.9 : Les différentes régions basées sur la notation de la figure 8.

Étant donné ces définitions, nous identifions les différents types d'exclusion mutuelle dans WSANs en se basant sur la précision de la sémantique d'exclusion mutuelle et l'intensité de l'action souhaitée.

Vedantham et al. [2] présentent quatre types d'exclusion mutuelle dans les WSANs :

II.3.4.1 Exclusion mutuelle basée ressources critiques

Dans ce type, l'objectif est de maximiser les régions d'actions non-chevauchées pour chaque actionneur dans la région d'évènement, dont le but d'utiliser un minimum de ressources des actionneurs. Par conséquent, dans cette définition, maximiser la région non chevauchée, implique définir le minimum de chevauchement dans la région d'action. Par exemple, dans une application d'extinction de feu, où les extincteurs automatique servent comme actionneurs pour éteindre le feu. Dans le cas où la quantité d'eau dans les extincteurs est limitée, il est nécessaire d'assurer qu'il y'a un minimum de gaspillages d'eau tout en éteignant le feu. Il est souhaitable que chaque actionneur sélectionné dans la région d'événement ait des régions d'actions non chevauchées de telle sorte que le nombre d'actionneurs sélectionnés pour couvrir la région d'événement est minimal. En d'autres termes, la nouvelle zone couverte par n'importe quel actionneur, qui est ajouté à l'ensemble de couverture déjà existante, devrait permettre de maximiser la région non-chevauchée, r_i, de cet actionneur.

Vedantham et al [2] ont défini ce type comme suit «Le problème d'exclusion mutuelle basée ressource critique est de déterminer l'ensemble minimal des actionneurs, M, qui maximise la fonction objective globale défini par la somme des fonctions objective individuelles de chaque actionneur dans M », où la fonction objective individuelle VM_i d'un actionneur a_i est donnée par :

$$\boldsymbol{VM_i = r_i}$$;

Prenons la figure 9, $r_1 > r_2 > r_3$, alors nous devons choisir a_1, puis a_2 et enfin a_3. Donc $M = \{a_1, a_2, a_3\}$.

II.3.4.2 Exclusion mutuelle basée type de chevauchement

Dans ce type, les actionneurs sont choisis d'une manière à maximiser les régions non chevauchées dans la région d'évènement, toute en réduisant le coût du nouvel chevauchement avec l'ensemble de couverture existant. Cette définition est applicable quand il y' a un seuil pour le niveau désiré d'action et n'importe quel coût d'action au-dessus de ce seuil est perçu comme indésirable. Prenons l'exemple de l'application automatique anti-intrusions cité précédemment, où les capteurs sont des caméras qui permettent de détecter la présence d'un intrus, et les actionneurs sont des armes toxiques tranquillisantes où une dose rend l'intrus seulement invalide, mais deux doses peuvent le tuer. Dans cette application, il est impératif de minimiser les nouveaux chevauchements car ils traduiront de double dose qui est mortelle. Autrement dit, il est nécessaire de choisir un nouvel actionneur qui assure un minimum de chevauchement avec le chevauchement existant.

Vedantham et al [2] ont défini ce type comme suit : «Le problème d'exclusion mutuelle basée-type de chevauchement est de trouver un ensemble minimal d'actionneurs, M, de telle sorte que chaque actionneur ajouté à la couverture maximise la surface non-chevauchée de cet actionneur et minimise tout chevauchement avec l'ensemble de couverture déjà existant ». En d'autre terme et comme écrit dans le type précédent d'exclusion mutuelle, le but est de trouver un ensemble M qui maximise la fonction objective globale défini par la somme des fonctions objective individuelles des actionneurs. La fonction objective individuelle est donnée par :

$$\boldsymbol{VM_i = r_i - \alpha \times n_i}\,,$$

Où *a* est une constante représente le coût engendré par le nouvel chevauchement dans la région d'évènement. Dans cette formule, pour maximiser VM_i et comme r_i est toujours positif, il faut minimiser $a \times ni$.

Dans la figure 9, avec *a=1, (par exemple),* en premier lieu on peut choisir n'importe quel actionneur sans avoir un chevauchement car aucun actionneur n'existe dans *M*, soit $\boldsymbol{a_1}$ par exemple car il assure plus de couverture par rapport aux deux autres actionneurs, puis on doit choisir l'actionneur a_3 car il génère moins de chevauchement avec a_1 et enfin en ajoute a_2.

II.3.4.3 Exclusion mutuelle basée région de chevauchement :

Dans cette définition, il est non seulement nécessaire de maximiser la région non-chevauchée couverte par chaque actionneur, mais aussi de minimiser la surface de région chevauchée (à la fois ancienne et nouvelle). Cela correspond à un scénario dans lequel toute action qui se produit au-delà du niveau souhaité est inacceptable et le chevauchement net devrait être minimisé, indépendamment de savoir si elle a eu lieu dans la même région ou ailleurs. Prenons l'exemple précédent de l'application d'extinction de feu, dans le cas où les régions d'actions des arroseurs (extincteurs) se chevauchent, les régions où se produisent des chevauchements se traduiront par des inondations. Dans ce scénario, mis à part la maximisation de la région non chevauchée, il est nécessaire que la somme de tous chevauchement soit minimisée indépendamment du fait qu'ils sont localisés ou non.

Vedantham et al [2] ont défini ce type comme suit : « le problème d'exclusion mutuelle basé chevauchement est de trouver un ensemble minimal d'actionneur, M, qui maximise le non-chevauchement et minimise la surface totale de la région chevauchée de l'ensemble de couverture ». En d'autre terme, l'objectif est de trouver un ensemble d'actionneur qui maximise la fonction objective globale. La fonction objective individuelle de chaque actionneur est donnée par :

$$\boldsymbol{VM_i = r_i - \beta \times (n_i + o_i)}$$

Où β est une constante qui représente le coût engendré d'avoir n'importe quel type de chevauchement dans la région d'évènement (n_i, o_i).

II.3.4.4 Exclusion mutuelle basée intensité :

Ce type est le plus générique dans le problème d'exclusion mutuelle dans les WSANs, où les actionneurs ne sont pas choisis seulement sur les nouvelles et les anciens chevauchements, mais aussi sur l'intensité de ces chevauchements. En d'autres termes, tous chevauchements au-dessus d'un seuil est considéré comme indésirable, et le poids de la fonction objective dépend du nombre de fois où les chevauchements se produit pour une région particulière (l'intensité de chevauchement). Prenons l'exemple précédent, lorsque les régions d'actions des actionneurs (extincteurs) se chevauchent. Dans un tel cas, les régions où des chevauchements se produisent seront inondées. Si le chevauchement se produit plusieurs fois, l'inondation sera grave. Dans ce scénario, en dehors de la maximisation de la région non chevauchée, il est nécessaire que la fonction objective globale soit optimale.

Vedantham et al [2] ont défini ce type comme suit : « le problème d'exclusion mutuelle basée intensité est de déterminer l'ensemble minimal d'actionneurs, M, qui maximise le non chevauchement et minimise le chevauchement total dans les régions en se basant sur l'intensité ». En d'autre terme, l'objectif est de trouver un ensemble d'actionneur qui maximise la fonction objective globale. La fonction objective individuelle de chaque actionneur est donnée par :

$$\boldsymbol{VM_i = r_i - \sum_{I_i} \beta_{Ii} \times I_i \times (n_i, o_i)}$$

Où β_{Ii} est le poids qui représente le coût engendré par un chevauchement avec une intensité I_i dans la région de l'évènement.

II.4 Autres type d'événements

Jusqu'à présent, nous avons discuté le problème d'exclusion mutuelle dans le cadre des événements statiques nécessitant un seul tour d'exécution sans prendre en considération les évènements dynamiques. Cependant, pour d'autres types d'applications, nous devrons relever les défis suivants tout en abordant le problème :

II.4.1 Différentes Intensités d'événement

Dans certaines applications, l'intensité de l'évènement n'est pas uniforme dans toute la région d'évènement (figure 10 (a)).Si l'intensité de l'évènement est différente dans deux régions non chevauchées, l'action doit être effectuée afin de refléter l'intensité événement désiré. Dans un tel cas, il est nécessaire d'exécuter l'action plusieurs fois selon l'intensité de l'évènement.

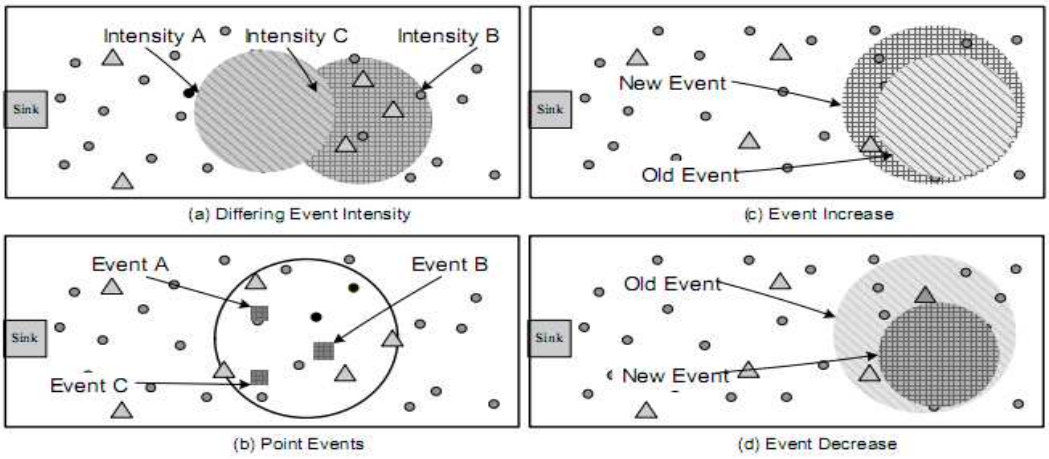


Figure 10: Les variantes d'évènements.

II.4.2 Evènements point/ Multi-point

Jusqu'à présent, nous avons seulement décrit le problème d'exclusion mutuelle pour un événement régional (sous forme d'une région). Le problème de l'évènement point/ multipoint est totalement différent. Dans ce cas la définition se réduit à la minimisation du nombre d'actionneurs répondants à tous les événements ponctuels dans le réseau (voir figure 10 (b)). Par exemple dans le cas d'une application de détection d'intrus, l'événement est l'occurrence des intrus, qui sont des événements ponctuels. Alors, le problème d'exclusion mutuelle se réduit à minimiser les actionneurs qui couvrent tous ces événements ponctuels en se basant sur la fonction objective.

II.4.3 Evènement dynamique

Ce défi se produit lorsqu'il y'a des multiples exécutions de l'actions pour faire face à un évènement dans la région d'évènement. Lorsqu'un évènement se réduit dans sa taille à cause des interventions des actionneurs, il est nécessaire que l'algorithme d'exclusion mutuelle s'exécute seulement dans la région d'évènement en cours (figure 10(d)). Dans le cas où la région d'évènement s'augmente dans sa taille, il est nécessaire que l'exclusion mutuelle soit fournie à la nouvelle région d'évènement comme le montre la figure 10 (c).

II.5 Couverture avec contraintes dans les WSANs

Dans les définitions précédentes de l'exclusion mutuelle dans les WSANs, nous avons vu que l'objectif de l'exclusion mutuelle est d'assurer une couverture totale de la région d'évènement toute en respectant des contraintes. Parmi ces contraintes, nous citons à titre indicatif: (i) l'utilisation optimale des ressources consommées dans le réseau soit par les actionneurs (eau, munition des armes, gaz …) ou bien par les capteurs (énergie), (ii) empêcher des situations

catastrophiques dans quelque types d'applications, (iii) des opérations correctes sur la région d'évènement et (iv) contrainte de temps…etc.

A cause de cette définition, nous proposons de changer le terme « exclusion mutuelle dans les WSANs » par « la couverture avec contraintes dans les WSANs ».

Dans ce travail, nous traitons le problème de couverture dans les WSANs en prenant en considération les contraintes de ressources consommées et du temps.

II.6 Conclusion

Dans ce chapitre, nous avons présenté la définition classique de l'exclusion mutuelle toute en décrivant le problème de l'exclusion mutuelle dans les différents systèmes et réseaux (Réseaux statiques ou filaires, réseaux cellulaires et les réseaux mobile ad-hoc). Nous avons montré que ce problème dans les WSANs est totalement différent de celle des réseaux cités précédemment. Nous avons présenté sa définition dans le cas d'un réseau de capteur et actionneurs sans fil (WSANs). Nous avons vu que la ressource critique dans cette définition est la région d'évènement. Dans cette nouvelle définition, le problème d'exclusion mutuelle devient un problème de couverture avec contraintes (optimisation des ressources, optimisation d'énergie temps de repense….). Nous avons introduit des exemples pratiques pour illustrer le problème d'exclusion mutuelle dans les WSANs. Nous avons aussi présenté les différents types d'exclusion mutuelle dans les WSANs proposés par Vedantham et al [2] selon les besoins des applications. Nous avons parlé, aussi, des variantes des évènements possibles, et enfin nous avons proposé d'utiliser la terminologie de « couverture avec contraintes » au lieu de l'exclusion mutuelle.

Chapitre 3

Couverture avec contraintes dans les WSANs : Etat de l'art

III.1 Introduction

Le problème de couverture avec contraintes dans les WSANs n'a pas reçu une grande attention dans les littératures. Le groupe de recherche de GNAN[1] [2] sont les premiers qui ont attaqué le problème sous la terminologie de l'exclusion mutuelle dans les WSANs. Dans ce qui va suivre, nous allons utiliser la nouvelle terminologie proposée pour ce problème, qui est **« *couverture avec contraintes dans les WSANS* »**

Derhab et al [3] ont proposé une autre alternative pour le calcul de la fonction objective, tout en ajoutant des améliorations dans la solution avec la réduction du champ d'action et la mobilité des actionneurs.

Dans ce chapitre, nous allons présenter les travaux existants qui traitent le problème de couverture avec contraintes dans les WSANs et les travaux liés à ce problème, tout en présentant quelques travaux de couverture dans les WSN.

III.2 Solutions de Vedantham et al

Vedantham et al [2] ont proposé deux approches pour résoudre le problème de couverture avec contraintes dans les WSAN, une approche centralisée et l'autre distribuée.

III.2.1 Approche centralisée

Dans cette approche, il existe un nœud central (SINK) qui s'occupe de la collecte des informations et le calcul de l'ensemble de couverture. Lorsque les capteurs détectent la présence d'un évènement, ils reportent ces informations vers le SINK. Ce dernier, en utilisant sa connaissance sur la topologie du réseau, calcul l'ensemble d'actionneurs de couverture en basant sur le critère de la fonction objective de chaque actionneur selon le type de la couverture avec contraintes (exclusion mutuelle) choisi. Ensuite le SINK transmet des commandes d'activation vers les actionneurs sélectionnés.

III.2.1.1 Mécanisme de l'approche

L'algorithme de cette approche fonctionne par sélection, à chaque étape, de l'actionneur qui possède une fonction objective maximale,défini par le type de couverture (exclusion

[1] Georgia Tech Networks and Mobile Computing Research Group

mutuelle). L'actionneur sélectionné est ajouté à l'ensemble de couverture existant dans cette étape. La fonction objective est mise à jour pour tous les actionneurs qui ont un chevauchement dans la région d'action en se basant sur les nouvelles valeurs de R_m, r_i, n_i et o_i.

Soit *M* l'ensemble d'actionneurs de couverture. Initialement M est vide, l'algorithme commence par inclure un actionneur arbitraire qui se trouve complètement dans la région d'évènement (maximise la fonction objective). À chaque étape, l'algorithme choisit l'actionneur, MAX_ACTOR, qui a la fonction objective maximal, VM_{MAX_ACTOR} (lignes 15-18 de l'algorithme 3.1). L'actionneur est ajouté à l'ensemble d'actionneurs de couverture, M, et la valeur de R_M est mise à jour en ajoutant la région nouvelle couverte par actionneur, MAX_ BENEFIT. En outre, l'ensemble d'actionneurs restant, ω, est mis à jour en retirant cet actionneur (lignes 20-22 de l'algorithme 1). N'importe quel actionneur, a_i, qui se chevauche avec cet actionneur, met à jour sa région non couverte, r_i, la nouvelle région chevauchée, n_i, et l'ancienne région chevauchée, o_i (lignes 23-25 de l'algorithme 1). L'algorithme se terminera quand la région d'événement est contenue dans R_M (ligne 11 de l'algorithme 1).

Input

1*: a_1,a_2,…, a_n* : actionneurs dans le champs d'action du WSAN, S : Sink

2: C : Commande envoyé à la région d'intérêt

3: RC : Région d'intérêt de la commande

4: Ω : Ensemble d'actionneurs dont leurs champs d'action se chevauchent avec RC

5: Ri : Champ d'action de l'actionneur ai

Output

6: L'ensemble minimale d'actionneurs de couverture de RC

7: Calculer l'ensemble minimale d'actionneurs de couverture de la région RC

8: M : L'ensemble d'actionneurs sélectionné comme membre de l'ensemble de couverture

9: RM : Région couverte par M

10: $w = \Omega$

11: ***while*** (RC $\not\subset$ RM)

12: MAX_BENIFIT = 0 (ou – $\alpha \times \pi \times R^2$)

13: **for each** ai $\in$ w

14 VMi = function de bénéfice de a

15: **if** (VMi > MAX_BENIFIT)

16 : MAX_BENIFIT = r_i

17: MAX_ACTOR = i

18: **end if**

19: **end for**

20: M = M $\cup$ MAX_ACTOR

21: $R_M = R_M$ + MAX_BENIFIT

22: $w = w$ – MAX_ACTOR

23: **for each** ($a_i \in$ w) and (($R_i \cap R_{MAX\_ACTOR}$) $\neq$ 0)

24: Update r_i, o_i, n_i

25: end for

26: end while

27: return M

Algorithme1 : algorithme centralisé de Vedantham et al.

III.2.1.2 Exemple d'illustration

Soit un exemple d'un évènement représenté dans la figure 11. Le cercle rouge, raillé représente la région d'évènement et les autres cercles représentent des actionneurs de a_1 à a_4.

Supposant que le type d'exclusion mutuelle est celui décrit dans section III.4.3 du deuxième chapitre). Initialement M est vide, dans ce cas on va choisi celui qui couvre le plus la région d'évènement (c à d celui qui à le plus grand r_i), dans ce cas l'actionneur choisi est a_2. On ajoute a_2 à l'ensemble *M,* donc $M=\{a_2\}$, tous les actionneurs restants se chevauchent avec l'actionneur a_2, dans ce cas on doit calculer la fonction objective VM_i qui est égale à

$\boldsymbol{VM_i = r_i - \beta\times (n_i + o_i),}$ et supposons qu'on donne un poids plus grand pour les chevauchements (c'est à dire on choisit les actionneurs qui ont un minimum de chevauchements avec *M*). Si β=2 par exemple, il est clair que c'est a_4 qui sera ajouté à *M*. $M = \{a_2,a_4\}$. De la même manière on ajoute a_1 et a_3 pour couvrir la région d'évènement.

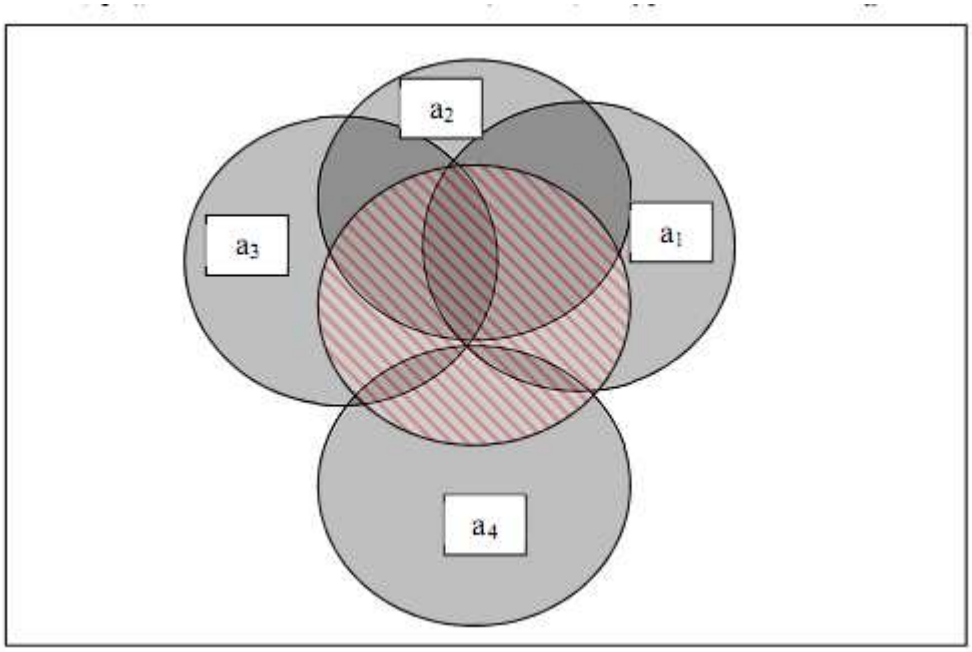


Figure 11: Exemple d'illustration de l'algorithme centralisé de Vedantham.

III.2.1.3 Optimalité de l'approche

Comme décrit dans [13],[32]et[33], le calcul de la couverture optimale pour un problème de couverture d'ensemble dans un graphe de réseau est NP-difficile. Les auteurs dans [32] et [33] ont conclu que le ratio de calcul pour les approches centralisée est directement proportionnel au rayon du réseau et proportionnel logarithmique au nombre de nœud dans le réseau, c.-à-d. $O(r_c \times log(\Omega))$, où r_c représente le rayon de l'évènement et Ω l'ensemble des actionneurs dans la région d'évènement.

III.2.2 Approche distribuée

Vedantham et al [2] ont proposé aussi une approche distribuée et complètement localisée pour résoudre le problème de couverture avec contraintes dans les WSANs. Cette approche basée sur la notion de région de dépendance. Cette région est définie comme suit : Pour un

capteur, la région de dépendance est la région où son rayon est égal à la somme de rayon de capture et le rayon d'action. Par contre pour un actionneur elle est donnée par la région où son rayon est égal à deux fois le rayon d'action.

Une fois les régions de dépendance pour tous les actionneurs dans la région d'évènement sont déterminées, les opérations de base de l'approche distribuée sont :

- La détermination de la fonction objective de chaque actionneur en se basant sur les directives envoyées par les capteurs vers les actionneurs dans ses régions d'évènement.
- L'émulation de la stratégie centrale dans chaque actionneur par un temps d'attente inversé proportionnellement à la fonction objective de cet actionneur. Si la fonction objective d'un tel actionneur est grande, le temps d'attente sera relativement petit et vice versa.
- La mise à jour de la fonction objective (et par conséquent le temps d'attente pour l'exécution) pour tous les actionneurs dans la région de dépendance de l'actionneur qui actionne pour une directive spécifique.

Dans ce qui suit, nous allons présenter les opérations dans les capteurs et les actionneurs.

III.2.2.1 Mécanisme de l'approche

Opérations dans les capteurs :

- Quand un capteur détecte la présence d'un événement, il signale d'abord l'information détectée au SINK. Dans l'approche NB, chaque capteur dans la région d'événement construit aussi un arbre de plus court chemin [10] pour chaque actionneur dans sa région de dépendance et utilise un mécanisme de fiabilité saut-par-saut dans le cas où il y a des pertes. Une fois l'arbre de deux sauts de routage est construit, il émet une directive *(REQUEST())* vers tous les actionneurs de sa région de dépendance. Chaque capteur, Si, insert l'information suivante avec la directive demande *REQUEST(Dir_id, Xsi, Ysi),* où Dir_id indique l'identifiant de la directive basée sur la requête du SINK et Xsi, Ysi les coordonnées du capteur Si. Les actionneurs attendent un temps égale à deux fois le délai de transmission, Td, pour assurer la réception de toute les directives REQUEST() dans sa région de dépendance, avant de commencer une action

Opération dans les actionneurs :

Lorsque un actionneur reçoit une directive REQUEST() depuis un capteur, il détermine d'abord la région additionnelle couverte par le capteur et ajoute cette région à la région d'événement déjà existante, comme indiqué dans les lignes 13-14 de l'algorithme 3.2. En outre, il détermine l'intersection de cette zone avec le champ d'action utilisé comme métrique virtuelle.

Cette métrique virtuelle est utilisée pour déterminer le temps d'attente pour cet actionneur, qui est mise à l'échelle appropriée pour avoir une limite supérieure de δ (lignes 15-16 de l'algorithme 2).

- Si le temps d'attente est inférieur ou égal à zéro, ce qui implique que le bénéfice de cet actionneur est suffisamment élevée pour agir immédiatement, la transmission de NOTIFY() est prévue pour indiquer que cet actionneur va exécuter de la directive rapidement. D'autre part, si la fonction d'attente est supérieur à zéro, l'actionneur reste dans l'état WAIT ().
- Si le message reçu est *NOTIFY()*, l'actionneur vérifie d'abord s' il est déjà prévu d'agir en vérifiant son *FLAG()*.Si le *FLAG()* est à *TRUE*, ce qui indique que l'actionneur courant est également ordonné pour agir. On compare la fonction objective des deux actionneurs, si la fonction objective de l'actionneur courant est plus élevée, ou si leur identifiant est petit avec les deux fonctions objectives similaires, il se met à l'état *TRANSMIT()* pour exécuter la directive (lignes 26-29 de l'algorithme.2). Dans le cas contraire, l'actionneur recalcule leur fonction objective ainsi le temps d'attente conclu pour que l'autre actionneur est ordonné pour agir. Le *FLAG()* est mis à *FALSE*, ce qui indique que l'actionneur n'est plus dans la phase d'exécution, et reste en *WAIT()* si le temps d'attente supérieur à 0.
- Si le temps d'attente atteint zéro, l'actionneur se met en phase de transmission, il va transmettre le message *NOTIFY()* avant l'exécution. L'actionneur va d'abord envoyer le message *NOTIFY ()* pour tous les actionneurs dans sa région de dépendance en utilisant la structure de routage déjà construite et attend pendant un temps correspondant à deux fois le délai de saut, 2×Td. Si dans l'intervalle, un message *NOTIFY()* est reçu, la procédure mentionnée dans les lignes 26-36 de l'algorithme 2 est suivie comme il est expliqué en haut.

Default State:
0 *Wait*(A_i)
Variables:
1 A_i: Actor Node id, $D(i)$: Directive ID,
2 $S_1 \ldots S_k$: Array of Requests from Sensors at A_i,
3 $R_{A_i}(k)$: Region enclosed by requests from k sensors at A_i,
4 $VM(A_i)$: Benefit fn. of A_i, $VM(A_i)_{init}$: Initial Benefit fn.,
5 $WT(A_i)$: Waiting time before execution for A_i,
6 $Ar(S_i)$: Maximum area covered by sensor S_i,
7 $Ar(A_i)$: Maximum area covered by actor A_i,
8 $T_{curr}(A_i)$: Current Time at actor, A_i,
9 $T_{init}(A_i)$: Time at actor, A_i when first request was received,
10 $Flag(A_i)$: Flag at A_i to denote $NOTIFY()$ sent or not,
11 $T_{note}(A_i)$: Time at actor, A_i when $NOTIFY()$ was sent,
Receive(A_i)
12 If $(MSG_{RX} == (REQUEST() or CANCEL())$
13 If $(MSG_{RX} == REQUEST(Dir\_id, X_{S_j}, Y_{S_j})$
14 $R_{A_i}(k+1) = R_{A_i}(k) \bigcup Ar(S_j)$
15 $VM(A_i) = R_{A_i}(k+1) \bigcap Ar(A_i)$
16 $WT(A_i) = \delta \cdot \frac{(Ar(A_i) - VM(A_i))}{(Ar(A_i) - VM(A_i)_{init})} + T_{init}(A_i) - T_{curr}(A_i)$
17 If $WT(A_i) \leq 0$
18 *Transmit*(A_i)
19 If $(MSG_{RX} == CANCEL(Dir\_id, X_{S_k}, Y_{S_k})$
20 $R_{A_i}(k-1) = Ar(S_1) \bigcup Ar(S_2) \bigcup \ldots \bigcup Ar(S_{k-1})$
21 $VM(A_i) = R_{A_i}(k-1) \bigcap Ar(A_i)$
22 If $VM(A_i) > 0$
23 $WT(A_i) = \delta \cdot \frac{(Ar(A_i) - VM(A_i))}{(Ar(A_i) - VM(A_i)_{init})} + T_{init}(A_i) - T_{curr}(A_i)$
24 Else If $VM(A_i) \leq 0$
25 $WT(A_i) = NULL$
26 If $(MSG_{RX} == NOTIFY(Dir\_id, VM(A_j), X_{A_j}, Y_{A_j})$
27 If $(Flag(A_i) == TRUE)$
28 If $(VM(A_i) > VM(A_j)) or ((VM(A_i) == VM(A_j)) and (i < j))$
29 Return to *Transmit*(A_i)
30 Else
31 Update r_i, o_i, n_i
32 Update $VM(A_i)$ based on benefit function
33 $WT(A_i) = \delta \cdot \frac{(Ar(A_i) - VM(A_i))}{(Ar(A_i) - VM(A_i)_{init})} + T_{init}(A_i) - T_{curr}(A_i)$
34 $Flag(A_i) == FALSE$
35 If $WT(A_i) \leq 0$ then *Transmit*(A_i)
36 Return to *Wait*(A_i)
Transmit(A_i)
37 $Flag(A_i) = TRUE$
38 Send $NOTIFY(Dir\_id, VM(A_i), X_{A_i}, Y_{A_i})$ to 2-hop actors
39 $T_0 = T_{curr}(A_i)$
40 Do
41 If (MSG_{RX})
42 *Receive*(A_i)
43 While $!(T_0 == 2 \cdot T_d)$
44 *Execute*(*Dir_id*)
Wait(A_i)
45 $VM(A_i)_{init} = 0$ (or) $-\alpha \cdot Ar(A_i)$ (or) $-\beta \cdot Ar(A_i)$ (or) $-\beta_{I_i} \cdot I_i \cdot Ar(A_i)$
46 Do
47 If (MSG_{RX})
48 *Receive*(A_i)
49 While $!(WT(A_i) == 0)$
50 *Transmit*(A_i)
51 $Flag(A_i) = FALSE$
52 $WT(A_i) = NULL$

Algorithme 2 : Algorithme distribué de Vedantham et al.

Dans ce mode, un actionneur peut mettre à jour sa fonction objective basée sur le message NOTIFY() reçu avant l'expiration du temps d'attente. Ce processus de mise à jour est similaire à celui de l'approche centralisée. Toutefois, contrairement à l'approche centralisé où un seul actionneur est sélectionné à chaque itération, dans l'approche distribuée, il est possible que plusieurs actionneurs sont choisis pour agir dans une itération (ou en même temps) si les actionneurs ne sont pas dans les régions de dépendance de chacun de l'autre.

III.2.3 Comparaison entre l'approche centralisée et l'approche distribuée

L'approche distribuée offre de meilleures performances en termes de coût de communication c'est-à-dire en termes de nombre de messages dans le réseau et temps de réponse à l'évènement à cause des opérations locales entre les différents nœuds du réseau.

Comme chaque actionneur prend ses décision localement et n'a pas une vue globale sur le réseau, l'approche distribuée est moins optimale en termes de coût de couverture (le coût de l'ensemble de couverture).

Par contre, l'approche centralisée présente de meilleures performances en termes de coût de couverture à cause de leur connaissance sur l'état du réseau, mais elle nécessite un nombre élevé de messages dans le réseau et aussi du temps pour répondre à l'évènement.

Les deux approches utilisent un des algorithmes heuristiques qui donnent des résultats approximatifs et pas des résultats exacts. C'est à dire que ces résultats ne sont pas forcement les meilleurs. Prenons l'exemple de la figure 11, les actionneurs choisis par l'algorithme sont quatre (04) actionneurs selon l'ordre : a_2, a_4, a_1 et enfin a_3. Par contre trois (03) actionneurs a_1, a_3 et a_4 sont suffisants pour couvrir la totalité de la région d'évènement ce qui génère moins de chevauchement entre les trois actionneurs dans cette région.

III.3 Solutions de Derhab et Zair

Vedantham et al [2] supposent intuitivement que le fait de maximiser les régions d'action non chevauchées va conduire à minimiser le taux de ressources supplémentaires. Cependant cette supposition est présentée sans preuve. De plus, ils considèrent que le coût de la région chevauchée, couverte par deux actionneurs ou plus, dépend seulement de la taille de sa région. Comme chaque nouveau chevauchement produit une quantité supplémentaire de ressources d'actionneur, cette méthode ne peut pas estimer correctement le coût supplémentaire des ressources des actionneurs

Derhab et al [3] ont proposé une autre méthode pour le calcul de la fonction objective, tout en ajoutant des améliorations dans la solution avec la réduction du champ d'action et la mobilité des actionneurs. Cette méthode est basée sur l'ajout du nombre de chevauchements entres les actionneurs. Ils ont proposé une autre fonction objective à minimiser tout en introduisant un poids au chevauchement, qui est le nombre d'actionneurs couvrant une région donnée comme illustrer dans la figure 12.

III.3.1 Calcul de la fonction objective

Soit les définitions et les notations suivantes :

- Un point dans une région ***R*** est dit *K*-actionneur-couvert(K≥1) s'il se trouve dans la région d'action de *K* actionneurs.
- Une région *A* est dite *K*-actionneur-couverte (K≥1) si chaque point dans *A* est k-actionneur-couvert. Le *K* est appelé degré de couverture de *A*, dénoté par $\delta(A)$.
- AR_i et A_i représentent le rayon d'action et la région d'action respectivement.
- Soit un actionneur a_i qui couvre une région *A* dans *R*, sa taille est *Size(A)*, la quantité de ressources consommées par a_i dans *A* est $Res(A,a_i)= a_i \times Size(A) \times T_i$, où T_i est le

temps nécessaire pour qu'un actionneur déclenche une action et a_i est la quantité de ressources consommées/unité de temps/unité de surface. On ignore T_i et a_i pour simplifier les calculs.

- On définit $S_i=\{q \in P | \delta(q)=i\}$ comme étant l'ensemble de sous-région dans la région d'évènement A dont le degré de couverture est i. Le degré maximal de couverture d'une configuration ζ dénoté par $\Delta(\xi)$ est le plus grand degré de couverture parmi toutes les sous-régions de R. La quantité de ressources consommée par les actionneurs dans la région d'action est donnée par :

$$Res\ (R) = \sum_{a_i \in \Omega} Size\ (A_i \cap R) = \sum_{i=1}^{K} \sum_{a \in S_i} i \times Size(a)$$

$$Res\ (R) = \sum_{i=1}^{\Delta(\xi)} \sum_{a \in S_i} i \times Size(a) = \sum_{a \in S_1} Size\ (a) + \sum_{i=2}^{\Delta(\xi)} \sum_{a \in S_i} (i-1) \times Size(a)$$

$$Res(R) = \sum_{a \in S_1} Size\ (a) +$$

$$\sum_{a \in S_2} Size\ (a) + \sum_{a \in S_2} Size\ (a) +$$

$$\sum_{a \in S_3} Size\ (a) + \sum_{a \in S_3} 2 \times Size\ (a) + \cdots$$

.

$$= Size\ (R) + \sum_{i=2}^{\Delta(\xi)} \sum_{a \in S_i} (i-1) \times Size(a)$$

Nous pouvons remarquer que l'équation est composée de deux parties : (1) Size(R) qui représente la taille réelle de la région d'évènement (la quantité exacte de ressources nécessaires pour la région R) et (2) les ressources supplémentaires dépensées inutilement.

Le problème de couverture dans ce cas devient le suivant : quelles sont les actionneurs situant dans la région d'évènement qui minimisent les ressources consommées supplémentaires. Donc l'objectif est de minimiser la fonction objective suivante :

$$f^M = Min(\sum_{i=2}^{\Delta(\xi^M)} \sum_{a \in S_i} (i-1) \times Size(a))$$

La figure 12 présente un exemple d'une configuration de couverture selon la fonction objective, les nombres 1, 2, 3 et 4 représentent le degré de couverture (i de l'équation précédente).

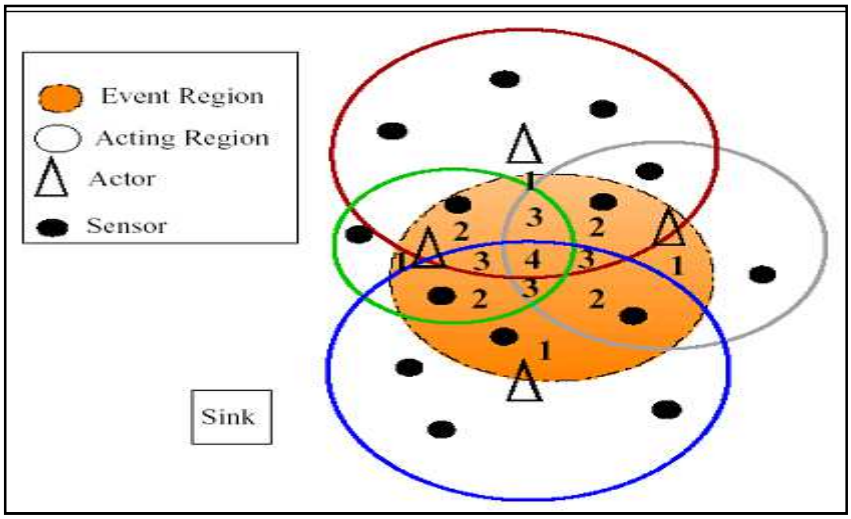


Figure 12 : Exemple de couverture.

III.3.2 Algorithme d'Exclusion Mutuelle Centralisé basée-Ressource (CRMEA)

Lorsque les capteurs détectent la présence d'un nouvel évènement dans le champ d'action, ils envoient un message vers le SINK en l'informant de cet évènement. Selon ces informations reçues, le SINK détermine le rayon et les coordonnées de l'évènement. Ensuite le SINK exécute l'algorithme CRMEA en deux phases : (1) phase initiale pour calculer l'ensemble d'actionneurs de couverture, *M*, qui minimise les ressources supplémentaires f^M (CRMEA) et (2) une phase d'optimisation qui consiste à exécuter la réduction de rayon d'action et/ou la mobilité des actionneurs (CRMEA+ R, CRMEA+M, CRMEA+R+M, CRMEA+M+R).

III.3.2.1 Phase initiale

Dans cette phase, le nœud SINK choisit à chaque étape l'actionneur qui ne génère aucune ressource supplémentaire (lignes 14-16 dans l'algorithme 3.3), ou qui génère un minimum de ressources supplémentaires (lignes 9-12 dans l'algorithme 3.3). Le premier actionneur sélectionné ne génère aucune ressource supplémentaire. L'actionneur sélectionné est ajouté à M (c.-à-d., l'ensemble des actionneurs choisis pour l'instant), et l'algorithme se termine lorsque la région d'évènement R est totalement couverte par M (la ligne 5 dans l'algorithme 3).

Algorithm 1 CRMEA

1: $M = \emptyset$;
2: $R_M = \emptyset$;
3: $\omega = \Omega$;
4: $f^M = 0$;
5: **while** $R \nsubseteq R_M$ **do**
6: $\Delta min = \infty$;
7: **for each** $a_i \in \omega$ **do**
8: $\Delta C = f^{M \cup a_i} - f^M$
9: **if** $\Delta C < \Delta min$ **then**
10: $\Delta min = \Delta C$;
11: $selected = a_i$;
12: **end if**
13: **end for**
14: **if** $\Delta min = 0$ **then**
15: $selected = \{a_i \in \omega | (A_i \cap R)$is the largest$\}$;
16: **end if**
17: $M = M \cup selected$;
18: $R_M = R_M \cup (A_{selected} \cap R)$
19: $\omega = \omega - selected$;
20: **end while**
21: return M;

Algorithme 3: la phase d'initialisation de CRMEA.

III.3.2.2 La phase d'optimisation de rayon d'action

Cette phase est exécutée si les actionneurs sont capables de contrôler leur rayon d'action.. Dans cette phase, le SINK teste pour chaque actionneur de l'ensemble de couverture, *M*, s'il peut réduire le rayon d'action tout en gardant une couverture totale de la région d'évènement *R*. Pour ce faire, le SINK exécute la fonction *ReduceRadius* pour obtenir le nouvel rayon d'action de l'actionneur (ligne 6 de l'algorithme 3.4). Si la réduction du rayon d'action d'un actionneur influe sur la couverture totale, cette fonction retourne une valeur négative (-1), cet actionneur doit être ignoré (ligne 7-8 de l'algorithme 3.4). Le SINK commence par les actionneurs dont la réduction de leur rayon d'action donne une meilleure valeur (fonction objective minimale), et il continue de le faire jusqu'à ce qu'il n'y plus un champ d'action à réduire.

Algorithm 2 CRMEA: Acting range optimization phase

```
1: ω = M; {M is the set returned by Algorithm 1}
2: while ω ≠ ∅ do
3:     selected = ∅;
4:     Δmin = f^M;
5:     for each a_i ∈ ω do
6:         Rnew = ReduceRadius(a_i);
7:         if Rnew = −1 then
8:             ω = ω − a_i;
9:         else
10:            calculate fnew, the new objective function assuming that AR_i =
               Rnew;
11:            if fnew < Δmin then
12:                Δmin = fnew;
13:                selected = a_i;
14:            end if
15:        end if
16:    end for
17:    if selected ≠ ∅ then
18:        ω = ω − selected;
19:        AR_selected = Rnew;
20:        f^M = fnew;
21:    end if
22: end while
```

Algorithme 4: phase d'optimisation de rayon d'action.

III.3.2.3 La phase d'optimisation en utilisant la mobilité des actionneurs

Dans cette phase et si les actionneurs sont capable de se déplacer (être mobile), le SINK teste pour chaque actionneur dans l'ensemble de couverture la possibilité de le déplacer loin dans la région d'action tout en préservant la couverture totale de la région d'évènement. Pour ce faire, le SINK exécute la fonction move (ligne 6 de l'algorithme 3.5).

Si la mobilité de l'actionneur influe sur la couverture totale, cette fonction retourne une valeur négative (-1,-1), cet actionneur doit être ignoré (ligne 7-8 de l'algorithme 3.5). Le SINK commence par les actionneurs dont leur mouvement minimise au maximum la fonction objective. Il continue l'opération jusqu'à ce qu'il n'y a aucun actionneur à déplacer.

Algorithm 3 CRMEA: mobility optimization phase

```
1: ω = M; {M is the set returned by Algorithm 1}
2: while ω ≠ ∅ do
3:     selected = ∅;
4:     Δmin = f^M;
5:     for each a_i ∈ ω do
6:         (newX, newY) = Move(a_i);
7:         if (newX, newY) = (−1, −1) then
8:             ω = ω − a_i;
9:         else
10:            calculate fnew, the new objective function assuming that l_i =
               (newX, newY);
11:            if fnew < Δmin then
12:                Δmin = fnew;
13:                selected = a_i;
14:            end if
15:        end if
16:    end for
17:    if selected ≠ ∅ then
18:        ω = ω − selected;
19:        selected's location= (newX, newY);
20:        f^M = fnew;
21:    end if
22: end while
```

Algorithme.5 : Algorithme d'optimisation par la mobilité des actionneurs.

Cette optimisation vise à minimiser les chevauchements entre les actionneurs de l'ensemble de couverture, M.

Les deux phases d'optimisation peuvent être exécutées l'un après l'autre pour obtenir de meilleurs résultats.

III.3.2.4 Etude de la solution

Cette solution donne de meilleures performances en termes de coût d'action et énergie par rapport à la solution centralisée de Vedantham et al [2]. Les extensions de CRMEA (CRMEA+R et CRMEA+M et leur deux combinaisons) donnent des meilleurs résultats en les comparant avec CRMEA et à la solution de Vedantham en termes de coût d'action. Par contre, dans CRMEA+M, CRMEA+M+R et CRMEA+R+M le temps de réponse à un évènement est plus élevé à cause du temps pris par les actionneurs pour se déplacer. Cependant, la vitesse de l'actionneur peut être ajustée pour répondre aux exigences des applications en termes de délai d'action.

III.4 Solution de Derhab et Lasla

Dans cette solution, Derhab et Lasla [4] ont étudié le problème de couverture d'actionneur dans le but de répondre aux exigences des applications d'irrigation de précision dans

les WSANs qui sont : (1) le volume d'eau appliqué par les actionneurs doit correspondre à la demande en eau des plantes et (2) minimiser la sur-irrigation au maximum. Cependant, ils ont se concentré principalement sur l'étude du problème de couverture minimale

Par conséquent, une caractéristique importante d'un système d'irrigation de précision est que le temps, l'emplacement et le volume d'eau appliqué par les actionneurs (arroseurs) doit correspondre à la demande en eau des plantes. La sur-irrigation cause gaspillage de l'eau et des engrais, salinité, qui ont un effet négatif sur le terrain agricole. D'où, maximiser le non chevauchement dans la zone d'action devra être pris en considération lors de la conception d'une configuration de couverture pour des applications d'irrigation de précision. Pour remédier à ce problème, en premier lieu, ils ont pris une nouvelle approche pour définir et résoudre le problème de coût de couverture d'actionneur dans les applications des WSN basé-irrigation de précision. Deuxièmement, en se basant sur cette nouvelle définition, ils ont proposé deux algorithmes pour le problème de couverture d'actionneurs : un algorithme centralisé et un autre distribué.

Comme l'objectif des applications d'irrigation de précision n'est pas de minimiser la quantité totale d'eau consommée par l'ensemble des actionneurs, elles ne considèrent que les demandes de plantes (c'est-à-dire à l'intérieur de la région d'événement). Elles ne prennent pas en considération les régions couvertes par des actionneurs de l'ensemble de couverture se situant en dehors de la région d'évènement. Dans ce cas, l'objectif est de sélectionner l'ensemble de couverture qui : (1) maximise la zone non-chevauchée dans la région d'événement afin de correspondre à la demande en eau par les plantes, et (2) minimise la sur-utilisation de l'eau sur les zones à l'intérieur de la région événement. Bien qu'on ne considère pas les zones situées en dehors de la région d'évènement, mais les zones qui contient des plantes qui font partie du champ agricole, sont irriguées avant d'atteindre le seuil défini par l'application, mais la région d'évènement et les régions qui l'entourent ont des taux d'humidité très proches, et donc leurs besoins en eau sont également très proches [4].

Derhab et al [4] ont pris les mêmes calculs de la fonction objective décrits en [3], donc le problème est de minimiser les ressources supplémentaires consommées dans la région d'évènement, ce qui implique la minimisation de la fonction objective suivante :

$$f^M = Min(\sum_{i=2}^{\Delta(\xi^M)} \sum_{a \in S_i} (i-1) \times Size(a))$$

Pour répondre aux besoins en eau des plantes à l'intérieur de la région d'évènement, on doit aussi maximiser le non chevauchement dans les zones couverte par M.

Pour atteindre ces deux objectifs, on essaye de maximiser la fonction objective suivante :

$$Maximise(\alpha \times \sum_{a \in S_1^M} Size(a) - \beta \times f^{M})$$

Dans cette fonction objective a et β sont deux poids constants telles que $a + \beta = 1$.

III.4.1 Algorithme Centralized Actor-Coverage-Irrig (CACI)

Cet algorithme adopte une approche centralisée, où le SINK construit l'ensemble de couverture en se basant sur des connaissances globales du réseau. Lorsque le SINK reçoit les informations de présence d'un évènement, à chaque étape, il sélectionne l'actionneur qui maximise la fonction objective. La fonction objective d'un actionneur est donnée par : $A \times new\_area_i - \beta \times \Delta C_i$, où : new_area_i représente la taille de la nouvelle zone dans R couverte par l'actionneur a_i, et $\Delta C_i = f^{M \cup a_i} - f^{M}$ représente le coût supplémentaire que a_i apportera en cas où il serait choisi (line 7-14 de l'algorithme 3.7).

Algorithm 1 CACI Algorithm at node a_i

```
1: M = ∅;
2: R_M = ∅;
3: ω = Ω;
4: f^M = 0;
5: while R ⊈ R_M do
6:     max_function = −∞;
7:     for each a_i ∈ ω do
8:         if a_i covers new area in R then
9:             ΔC_i = f^{M∪a_i} − f^M
10:            function = α × new_area_i − β × ΔC_i
11:            if function > max_function then
12:                max_function = function;
13:                selected = a_i;
14:            end if
15:        else
16:            ω = ω − a_i;
17:        end if
18:    end for
19:    M = M ∪ selected;
20:    R_M = R_M ∪ (A_selected ∩ R)
21:    ω = ω − selected;
22: end while
23: Send command (start irrigation) to every actor M;
```

Algorithme.6 : Algorithme centrale CACI.

L'actionneur qui n'apporte aucune nouvelle couverture dans la région, il ne sera pas inclus dans M (ligne 15- 17 de l'algorithme 3.7). L'actionneur sélectionné est ajouté à M et l'algorithme se termine lorsque la région d'évènement est totalement couverte par M. Enfin, le SINK envoie à chaque actionneur de M un message de commande pour déclencher l'action.

III.4.2 Algorithme Distributed Actor-Coverage-Irrig (DACI)

Puisque dans l'approche distribuée de Vedantham, les actionneurs impliqués dans l'algorithme ne sont pas dans la région de dépendance l'un de l'autre, et par conséquent leur coordination est imparfaite. Il est possible que plusieurs actionneurs décident à agir en même temps, ce qui mène à une couverture d'actionneur moins optimale. Afin de résoudre ce problème, le schéma de communication de l'algorithme distribué devrait être conçu d'une façon que les actionneurs qui partagent la même région d'événement doivent se connaître. Pour ce faire, chaque capteur diffuse l'événement détecté dans un rayon de $(2 \times R_{max} + AR)$, où R_{max} est le rayon maximum d'une région d'événement. En outre, chaque actionneur doit se rendre compte de *son v*oisinage de distance $(2 (R_{max} + AR))$, comme représenté dans la figure 13.

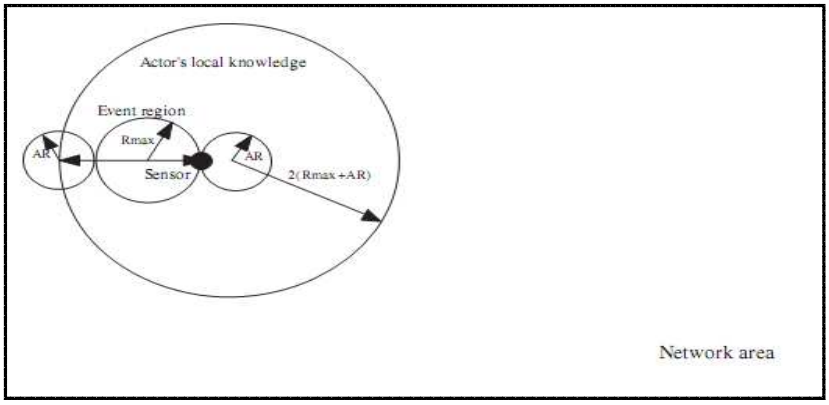


Figure 13 *: (2 (Rmax + 1'AR))* connaissance locale d'un actionneur dans le réseau.

Chaque capteur à l'intérieur de R diffuse l'événement qu'il détecte (c.-à-d., la quantité d'humidité dans la terre est au-dessous du seuil requis) à tous les actionneurs dans son (2 × Rmax + AR) voisinage qui sont intersectés avec R. Quand un actionneur reçoit une telle information, il détermine d'abord la région d'événement supplémentaire couverte par le capteur et ajoute cette région à la région d'événement déjà existante.

L'actionneur attend un temps donné pour assurer la réception de tous les événements dans la nouvelle région d'événement formée. Quand ce temps s'expire, tous les actionneurs qui sont intersectés avec la nouvelle région d'événement se connaissent. L'exécution de DACI est montrée dans l'algorithme 3.8. Le principal avantage de DACI est qu'il ne nécessite aucune coordination explicite entre les actionneurs, c'est-à-dire aucun message échangé entre les actionneurs pour construire l'ensemble de couverture. Tous les actionneurs intersectés avec R commencent par la même ω (ligne 3 de l'algorithme 3.8. Par conséquent, une coordination implicite peut être exploitée, c'est-à-dire que les informations de coordination sont déduites de l'environnement mais ne sont pas communiquées de manière explicite par des messages de

signalisation. Supposons que ω est un ensemble ordonné selon leur ID, et donc les éléments de ω sont choisis dans chaque actionneur dans le même ordre. Par conséquent, tous les actionneurs retournent le même ensemble *M*.

Algorithm 2 DACI Algorithm at node a_i

```
1: M = ∅;
2: R_M = ∅;
3: ω = the set of actors within (2(R_max + AR))-neighborhood that intersect with
   the event region R;
4: f^M = 0;
5: while R ⊈ R_M do
6:     max_function = −∞;
7:     for each a_i ∈ ω do
8:         if a_i covers new area in R then
9:             ΔC_i = f^{M∪a_i} − f^M
10:            function = α new_area_i − βΔC_i
11:            if function > max_function then
12:                max_function = function;
13:                selected = a_i;
14:            end if
15:        else
16:            ω = ω − a_i;
17:        end if
18:    end for
19:    M = M ∪ selected;
20:    R_M = R_M ∪ (A_selected ∩ R)
21:    ω = ω − selected;
22:    if selected = a_i then
23:        START Irrigation;
24:    end if
25: end while
26: return M;
```

Algorithme 7: Algorithme DACI à l'actionneur a_i.

III.4.3 Etude de la solution

L'algorithme centralisé CACI présente un coût de communication de $O(I+R^2_{max})\sqrt{N}$ et un temps de réponse de $O(2\sqrt{N}+Y)$. Par contre l'algorithme distribué DACI présente un coût de communication de $O(a^2 \times R^2_{max})$, un temps de réponse égal à $O(D)$ et un temps d'attente égal à $O(Y)$, où N est le nombre de nœuds(capteur et actionneur) dans le réseau, *Y :* le temps d'attente, *I* : la taille de l'ensemble de couverture, *a :* le rayon de la région d'évènement et *D* : la distance entre la région d'évènement et l'actionneur dans Ω(l'ensemble d'actionneur dans la région d'évènement) .

Le principal avantage de DACI est qu'il ne nécessite pas des messages supplémentaires pour exécuter la phase de coordination, ce qui conduit à réduire le retard évènement-action. En outre, son coût de communication est majoré par (R^4_{max}) comme l'algorithme NB de Vedantham. Le schéma de communication de l'algorithme DACI est conçu d'une façon à garder les avantages

des deux approches centralisée et distribuée sans hériter leurs faiblesses. DACI construit l'ensemble d'actionneurs de couverture avec le même coût optimal que CACI et le même coût de communication que l'algorithme NB de Vedantham. En plus DACI est meilleur que l'algorithme NB en termes de temps de complexité car il utilise une coordination implicite au lieu d'une coordination explicite pour sélectionner l'ensemble d'actionneurs de couverture.

III.5 Autre travaux liés à la couverture avec contraintes dans les WSANs

En plus de ces travaux sur la couverture avec contraintes dans les WSANs, il existe d'autres travaux similaires qui traitent ce problème. **Tomasso Melodia et al** [30] ont traité le problème de coordination dans les WSAN afin de couvrir une région quelconque, coordination capteur-actionneur et actionneur-actionneur. Dans le premier type de coordination, ils ont divisé la région d'évènement en clusters, dans chaque cluster il y'a un actionneur collecteur des informations de l'évènement depuis les capteurs membres du cluster. Par contre dans le deuxième type de coordination (actionneur- actionneur), ils ont introduit un modèle pour la coordination actionneur-actionneur dont l'objectif est d'assigner de façon optimale des tâches aux différents actionneurs pour achever en collaboration le but global (couverture d'une région). Ils ont proposé une technique qui permet de réduire l'énergie nécessaire pour les actions déclenchées qui consiste à réduire le rayon d'action de quelques actionneurs et cela basé sur une fonction qui cherche à maximiser la moyenne de l'énergie résiduelle dans l'ensemble des actionneurs de chaque partition ou cluster. Ce problème est différent de celui traité dans [2], [3] et [4] car il cherche à optimiser seulement la consommation d'énergie des actionneurs. **Akkaya et Younis** [31] ont abordé le problème de relocalisation des actionneurs après le déploiement dans le champ d'action. L'objectif est de trouver la meilleure localisation des actionneurs pour fournir une couverture maximale de la région d'intérêt et minimiser le temps de collecte de données ainsi que le temps d'exécution des actions par les actionneurs tout en prenant en considération la contraire d'énergie des capteurs.

Les deux travaux décrits précédemment sur la couverture dans les WSANs ne traitent pas notre problème. Soit ils cherchent à minimiser l'énergie consommée dans le réseau, soit ils cherchent une localisation optimale des actionneurs pour couvrir la totalité de la région d'intérêt pour assurer la couverture totale et la communication dans le réseau. Ces objectifs sont différents de notre objectif, car nous cherchons à minimiser tous les ressources consommées dans le réseau (ressources des actionneurs, ressources énergétiques,) et satisfaire quelques contraintes des applications des WSANs.

III.6 Conclusion

Dans ce chapitre, nous avons présenté les travaux existant dans la littérature qui traitent le problème de couverture avec contraintes dans les WSANs ainsi qu'une brève présentation des deux travaux liés à ce problème. Nous avons vu que peu de travaux qui traitent le problème de couverture avec contraintes dans les WSANs. Parmi les travaux nous avons cité ceux de Vedantham et al dans [2], ceux de Derhab et Zair [3] et de Derhab et Lasla [4].

Les autres travaux de [30] et [31] ne traitent pas notre problème, il utilise le terme couverture pour optimiser l'énergie consommée par les nœuds du réseau ou bien pour assurer la couverture totale et la communication dans le réseau sans chercher d'optimiser les ressources des actionneurs.

Étant donné que, ni les approches distribuée, ni les approches centralisées ne donnent des performances optimales, nous allons proposer dans le chapitre suivant une architecture semi-distribuée pour résoudre le problème de couverture avec contraintes dans les WSANs, tout en exploitant les avantages des approches centralisées et distribuées dans le cas des évènements dynamique.et faisant un compromis entre eux.

Chapitre 4

Architecture semi-distribuée pour la couverture avec contraintes dans les WSANs

IV.1 Introduction

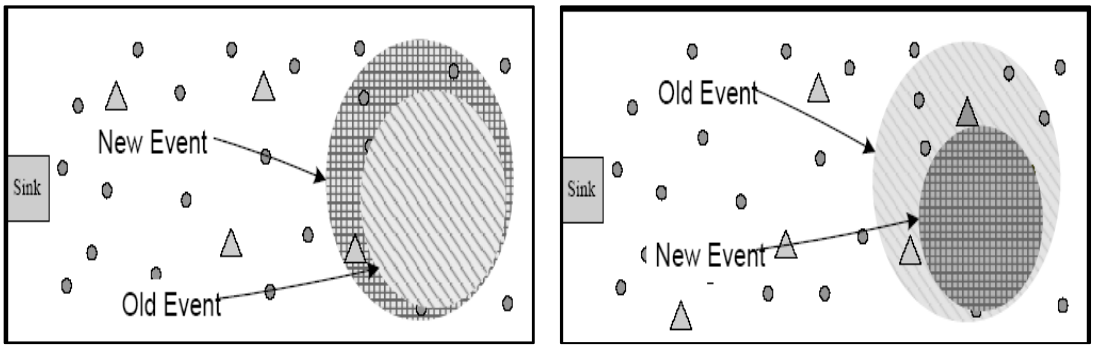
Dans le chapitre précédent, nous avons vu que l'approche distribuée a l'avantage de produire un nombre réduit de messages circulants dans le réseau et répond rapidement à l'évènement. Cependant et puisque chaque actionneur prend des décisions d'une façon autonome et ne possède pas une vision globale sur tout le réseau, le coût de l'ensemble de couverture sera moins optimal que celui de l'approche centralisée. D'autre part, les approches centralisées décrites dans [2] et [3] montrent de mauvaises performances en termes de nombre de messages circulants dans le réseau depuis les capteurs vers le SINK (pour transférer les informations captées) et depuis le SINK vers les actionneurs de l'ensemble de couvertures (commencer les actions sur la région d'évènement). A partir des avantages et des inconvénients de ces approches, nous déduisons que l'approche centralisée est une approche appropriée aux évènements statiques et n'ont pas aux évènements dynamiques qui changent fréquemment (coût élevé de communication, nécessites plus de temps pour réagir sur l'évènement). Par contre, l'approche distribuée est moins couteuse en termes de messages et plus rapide en termes de temps de réponse. Cependant, elle produit un ensemble de couverture moins optimal que l'approche centralisée.

Nous avons vu aussi que les travaux existants de la couverture avec contraintes dans les WSANs ne traitent pas le cas des évènements dynamiques.

Après une durée bien définie d'exécution, la région d'évènement se réduit ou s'augmente dans sa taille selon l'environnement. Dans le cas de la réduction de la taille de la région d'évènement, des actionneurs restent actifs durant le temps d'exécution d'actions inutilement, ce qui implique des ressources supplémentaires consommées sans aucun intérêt dans la région. Dans le deuxième cas où la taille de la région d'évènement s'augmente, il est nécessaire d'activer des actionneurs d'une façon optimale pour faire face au changement de l'environnement le plutôt possible en cas d'une application temps réel.

Par exemple dans le cas d'un incendie, les flammes sont plus condensées dans le centre de l'incendie, donc le feu commence de s'éteindre à partir des périmètres vers le centre ce qui produit des actionneurs activés inutilement dans les frontières de l'incendie c'est à dire des ressources supplémentaires consommées. Dans ce cas, il est nécessaire que la solution de

couverture avec contraintes prenne ce cas en considération. La même chose lorsque l'incendie se propage dans la région, il est nécessaire d'activer d'autres actionneurs pour l'éteindre. La figure 14 illustre les deux phénomènes de la réduction et l'augmentation de la région d'évènement.



(a) (b)

Figure 14: Evènement dynamique : (a) augmentation, (b) réduction.

Pour ces raisons, trouver une solution pour faire face à ces deux cas apparu est nécessaire.

Etant donné, comme nous avons vu, que ni les approches distribuée, ni les approches centralisées ne donnent des performances optimales en termes de coût de communication et de couverture, notre contribution consiste à proposer une architecture semi-distribuée pour résoudre le problème de couverture avec contraintes dans le cas des évènements dynamiques et exactement le cas d'une réduction de la région d'évènement. Cette architecture exploite les avantages de la solution centralisée en termes de coût de couverture et les avantages de la solution distribuée en termes de coût de communication.

Dans cette architecture, on exécute l'algorithme centralisé de couverture avec contraintes quand l'évènement est créé, et l'algorithme distribué lorsque la taille de l'évènement change (le cas des évènements dynamique). Elle complète les solutions centralisées proposée dans [2] et [3] avec l'ajout d'un algorithme distribué exécuté par les actionneurs lorsqu'il y' a une réduction dans la taille de la région d'évènement.

Dans cette architecture, nous utilisons deux niveaux d'exécution, le premier niveau se base sur la solution centralisée proposée par Derhab et al [3], où le SINK exécute un algorithme centralisé appelé CRMEA afin de sélectionner l'ensemble optimal des actionneurs qui maximise une fonction objective (minimise les ressources consommées dans le réseau) et envoie des messages (directives) de commandes d'activation vers ces actionneurs concernés. Ce message contient la liste des actionneurs de couverture, *M*. Cette liste est une liste ordonnée selon le ID des actionneurs. Le deuxième niveau est un algorithme distribué exécuté par les actionneurs de la liste *M* concernés par le changement de la région d'évènement. Chaque actionneur invoque son système local de contrôle pour exécuter un algorithme de redondance.

Dans le reste du chapitre, nous allons présenter le modèle utilisé dans notre solution ainsi que la fonction objective à utiliser pour sélectionner l'ensemble des actionneurs de couverture, *M*, et enfin nous détaillerons l'architecture proposée pour faire face aux évènements dynamiques tout en présentant les algorithmes utilisés.

IV.2 Modèle du réseau

Nous considérons un modèle de réseau de capteurs et actionneurs sans fil (WSAN) où les actionneurs et les capteurs sont déployés d'une manière aléatoire de telle sorte qu'ils couvrent toute la région d'intérêt, et il existe une station de base qui peut gérer le réseau et envoyer des commandes aux actionneurs. Nous supposons aussi les hypothèses suivantes :

- **Localisatio**n : nous supposons que les capteurs et les actionneurs sont capables de déterminer leur position géographique soit par le GPS [5] ou via un algorithme de localisation [6].
- **Protocole de routage :** nous supposons qu'il existe un protocole de routage fiable pour délivrer les messages dans le réseau entre les différents types de nœuds du réseau.
- **Région d'action d'un actionneur :** est la région circulaire où l'actionneur a_i peut agir, dénotée par A_i.
- **Région de capture :** est la région circulaire où les capteurs peuvent capter des informations, dénotée S_i.
- **Type d'évènement :** les évènements sont modélisée par des disques de rayon R_e et nous supposons qu'ils sont dynamiques, c'est-à-dire la région d'évènement change pendant l'exécution des actions par les actionneurs, donc il nécessite plusieurs rangs ou phases d'exécution.
- Nous supposons aussi que les actionneurs ont la possibilité de se déplacer et leur champ d'action est dynamique. Donc, on peut se retrouver dans quatre(04) cas : actionneur statique et son champ d'action fixe, actionneur statique avec champ d'action dynamique, actionneur mobile et son champ d'action fixe, et enfin actionneur mobile et son champs d'action dynamique.

IV.3 Définitions et notations

Avant d'aborder le cas des évènements dynamiques dans le problème de couverture avec contraintes, nous présentons quelques notions et définitions nécessaires pour la formulation de notre problème.

- Un point dans une région ***R*** est dit *K*-actionneur-couvert($K \geq 1$) s'il se trouve dans la région d'action de *K* actionneurs.
- Une région A est dite *K*-actionneur-couverte ($K \geq 1$) si chaque point dans A est k-actionneur-couvert. Le *K* est appelé degré de couverture de A, dénoté par δ(A).
- Nous prenons les notations décrites dans le travail de *Derhab* et *Zair* [3]. Nous définissons Ω comme étant l'ensemble des actionneurs qui se chevauchent avec la région d'évènement, $\varepsilon(R, \Omega)$ appelé la configuration de couverture d'actionneur si chaque point de R est dans le champ d'action d'au moins un actionneur de Ω. Comme illustre la figure 15, une configuration ζ partitionne la région d'évènement R en un ensemble de sous-régions disjointe dans R, dénoté par P.
- Comme décrit dans le travail de *Derhab* et *Zair*, nous définissons $S_i=\{q \in P \mid \delta(q)=i\}$ comme étant l'ensemble de sous-régions dans la région d'évènement A où leur degré de couverture est *i*. Le degré maximal de couverture d'une configuration ζ dénoté par Δ (ζ) est le plus grand degré de couverture parmi toutes les sous-régions de R.
- Considérons un actionneur a_i qui couvre une région $A \subseteq R$ dont sa taille est Size(A). Donc la quantité des ressources dépensées par a_i dans la région A est $Res(A,a_i)=ai \times Size(A) \times Ti$, tel que T*i* est le temps nécessaires pour exécuter l'action et a_i est la quantité des ressources consommées par unité de temps par unité de surface. Dans le reste de ce travail, nous ignorons a_i et T pour des raisons de simplicité.

Le problème de couverture avec contraintes défini dans le travail de *Derhab* et *Zair* consiste à trouver l'ensemble d'actionneurs $M \subseteq \Omega$ qui couvrent la région $R_M=R$ tel que la quantité de ressources supplémentaire consommée par ces actionneurs de M est minimale. Formellement, l'objectif est de minimiser la fonction objective f^M définie par :

$$f^M = Min\left(\sum_{i=2}^{\Delta(\xi^M)} \sum_{a \in S_i^M} (i-1) \times Size(a)\right)$$

Dans la figure 15, les valeurs 1, 2, 3, 4 représentent le degré de couverture de la sous-région (la valeur de *i* dans la formule précédente). La région d'évènement R_M représentée par la couleur orange, le champ d'action des actionneurs représenté par les disques vide, les triangles et les disques noirs représentent, respectivement, les actionneurs et les capteurs.

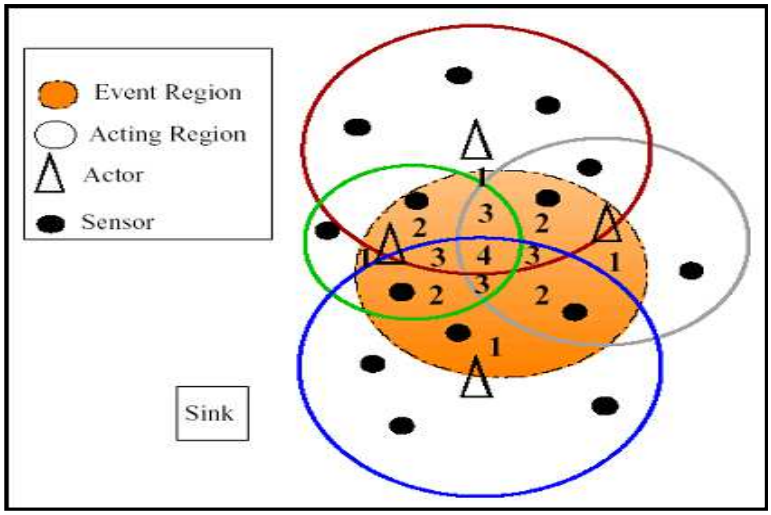


Figure 15 : Configuration de couverture des actionneurs

IV.3.1 Architecture semi-distribuée pour la couverture avec contraintes

Nous avons vu dans le chapitre précédent que les deux travaux de couverture avec contraintes dans les réseaux de capteurs et actionneurs sans fil ne prennent pas en considération le cas des évènements dynamiques. Ces solutions laissent les actionneurs actifs durant toute la durée d'exécution, ce qui engendre des ressources supplémentaires consommée par les actionneurs inutilement. Par exemple, dans le cas d'une extinction de feu, l'incendie se dégrade depuis les périmètres vers le centre, dans ce cas des extincteurs qui se situent dans les périmètres restent actifs jusqu'au la fin de l'opération.

Dans notre contribution, nous nous concentrons sur les évènements dynamiques, où la région d'évènements diminue dans sa taille comme résultat de l'exécution d'une directive d'action envoyée par le SINK après l'exécution de la solution centralisée proposée par *Derhab* et *Zair* dans [3].

Pour traiter ce cas, nous proposons une architecture qui se compose de deux niveaux hiérarchiques : le premier niveau utilise l'algorithme centralisé du CRMEA proposé par **Derhab** et **Zair** dans [3]. Ce dernier exécuté par la station de base (SINK) lorsqu'un nouvel évènement est détecté et le deuxième niveau est un algorithme distribué exécuté par les actionneurs lorsque les capteurs détectent un changement significatif dans la région d'évènement (réduction de la région ou bien une augmentation). La figure 16 illustre la boucle fermée de notre architecture.

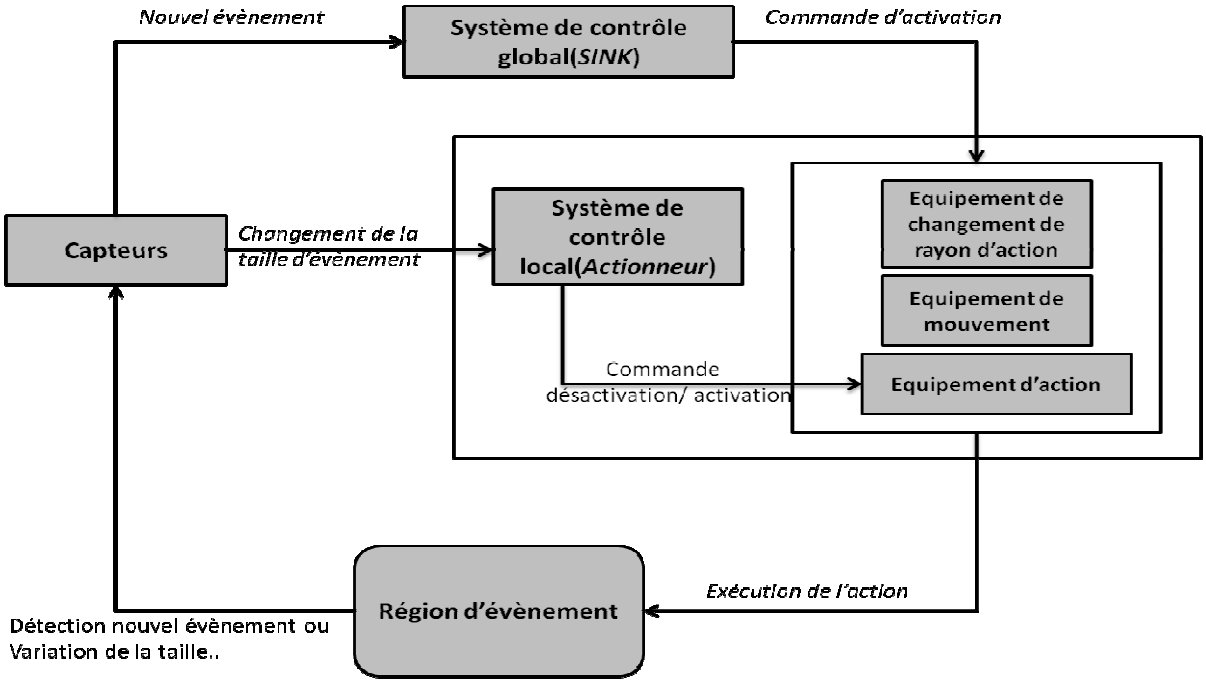


Figure 16 : Architecture semi-distribuée de couverture avec contraintes.

Dans ce qui suit nous allons présenter tout d'abord la solution centralisée proposée par *Derhab* et *Zair* et les deux optimisations : optimisation de champs d'action des actionneurs et la mobilité des actionneurs. Ensuite, nous présentons l'algorithme de redondance quand la taille de l'évènement diminue.

IV.3.2 Algorithme d'exclusion mutuelle centralisée basée-ressource (CRMEA) :

Lorsque les capteurs détectent l'existence d'un nouvel évènement, ils envoient les informations de détection à la station de base (SINK), supposant que le SINK est capable de déterminer les coordonnées de l'évènement et leur rayon en se basant sur les informations collectées à partir des capteurs. Le SINK exécute le CRMEA en deux phase : (1) la phase initiale calcul l'ensemble des actionneurs *M* pour *R* qui minimise le coût supplémentaire des ressources **(f^M (R_M, M)),** et (2) phase d'optimisation qui exécute un ou les deux extensions d'optimisation : réduction de la région d'action et la mobilité.

La première phase est présentée dans l'algorithme 3 du chapitre précédent, et les deux optimisations sont montrées dans les algorithmes 4 et 5, respectivement du même chapitre.

A la fin de la première phase, un ensemble *M* d'actionneurs est sélectionné pour être l'ensemble des actionneurs à activer pour agir sur la région d'évènement.

Les deux extensions peuvent être combinées c'est-à-dire exécutées une après l'autre.

IV.3.3 Algorithme de redondance distribué

Lorsque le SINK construit l'ensemble ***M*** des actionneurs, il envoie à chaque actionneur de cet ensemble un message de commande (une directive) contenant une liste ordonnée des actionneurs de ***M*** appelé ***L***. Si le SINK exécute l'extension de réduction de rayon ou l'extension de mobilité, le message de commande va contenir aussi le nouveau rayon d'action et/ou les nouvelles coordonnées de chaque actionneur de ***M***. Après avoir effectué l'action sur l'événement, il est probable que la région événement diminue dans sa taille comme illustré dans la figure 17. Dans ce cas, les capteurs détectent un changement significatif dans la région d'évènement (des valeurs captées au-dessous du seuil défini par l'application), ils informent les actionneurs dans sa région locale sur ce changement.

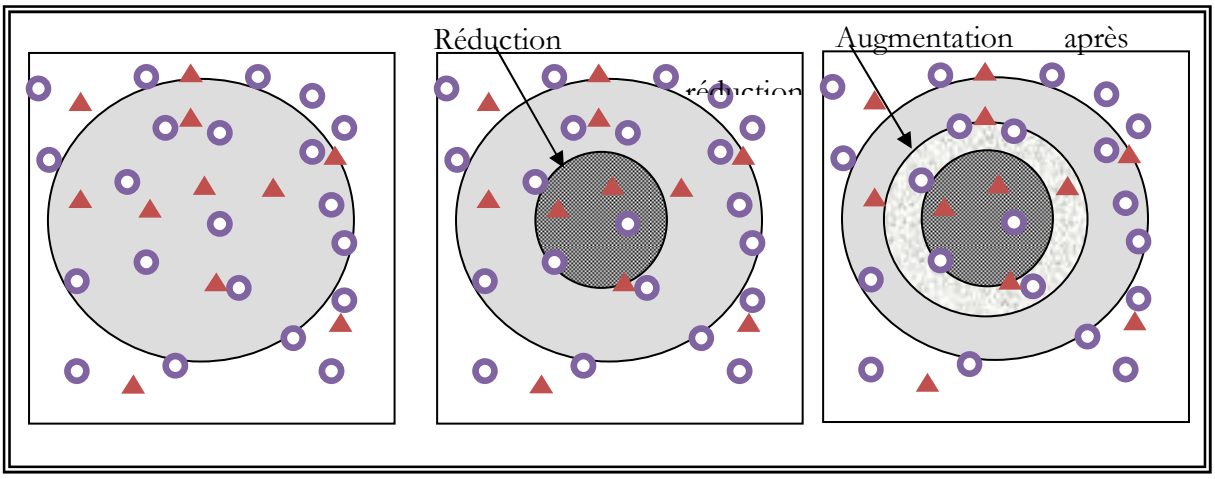


Figure 17 : Réduction et augmentation de la région d'évènements (a) l'ancien évènement, (b) réduction de l'évènement et (c) augmentation après réduction.

La définition de la région locale est : l'ensemble des actionneurs où la distance entre le capteur et l'actionneur inferieur ou égale à la somme des deux rayons : de capture et d'action, comme illustre la figure 18.

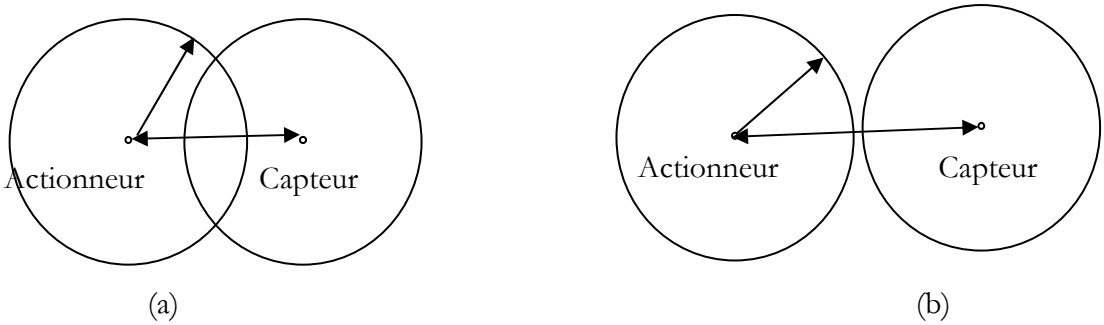


Figure 18 : Région locale des capteurs : (a) l'actionneur se situe dans la région locale du capteur, (b) l'actionneur n'est pas dans la région locale du capteur.

Lorsque les actionneurs de l'ensemble ***M*** reçoivent des informations de changement (réduction de la région d'évènement) depuis les capteurs, chaque actionneur dans la liste ***M*** exécute l'algorithme de redondance (Algorithme 8).

Algorithme 8 : Algorithme de redondance à l'actionner a_k :

```
1 : Bool= True ;
2 :     While Bool = True do
3 :             Bool =false ;
4 :             ΔMax =0 ;
5 :             Selected=Null ;
6 :         for each a_i ∈ M do //{ M est l'ensemble des actionneurs envoyé par le SINK}
7 :             if (M-a_i) encore couvre M then
8 :               Bool = True ;
9 :               ΔC = f^M- f^M- a_i ;
10 :                    if ΔC > ΔMax then
11 :                      ΔMax = ΔC ;
12 :                      Selected = a_i
13 :                    end if
14 :            end if
15 :        endfor
16 :         M = M- selected ;
17 :         stop action if (Selected = a_k) ;
18 :    end while
```

Algorithme 8 : L'algorithme de redondance.

Cet algorithme cherche les actionneurs dont l'élimination ou la désactivation n'influent pas sur la couverture totale de la région d'évènement (ligne 7 de l'algorithme 1). Il commence par l'élimination des actionneurs qui minimisent plus la fonction de coût des ressources supplémentaires jusqu'à ce qu'aucun actionneur n'est trouvé (lignes 9 à 12). Comme la liste ***L*** envoyée vers les actionneurs dans le message de commande est une liste ordonnée, tous les actionneurs retournent le même résultat. Lorsqu'un actionneur trouve lui-même comme un élément de la liste des actionneurs redondants, il arrête l'action sur la région d'évènement (se désactive).

IV.4 Conclusion

Dans ce chapitre nous avons proposé une architecture Semi-distribuée pour résoudre le problème de couverture avec contrainte dans le cas des événements dynamiques (réduction pendant l'exécution de l'action). Nous avons vu que notre architecture utilise deux niveaux d'exécution, un algorithme central exécuté par le SINK dans le cas d'un nouvel évènement détecté et un algorithme distribué exécuté par les actionneurs dans les cas de réduction de la région d'évènement (algorithme d'optimisation de redondance). Nous avons vu que notre architecture se base sur l'algorithme centralisé proposé par *Derhab* et *Zair* dans [3] et les deux optimisations : réduction de rayon d'action et la mobilité. Dans le chapitre suivant nous allons présenter les résultats de simulation de notre architecture tout en les comparants avec les résultats obtenus par les solutions proposées dans [2] et [3].

Chapitre 5

Résultats de simulation et évaluation des performances

V.1 Introduction

Apres avoir détaillé la conception de notre architecture semi-distribuée de la couverture avec contraintes dans les WSANs, nous allons présenter dans ce chapitre une analyse de la complexité de temps et de communication de notre architecture ainsi que les paramètres utilisés pour l'évaluation de performance de notre architecture et ceux de l'environnement de simulation. Puis nous discuterons les résultats obtenus tout en les comparant avec ceux obtenus dans [2] et [3].

V.2 Scenario de simulation

Après la réception des informations de présence d'un évènement depuis les capteurs, le SINK reformule les informations sur la région d'évènement telle que les coordonnées de son centre et leur rayon et calcule l'ensemble d'actionneurs de couverture en exécutant l'algorithme 3 du chapitre 3, proposé par Derhab et al [3], et/ou les optimisations proposées. Après une durée bien définie et à cause de l'intervention des actionneurs sur la région d'évènement, cette dernière est diminuée dans sa taille. Chaque capteur, dans la région où la réduction se produit, informe les actionneurs dans sa région locale de ce changement. Les actionneurs concernés invoquent leur système local pour exécuter l'algorithme de redondance localement.

V.3 Analyse de complexité

Dans cette section, nous analysons la complexité de communication (CC) et la complexité de temps de notre architecture semi-distribuée de couverture avec contrainte dans les WSANs et comparer les résultats obtenus avec les solutions centralisées et distribuées proposées dans [2] et [3]. La complexité de la communication mesure le nombre de transmission d'un saut nécessaire par l'algorithme pour effectuer l'action. La complexité de temps, ou délai événement-action, mesure la différence de temps entre l'occurrence de l'événement et l'exécution de l'action correspondante. Cette comparaison est présentée dans le tableau 1.

	Solution centralisée	Solution distribuée	Solution semi-distribuée
CC(évènement statique)	$O\left((I+1)\sqrt{N}\right)$	$O(K^2+K)$	$O((I+1)\sqrt{N})$
CC(évènement dynamique)	$O((\lambda+1)(I+1-\frac{\lambda}{2})\sqrt{N})$	$O((\lambda+1)(K^2+K))$	$O((I+1)\sqrt{N}+\lambda K)$
TC(évènement statique)	$O(\sqrt[2]{N})$	$O(W+1)$	$O(\sqrt[2]{N})$
TC(évènement dynamique)	$O(\sqrt[2]{N})$	$O(W+1)$	$O(1)$

Tableau 1 : Comparaison de complexité des solutions de couverture

Dans cette comparaison, nous supposons qu'il y'a *N* capteurs et actionneurs distribuée aléatoirement dans le champ d'action *A*. La densité des nœuds reste constante lorsque le nombre de nœuds augmente, et la zone *A* augmente dans sa taille aussi avec *N*. Comme la distance entre deux points déployés uniformément dans un carré de taille *a×a,* est proportionnelle à *a,* il est prévu que le nombre de sauts entre deux nœuds aléatoires augmente proportionnellement à *N*. Nous supposons également qu'il y'a *K* actionneurs dans la région locale de chaque événement. La taille de la région locale est constante, et donc le coût de communication au sein de cette région est *O(1). W* représente le temps d'attente de l'actionneur dans l'algorithme distribué de Vedantham. La taille de l'ensemble de couverture initialement construit par l'algorithme centralisé est dénoté par *I*, telle que *I<K*. Pendant la présence de l'évènement, la taille de la région d'évènement diminue **λ** fois.

Solution centralisée [2][3]: Lorsqu'un nœud capteur détecte la présence d'un nouvel événement, il envoie les informations détectées au SINK. Celui-ci envoie un message de commande à chaque actionneur dans l'ensemble d'acteur de couverture. Comme le nombre moyen de sauts entre deux nœuds aléatoires est proportionnelle à $\sqrt{N}$, cette opération conduit à un coût de communication de $O\left((I+1)\sqrt{N}\right)$ et un temps de réponse de $O(\sqrt[2]{N})$. Lorsque la taille de la région d'évènement est diminuée, le SINK doit reconstruire l'ensemble de couverture. Si I_c et I_p sont les cardinalités de l'ensemble actuel et précédent de couverture respectivement, la différence entre I_c et I_p n'est pas plus de 1, car nous considérons la complexité dans le pire de cas.

Nouvel évènement détecté : coût = $(I+1)\sqrt{N}$

1er changement coût = $I\sqrt{N}$

2eme changement coût = $(I-1)\sqrt{N}$

……..……..

λeme changement coût = $(I-(\lambda-1))\sqrt{N}$.

La somme de ces parties nous donne une complexité de communication égale à $((\lambda + 1)(\mathrm{I} + 1 - \frac{\lambda}{2})\sqrt{\mathrm{N}})$

Solution distribuée de Vedantham[2]: Lorsqu'un capteur détecte un nouvel événement, il envoie un message *demande* à tous les actionneurs de sa région locale. En plus, chaque actionneur envoie un message *NOTIFY* à *K−1* actionneurs dans sa région locale afin de construire l'ensemble d'actionneurs de couverture, ce qui conduit à un coût de $O(K^2+K)$. Dans le cas d'un évènement dynamique, cette opération se répète λ fois. Avant de commencer l'action, chaque actionneur attend un temps *W*, et attend un temps de *O(1)* pour recevoir le message REQUEST depuis les capteurs.

Solution semi-distribuée : le coût de communication en cas d'un évènement statique dans une solution semi-distribuée est le même que celui d'une solution centralisée (c'est-à-dire $O\left((I+1)\sqrt{N}\right)$. Chaque fois, que la taille de la région d'évènement se diminuée, les capteurs envoient un message *REQUEST* à tous les actionneurs dans sa région locale et comme ça il ne connaît pas l'ensemble d'actionneurs de couverture construit par le SINK. Apres la réception de ce message, chaque actionneur exécute l'algorithme 4.1 et n'a pas besoin de coordonner avec les autres actionneurs. Par conséquent, le coût de communication en cas d'événement dynamique est de $O((I+1)\sqrt{N}+\lambda K)$. En ce qui concerne le temps de réponse, l'algorithme semi-distribué donne la même complexité que celui du centralisé quand un événement est détecté pour la première fois. Quand la taille de l'évènement se diminue, l'algorithme ne nécessite que le temps de recevoir l'information de réduction.

Comparaison : dans le cas d'un ancien évènement, l'algorithme semi-distribué donne des mauvais résultats par rapport aux autres approches, tandis que l'algorithme centralisé donne les mêmes résultats. Par contre, en termes de complexité de communication, il est facile de montrer que l'algorithme semi-distribué est meilleur que celui distribué si les conditions suivantes sont vraies :

Dans le cas d'un évènement statique: $K>K_1 = \frac{\sqrt{4(I+1)\sqrt{N}+1}-1}{2} \square \sqrt{(I+1)\sqrt{N}}$

Dans le cas d'un évènement dynamique:

$$K > K_2 = \frac{\sqrt{4(\lambda+1)(I+1)\sqrt{N}+1}-1}{2(\lambda+1)} \square \sqrt{\frac{(I+1)\sqrt{N}}{\lambda+1}}$$

V.4 Résultats de simulation :

Nous avons utilisé le simulateur de réseau NS-2 pour simuler la complexité de temps et la complexité de communication. Nous avons choisi le protocole de routage temps réel SPEED [25] pour assurer la délivrance des messages depuis les capteurs vers le SINK et depuis le SINK vers les actionneurs dans le cas de l'architecture centralisée ainsi que depuis les capteurs vers les actionneurs dans la région locale dans le cas de l'architecture semi-distribuée et l'architecture distribuée.

Pour toutes les simulations, 120 actionneurs et 500 capteurs sont déployés d'une façon aléatoire sur une surface carrée de 400m×400m, la station de base (SINK) est un nœud puissant se situe à l'extrémité de la zone d'intérêt et sa position est (0,0). Le rayon de communication et de capture pour les capteurs est fixé à 30m, et le rayon de communication et d'action pour les actionneurs est fixé à 30m aussi.

Pour la simulation de couverture, nous avons implémenté tous les algorithmes avec le langage de programmation C++, sous un micro-ordinateur équipé d'un microprocesseur DUAL CORE 2.0GHz et 2Go de RAM.

V.4.1 Complexité de temps

Cette métrique calcule le temps de réponse en fonction de la distance entre la région d'évènement et le SINK pour un évènement de rayon égale à 60m, tout en variant la distance entre 10m et 300m. Ainsi le temps de réponse en fonction du rayon d'évènement tout en fixant la distance depuis le centre de l'évènement et le SINK à 141m, 282m et à 424m et nous varions le rayon de l'évènement de 10m à 250m.

La figure 19 représente l'impact de la distance entre la région d'évènement et le SINK sur le temps de réponse des actionneurs.

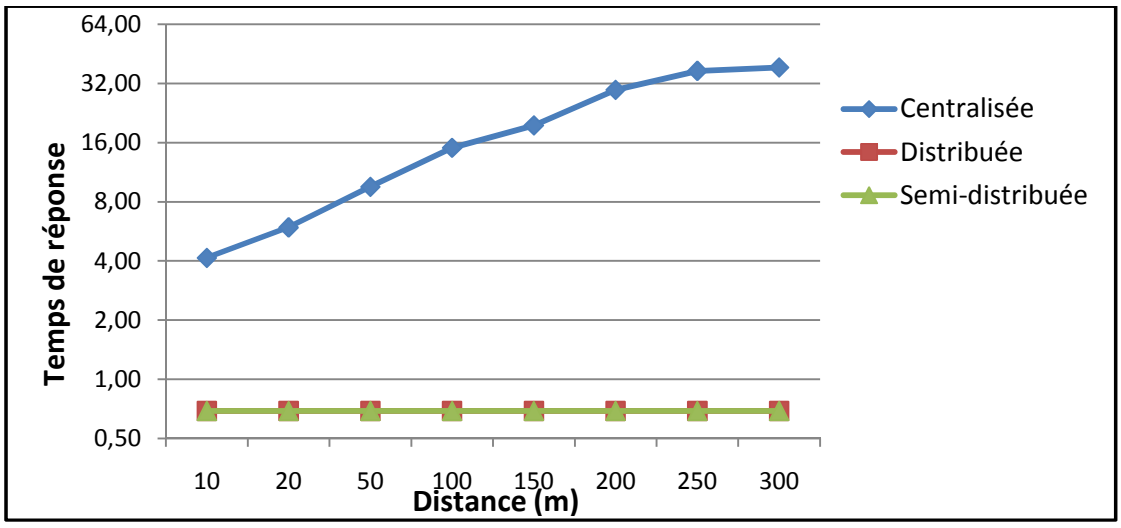


Figure 19: Temps de réponse en fonction de la distance vers le SINK

Nous remarquons que, dans la solution centralisée, le temps de réponse augmente avec l'augmentation de la distance vers le SINK à cause de nombre de sauts nécessaire pour que le message arrive au SINK et aussi pour que le SINK envoie des commandes d'activation vers les actionneurs de l'ensemble de couverture. Par contre le temps de réponse pour les deux autres solutions, distribuée et semi-distribuée, avec un évènement d'un rayon de 60m est presque constant car le SINK n'a pas besoin d'intervenir dans cette opération.

La figure 20 représente l'impact de la taille de l'évènement sur le temps de réponse.

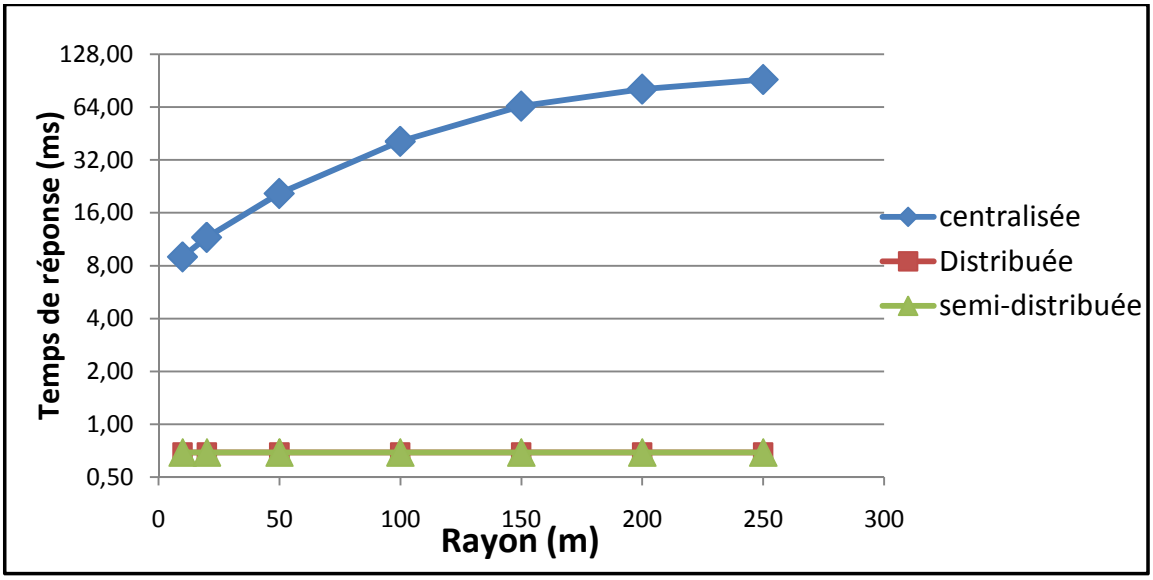


Figure 20 : Temps de réponse en fonction de la taille d'évènement.

Dans le graphe de la figure 20 nous remarquons que, dans la solution centralisée, le temps de réponse augmente avec l'augmentation de la taille d'évènement. Cette augmentation est due aux nombre de capteurs détectant l'évènement, ce qui engendre plus de messages depuis les capteurs vers le SINK. Par contre, dans les deux approches, distribuée et semi-distribuée, le temps de réponse ne dépend pas de la taille de l'évènement.

V.4.2 Complexité de communication :

Cette métrique calcule le coût de communication en fonction de la distance entre l'évènement et le SINK et en fonction de la taille d'évènement. Pour le coût de communication en fonction de la distance, nous fixons le rayon à 60m et nous varions la distance entre l'évènement et le SINK de 10 à 300m. La figure 21 montre l'impact de la distance sur le coût de communication.

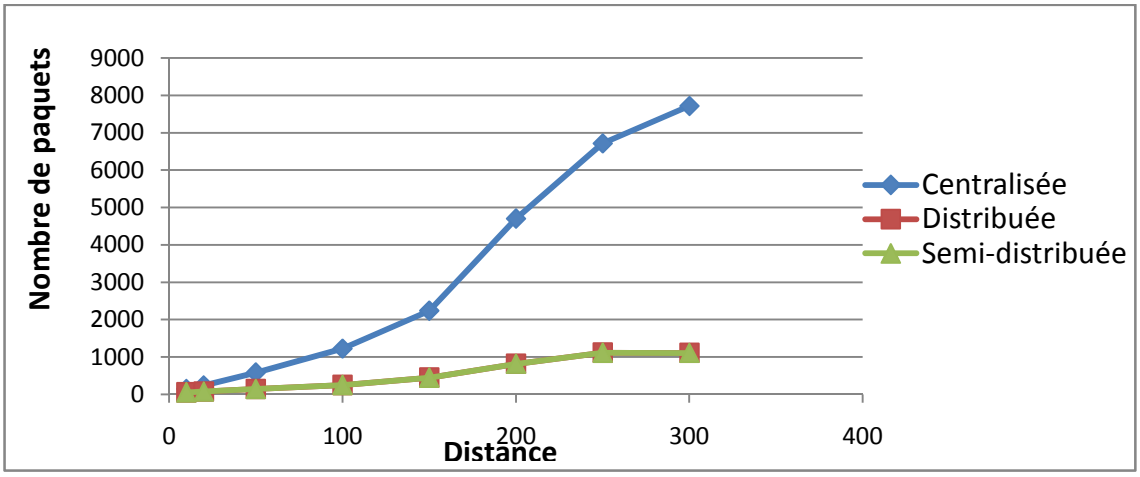


Figure 21 : le coût de communication en fonction de la distance.

Dans ce graphe, nous remarquons que les deux solutions distribuée et semi-distribuée représentent meilleur coût de communication par rapport à la solution centralisée, à cause des décisions locale. Nous remarquons aussi que le coût de communication n'augmente pas avec l'augmentation de la distance vers le SINK dans les deux approches distribuée et semi-distribuée.

En termes de coût de communication en fonction de la taille d'évènement, nous fixons une distance de l'évènement vers le SINK et nous varions le rayon de l'évènement de 10 à 300m. La figure 22 représente l'influence de la taille d'évènement sur le coût de communication.

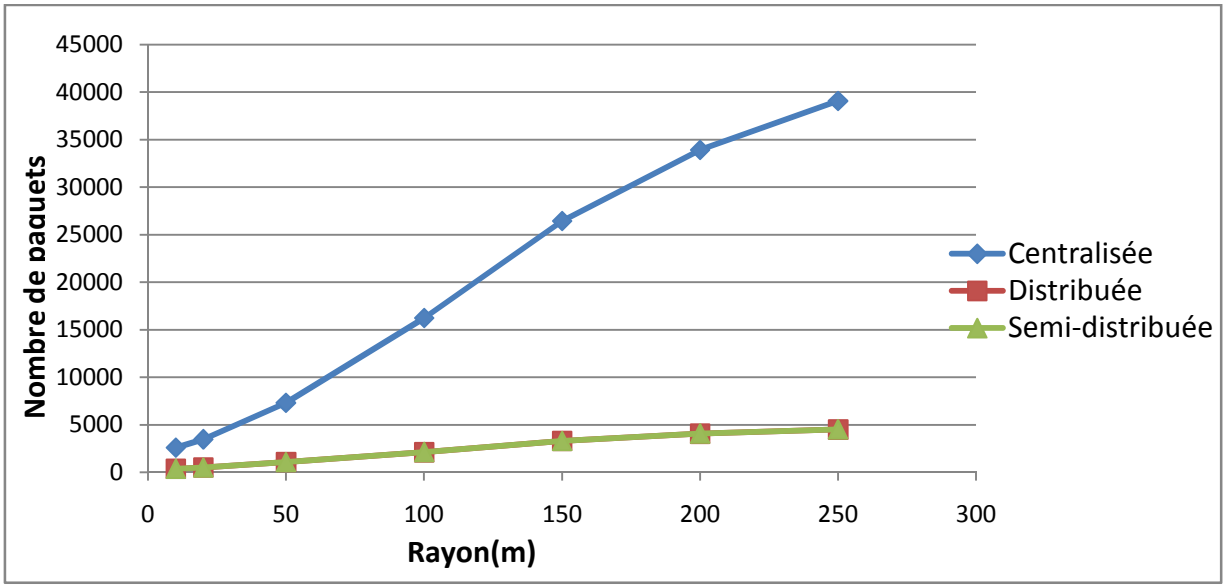


Figure 22: Coût de communication en fonction de la taille d'évènement.

Dans le graphe de la figure précédente, lorsque la taille de l'évènement augmente dans la solution centralisée, le nombre de capteurs dans cette région augmente aussi, ce qui engendre une

augmentation des messages depuis les capteurs détectant la présence de l'évènement vers le SINK et vice versa. Par contre dans les deux autre approches, distribuée et semi-distribuée, la taille de l'évènement n'influe pas sur le coût de communication à cause des décisions locale.

V.4.3 Simulation de couverture

Pour évaluer l'optimalité de notre architecture semi-distribuée, nous comparons leurs performance avec les solutions de Vedantham [2] et Derhab [3].

Dans cette simulation, 120 actionneurs avec un rayon d'action de 30m, déployés aléatoirement dans un champ d'action de 400×400 de telle sorte qu'ils assurent une couverture totale.

Dans nos résultats expérimentaux, chaque point représente la moyenne de 50 exécutions. Nous définissons les métriques suivantes pour évaluer notre architecture proposée pour la couverture avec contraintes dans les WSANs dans le cas des évènements dynamiques :

- **Coût de couverture :** défini comme la quantité des ressources dépensées sur la région d'évènement. Elle représente la somme des ressources requises et les ressources supplémentaires.
- **Coût d'énergi**e : l'énergie consommée par tous les actionneurs pour effectuer l'action. Nous supposons qu'un actionneur dont le rayon d'action est R, consomme $R^2 \times \pi$ Joules. Durant le mouvement, l'actionneur consommera une énergie supplémentaire proportionnelle à sa vitesse V.
- **Degré de couverture** : c'est le rapport entre le coût d'action et la taille de la région d'évènement donc plus cette valeur est proche de un (1) plus on a une meilleure couverture.

  Degré de couverture = $(S + f^M(R_M, M))/S = 1 + f^M(R_M, M)/S$, où S est la surface de la région d'évènement.
- **Degré maximal de couverture :** représente le nombre maximal d'actionneurs qui peuvent couvrir n'importe quel point dans la région d'évènement.

  La figure 23 représente le coût d'action en fonction de pourcentage de réduction

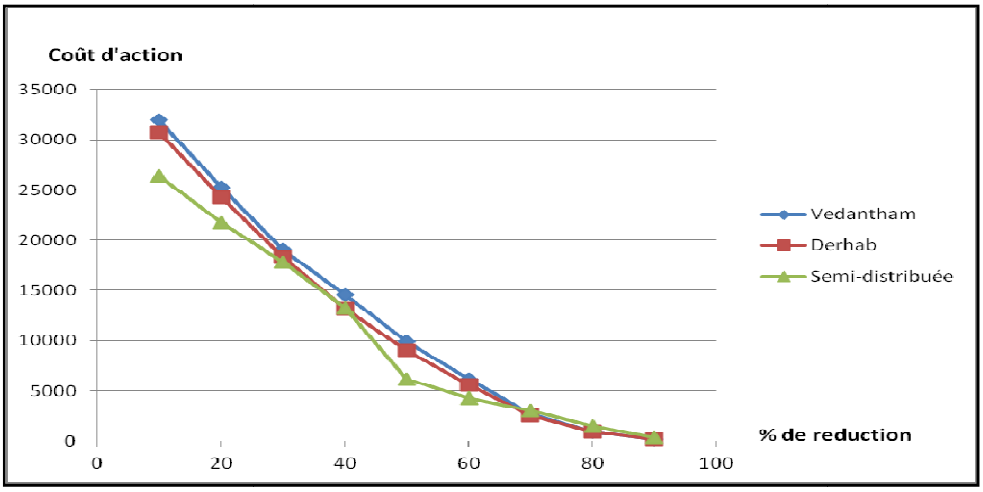


Figure 23 : Coût d'action en fonction de % de réduction.

Nous remarquons que les trois solutions donnent les mêmes résultats, lorsque le % de réduction augmente, le coût d'action diminué.

La figure 24 représente le coût d'énergie en fonction de la % de réduction.

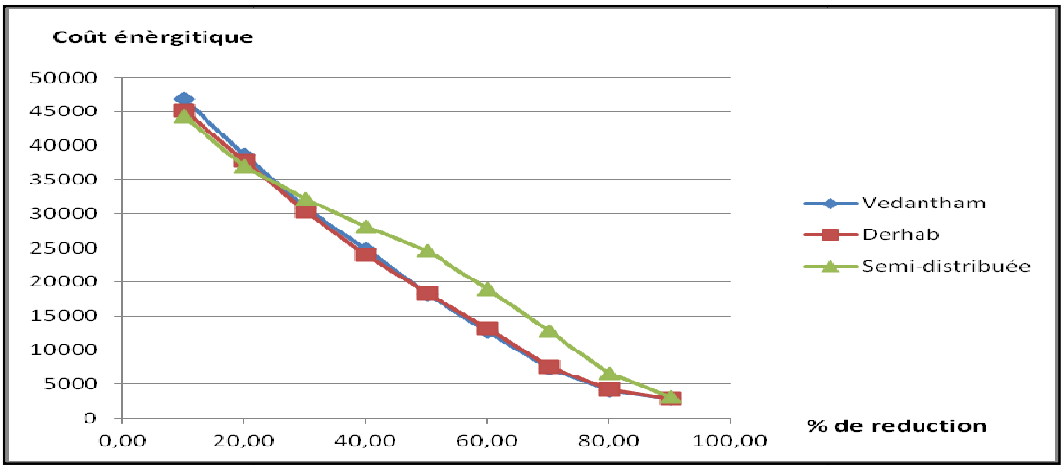


Figure 24 : Coût d'énergie en fonction de pourcentage de réduction de réduction.

Le coût d'énergie diminue avec l'augmentation du pourcentage de réduction dans toutes les solutions à cause de la réduction de la surface d'évènement. Nous remarquons que notre solution donne des performances semblables aux deux autres solutions de Vedantham et Derhab et Zair car les actionneurs ont la connaissance totale sur l'ensemble de couverture précédemment construit.

La figure 25 représente le degré de couverture en fonction du pourcentage de réduction.

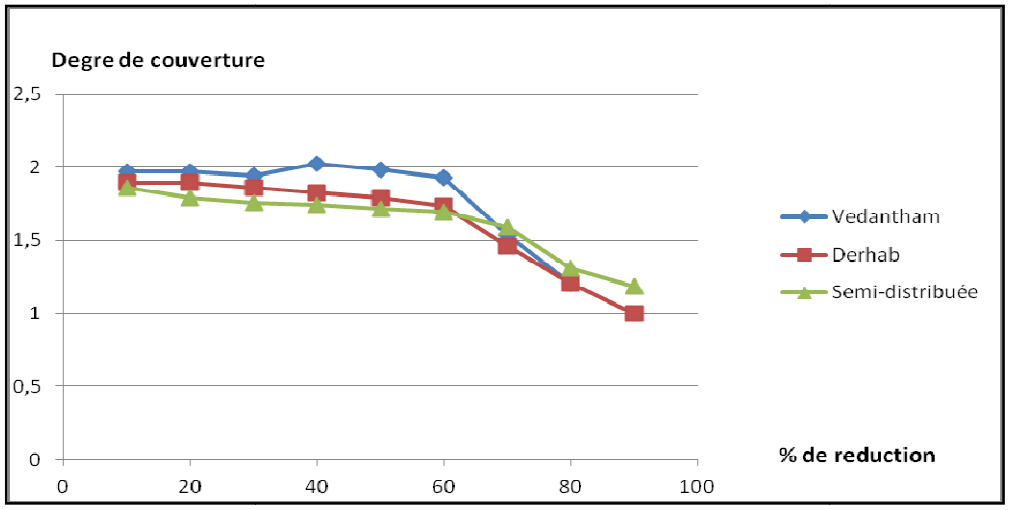


Figure 25 : Degré de couverture en fonction de % de réduction

Le degré de couverture diminue en augmentant le % de réduction jusqu'au obtenir le minimum vers le1 qui est le meilleur résultat.

La figure 26 représente le degré maximal de couverture

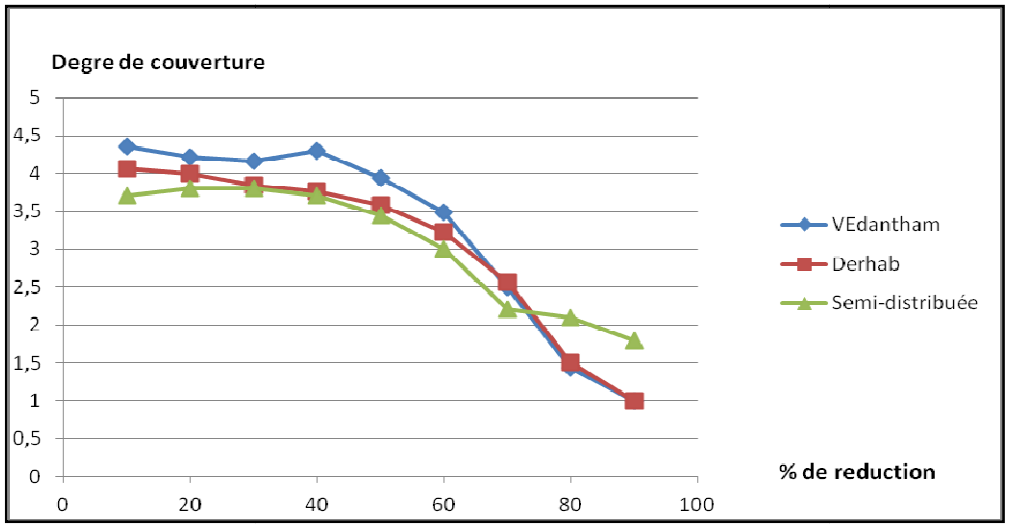


Figure 26 : Degré de couverture maximal

Nous remarquons que le degré de couverture diminue en augmentant le % de réduction à cause de la diminution de la région d'évènement.

V.5 Conclusion :

D'après les résultats de simulation, on constate que notre approche donne des performances meilleures que les deux solutions centralisée et distribuée en termes de coût de communication et de temps de réponse, et elle donne des performances similaires à la solution centralisée de Derhab et al [3]. Nous remarquons aussi que le coût d'action et le coût d'énergie se diminuent la réduction de la taille de la région d'évènement à cause de la réduction du nombre

d'actionneurs intervenant après le changement dans la région d'évènement. Aussi le coût de communication et le temps de réponse se diminuent avec la réduction de la distance vers le SINK et le rayon d'évènement. Finalement notre architecture a bénéficié des avantages de la solution distribuée en terme de coût de communication et temps de réponse dû aux décisions locales des actionneurs et les avantages de la solution centralisée en termes de cout de couverture et énergie dû à la connaissance de l'ensemble d'actionneurs de couverture calculé par le SINK dans la première phase de notre solution.

Conclusion générale

Les réseaux de capteurs et d'actionneurs constituent une très grande importance dans les présentes et futures applications grâce à leur faible coût, facilité de déploiement, et surtout les bénéfices qui peuvent apporter.

Dans ce travail, et après une étude des travaux existants dans la littérature, nous avons constaté que la terminologie exclusion mutuelle dans les WSANs n'est pas convaincante, car elle cherche toujours à trouver une couverture optimale sous quelques contraintes dans les WSANs. Pour cela, nous avons proposé de changer la terminologie vers la couverture avec contraintes dans les WSANS. Nous avons présenté un état de l'art sur les travaux dans le domaine de couverture avec contraintes, et après nous avons fait une comparaison entre les approches distribuées et centralisées où nous avons constaté que ni les approches distribuée ni les approches centralisées ne donnent des résultats optimales.

Nous avons proposé une architecture semi-distribuée de couverture avec contraintes pour bénéficier des avantages de la solution centralisée en terme de couverture et de bénéficier des avantages de la solution distribuée en terme de communication et de temps de réponse. Nous avons appliqué cette architecture sur les évènements dynamiques et plus précisément sur le cas de réduction des évènements.

Pour valider notre proposition, nous avons comparé par simulation notre architecture et celle proposée par Vedantham et al dans [2] et Derhab et Zair dans [3]. Les résultats de simulation montrent l'optimalité de notre solution en termes de coût de communication et temps de réponse.

Bibliographie:

[1] Ian F. Akyildi and Ismail H. Kasimoglu. “Wireless sensor and actor networks: research challenges” Ad Hoc Networks Journal, 2(4):351-367, 2004

[2] Ramanuja Vedantham, Zhenyun Zhuang, and Raghupathy Sivakumar. “Mutual exclusion in wireless sensor and actor networks”. In 3rd Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2006), 2006.

[3] Abdelouahid Derhab, and Mustapha Zair, *A Resource-based Mutual Exclusion Algorithm supporting Dynamic Acting Range and Mobility for Wireless Sensor and Actor Networks*, The International Workshop on Mobility in Wireless Sensor Networks (MobiSensor 2010) in conjunction with 6th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2010), June 2010**.**

[4] Abdelouahid Derhab, Nourredine Lasla: Distributed algorithm for the actor coverage problem in WSN-based precision irrigation applications. DCOSS 2011: 1-6.

[5] Nirupama Bulusu John, John Heidemann, and Deborah Estrin. Gps-less low cost outdoor localization for very small devices. IEEE Personal Communications Magazine, 7:28-34, 2000.

[6] Sadaf TANVIR, Localisation dans les Réseaux de Capteurs - Protocoles de Communication efficacité Energétique. Thèse de doctorat de l’UNIVERSITE DE GRENOBLE présentée le 25 Mars 2010.

[7] L. Lamport. “Time, clocks and the ordering events in a distributed system” Communication of the ACM, 21(7): 558-565, July 1978.

[8] Ricart et Agrawala

[9] O.S.F. Carvalho and G. Roucairol. On mutual exclusion in computer networks. Communications of the ACM, 26 :146–147, 1983. ……..

[10] Y.I. Chang. Notes on maekawa’s o(√n) distributed mutual exclusion algorithm. Proc. Symposium on Parallel and Distributed Systems, 1992.

[11] I. Suzuki and T. Kasami. A distributed mutual exclusion algorithm. In ACM Symposium on Principles of Distributed Computing (PODC), 1985.

[12] Y.I. Chang,M. Singhal, and M.T. Liu. A hybrid approach to mutual exclusion

for distributed systems. Proc 1990 Annual Inter. Comp. Soft. And Appli. Conf. IEEE Computer Soc, 1990. [13]. Raymond

[13] R. D. Carr, S. Doddi, G. Konjevod, and M. V. Marathe, "On the Red-Blue Set Cover Problem," in *Symposium on Discrete Algorithms*, 2000,pp. 345–353.

[14] T. Imienlinski and B.R. Badrinath. Querying in highly mobile distributed environments. Proceedings of the 18th VLDB, 1992.

[15] M. Singhal. A class of deadlock free maekawa type algortihms for mutual exclusion in distributed systems. Distributed Computing, 4 :145–159, 1985. **[16]** J. Walter, J. Welch, and N. Vaidya « A Mutual Exclusion Algorithm for Ad Hoc Mobile Networks » Wireless Networks 7, 585–600, 2001. Kluwer Academic Publishers. Manufactured in The Netherlands.

[17] R. Baldoni, A. Virgillito, and R. Petrassi. "*A distributed mutual exclusion algorithmfor mobile ad-hoc networks*". In Proc of the 7th IEEE Symposium on Computer and Communications (ISCC 2002), 2002.

[18] M. Benchaïba , M. Haddad et M. Ahmed-Nacer, "*Une approche de solution au problème d'exclusion mutuelle dans les réseaux mobiles ad hoc*" , rapport de recherche, USTHB, Février 2006.

[19] Abdelouahid Derhab and Nadjib Badache. "*A distributed mutual exclusion algorithm over multi-routing protocol for mobile ad hoc networks*". International Journal of Parallel, Emergent and Distributed Systems (IJPEDS), 23(3):197 { 218, 2008}.

[20] J. Walter, J. Welch, and N. Vaidya, "*A k-mutual exclusion algorithm for ad hoc mobile networks*", Dial M for Mobility workshop, Dallas, TX, USA, 1998, October.

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

[22] E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis, V.Z. Groza, Sensor-based information appliances, IEEE Instrumentation and Measurement Magazine 3 (4) (2000) 31–35

[23] M. Conti, S. Giordano, G. Maselli, G. Turi, "Cross-layering in mobile ad-hoc network design", IEEE Computer, Special Issue on AdHoc Networks 37 (2) (2004) 48–51.

[24] H.S. Kim, T.F. Abdelzaher, W.H. Kwon, "Minimumenergy asynchronous dissemination to mobile sinks in wireless sensor networks", in: Proc. of the

First ACM Int. Conf. on Embedded Networked Sensor Systems (ACM Sensys_03), November 2003, pp. 193–204

[25] T. He, J. Stankovic, C. Lu, T. Abdelzaher, "SPEED: A realtime routing protocol for sensor networks", in: Proc. IEEE Int. Conf. on Distributed Computing Systems (ICDCS),Rhode Island, USA, May 2003, pp. 46–55.

[26] B. Tavli, W. Heinzelman, "TRACE: Time reservation using adaptive control for energy" e.ciency, IEEE Journal on Selected Areas of Communication 21 (10) (2003) 1506–1515.

[27] M.Caccamo, L.Y. Zhang, L. Sha, G. Buttazzo, "An implicit prioritized access protocol for wireless sensor networks", in: Proc. IEEE Real-Time Systems Symp, December 2002, pp. 39–48.

[28] A.J. Goldsmith, S. Wicker, "Design challenges for energy constrained ad-hoc wireless networks", IEEE Wireless Communications 9 (4) (2002) 8–27.

[29] I. Chlamtac, M. Conti, J.N. Liu, "Mobile ad-hoc networking : imperatives and challenges", Ad Hoc Networks 1(1) (2003) 13–64.

[30] Tommaso Melodia, Dario Pompili, Vehbi C. Gungor and Ian F. Akyildiz « Communication and Coordination in Wireless Sensor and Actor Networks », IEEE transactions on mobile computing, vol.6, N°.10,october 007.

[31] Kemal Akkaya, Mohamed Younis, "COLA: A Coverage and Latency aware Actor Placement for Wireless Sensor and Actor Networks", i the Proceedings of IEEE Vehicular Technology Conference, 2006, Montreal, Que.

[32] U. Feige, "A Threshold of ln n for Approximating Set Cover," in *28th ACM Symposium on Theory of Computing*, Philadelphia, USA, May 1996, pp. 314–318.

[33] D. Johnson, "Approximation Algorithms for Combinatorial Problems," in *Journal of Computer and System Sciences*, 1974.