الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Seddik Benyahia - Jijel

Faculté des Sciences Exactes et Informatique

Département d'Informatique

Mémoire

pour l'obtention du diplôme de

Master

en Informatique

Option : Réseaux et Sécurité

Thème

Réseaux de capteurs sans fil : Mesure de l'efficacité énergétique des techniques de clustering

Présenté Par :

MENHANE Chihab Eddine

ABDELOUAHAB Aissa

Encadré par :

Dr. KARA Messaoud

Année universitaire : 2018-2019

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Seddik Benyahia - Jijel
Faculté des Sciences Exactes et Informatique
Département d'Informatique

1/2

Mémoire

pour l'obtention du diplôme de

Master

en Informatique
Option : Réseaux et Sécurité

Thème

Réseaux de capteurs sans fil : Mesure de l'efficacité énergétique des techniques de clustering

Présenté Par :

MENHANE Chihab Eddine
ABDELOUAHAB Aissa

Encadré par :

Dr. KARA Messaoud

Année universitaire : 2018-2019

Résumé

Un réseau de capteurs sans fil (RCSF) est constitué d'un ensemble de nœuds equipés de capteurs afin de surveiller une ou plusieurs grandeurs physiques sur une zone d'intérêt pour répondre à diverses applications dans différents domaines. Chaque nœud dispose, en général, de faibles ressources : puissances de calcul faible, mémoire limitée, portée de communication réduite et surtout une source d'énergie limitée (une pile) qui affecte directement la durée de vie d'un capteur et par conséquent la durée de vie du réseau entier.

La contrainte énergétique est le problème le plus important dans les RCSFs. Étant donnée que la majeure partie de l'énergie du réseau est consommée pendant le processus de communication, il est nécessaire d'utiliser des techniques efficaces pour la réduction de la consommation d'énergie.

Un moyen pour réduire la consommation d'énergie dans les RCSFs consiste à utiliser des algorithmes de clustering qui permet de partitionner le réseau en clusters. A chaque cluster est élu un cluster-head qui organise la communication intra-cluster et se charge de la communication avec la station de base. Ce qui, permet de réduire efficacement le nombre de communications et prolonger ainsi la durée de vie du réseau.

Dans ce mémoire, nous nous sommes interessés à un protocole de clustering écoénergétique basé sur les algorithmes de colonies de fourmis ACO-C (Ant Colony Optimization for Clustering). Ce protocole est mise en œuvre au niveau de la station de base.

En utilisant des fonctions de coût appropriées et de l'agrégation de données des nœuds de chaque cluster, il minimise et répartisse le coût des transmissions longue distance.

Afin évaluer ce protocole, nous avons développé sous Java un outil de simulation CLUSTERING_APPLICATION.

Le protocole ACO-C que nous avons amélioré a été comparé avec le protocole LEACH-C. Les résultats de simulation montrent que le protocole ACO-C est plus peformant en termes d'efficacité énergétique et de prolongation de durée de vie du réseau.

Remerciements

Louange à Dieu, le miséricordieux, sans lui rien de tout cela n'aurait pu être. Nous tenons à réitérer nos remerciements les plus forts au Dr. KARA Messaoud, pour ses conseils et son soutien qui ont fait beaucoup la différence au long de la préparation de ce mémoire et nous a aidé à le finaliser.

Notre gratitude va aussi aux membres de jury d'avoir accepté de juger ce travail.

Un énorme MERCI à nos familles et amis pour leur éternel soutient et la confiance qu'ils ont en nos capacités.

Nous remercions également tous ceux qui ont contribué de près ou de loin à la concrétisation de ce travail. Qu'ils trouvent tous ici l'expression de nos sincères considérations..

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

ACA-LEACH	Ant Colony Algorithm Low Energy Adaptive Clustering Hierarchy
ACO	Ant Colony Optimisation
ACO-C	Ant Colony Optimization for Clustering
ATPC	Adaptive Transmission Power for Wireless Sensor Networks
BS	Station De Base
CH	Cluster-Head
EEUC	Energy-Efficient Unequal Clustering
HEED	Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad-hoc Sensor Networks
LEACH	Low Energy Adaptive Clustering Hierarchy
LEACH-C	Low Energy Adaptive Clustering Hierarchy Centralized
LMA	Local Mean Algorithm
LMN	Local Mean of Neighbors Algorithm
MAC	Media Access Control
MN	Nœud Membre
PEGASIS	Power-Efficient Gathering in Sensor Information Systems
QoS	Qality of Service
RCSF	Réseau de capteurs sans fil
STEM	Sparse Topology and Energy Management
TDMA	Time-Division Multiple Access
TEEN	Threshold sensitive Energy Efficient sensor Network
TSP	Travelling Salesman Problem
WSN	Wireless Sensor Network

La convergence des progrès, d'une part dans la microélectronique et, d'autre part, dans les technologies de communication sans fil, a permis la production de capteurs de communication à faible coût et à faible consommation et connectés en réseau. Dans le contexte de surveiller et de contrôler les environnements hostiles, ces petits objets électroniques ont pour objectif de collecter et éventuellement de traiter des grandeurs physiques (luminosité, mouvement, température, pression atmosphérique, etc). De plus, ils doivent être indépendants, de petite taille et peuvent être déployés de manière dense et aléatoire dans le champ supervisé. Ces dispositifs sont généralement appelés "nœuds capteurs" (ou simplement "capteurs") et peuvent être distribués sur terre, dans des véhicules, dans des bâtiments et même sur le corps humain. Parce qu'ils peuvent être connectés sans être physiquement connectés. En plus, récemment ces composants sont munis d'un système de communication leur permettant de communiquer avec d'autres capteurs, et ainsi les réseaux de capteurs sans fil sont nés.

Ces réseaux sont différents des autres réseaux sans fil car ils ont, en général, les spécificités suivantes : une grande densité, faible débit, faible capacité d'énergie et un environnement inaccessible. Ces deux dernières spécificités ont fait de l'énergie une contrainte très importante puisque les batteries des capteurs ne sont pas généralement rechargeables ni remplaçables. Pour prolonger la durée de vie d'un réseau de capteur sans fil tout en assurant les trois tâches principales d'un nœud capteur : capture, traitement et envoi des données, il faut bien conserver l'énergie des nœuds capteurs. Parmi ces trois tâches, l'envoi des données ou la communication est la tâche qui consomme la plus grande partie de l'énergie.

Les réseaux de capteurs sans fil sont utilisés aujourd'hui dans de nombreuses applications qui incluent notamment : la domotique, la santé, le domaine militaire ou encore la surveillance de phénomènes environnementaux. Toutefois, le dénominateur commun de toutes les applications de réseaux de capteurs reste la limite des capteurs en raison de leurs ressources matérielles limitées dont la plus contraignante est l'énergie. Ce mémoire a pour objectif d'étudier les techniques permettant d'améliorer l'efficacité énergétique des réseaux de capteurs sans fil denses.

Le clustering est une approche efficace de contrôle de la topologie permettant de maximiser la durée de vie et l'évolutivité des RCSFs. Le routage basé sur la hiérarchie fait partie du routage basé sur le groupe et consiste à créer une hiérarchie virtuelle entre les nœuds du réseau de capteurs. Cette classe de techniques de routage est généralement conçue pour les réseaux à grande échelle

et vise à maintenir efficacement la consommation d'énergie et augmente la durée de vie du réseau en organisant l'ensemble du réseau en clusters.

Chaque cluster est dirigé par un nœud appelé Cluster-Head (CH) qui reçoit les données des nœuds situés dans le groupe. Les CHs se communiquent afin de trouver un meilleur itinéraire jusqu'au nœud collecteur ou à la station de base (BS). Ceci est fait afin de réduire la consommation d'énergie des nœuds capteurs en réduisant le nombre des messages transmis/reçus au noeud récepteur.

Les algorithmes basés sur les colonies de fourmis (ACO – Ant Colony Optimisation) sont l'une des solutions aux problèmes de clustering dans lequel N objets sont affectés à K clusters. La fonction objectif, qui doit être minimisée, est en général le total euclidien des distances entre chaque noeud et son cluster-head.

Dans ce mémoire, nous concentrons nos efforts sur les techniques permettant de prolonger la durée de vie du réseau, notament les techniques de clustering basé sur les algorithmes de colonies de fourmies.

Organisation du mémoire

Ce mémoire est organisé en 4 chapitres présentés comme suit :

- ✓ ***Chapitre 1 : Etat de l'art sur les Réseaux de Capteurs Sans Fil et la consommation d'énergie***

  Dans ce chapitre, nous présentons d'abord un état de l'art sur les réseaux de capteurs sans fil : leurs définitions, caractéristiques, spécificités, applications, contraintes …

  Ensuite, nous aborderons le sujet de consommation et de concervation de l'énrgie dans les RCSFs. Nous étudiant les facteurs intervenant dans la consommation d'énergie et les techniques permettant de la réduire et ainsi prolonger la durée de vie du réseau.

- ✓ ***Chapiter 2 : Etude des protocoles de routage hiérarchique et des techniques de clustering dans les réseaux de capteurs sans fil (RCSFs)***

  Dans ce chapitre, nous nous intéressons aux approches de routage hiérarchique et étudier les techniques de clustering dans les réseaux de capteurs sans fil. Dans la première partie, nous discuterons des caractéristiques, objectifs et des avantages du clustering et nous parlerons dans la seconde partie de certaines techniques de routage hiérarchique.

- ✓ ***Chapiter 3 : Les protocoles de clustering implémentés***

  Dans ce chapitre on commence par introduire les algorithmes de colonies de fourmis (ACO), puis on présentera les deux protocoles de clustering que nous avons sélectionnés pour l'implémentation. Le premièr protocole est basé sur les algorithmes de colonies de fourmis (ACO for Clustering). Le deuxième est basé sur la technique de recuit simulé (LEACH-C).

- ✓ ***Chapiter 4 : Implémentation et résultats de simulations***

  Dans ce chapitre, nous allons tout d'abord présenter l'outil CLUSTERING_APPLICATION que nous avons développé pour l'implémentation et la simulation de deux algorithmes de clustering dans les réseaux de capteurs sans fil (ACO-C [39], LEACH-C [77]).

  Ensuite, avec cet outil nous effectuons des simulations afin d'évaluer la performance de l'algorithme (ACO-C), par rapport au protocole LEACH-C.

Le mémoire se termine par une conclusion générale et des perspectives du travail réalisé.

Chapitre 1 : Etat de l'art sur les Réseaux de Capteurs Sans Fil et la consommation d'énergie

1.1 Introduction

Les réseaux de capteurs sans fil (RCSFs) sont constitués d'un grand nombre de nœuds capteurs qui sont généralement alimentés par batterie et conçus pour fonctionner pendant une longue période. Les domaines d'application sont nombreux et variés, tels que le domaine environnemental, médical ou militaire. L'atout majeur de ce dispositif est un déploiement à grande échelle sans aucune maintenance. Les capteurs n'ont pas besoin d'une infrastructure établie pour parvenir à transmettre des données vitales à l'étude sur l'environnement. Il est nécessaire également de garantir une bonne qualité de service, car les réseaux de capteurs sans fils doivent intégrer des mécanismes qui permettent aux utilisateurs de prolonger la durée de vie du réseau en entier. La durée de vie du réseau de capteurs est la période de temps pendant laquelle tous les nœuds peuvent : maintenir la connectivité, couvrir le domaine ensemble, ou maintenir le taux de perte d'information en dessous d'un certain niveau. La vie du système est donc liée à la vie nodale, bien qu'elle puisse différer. La vie nodale est la vie d'un nœud du réseau. Cela dépend principalement de deux facteurs :

- L'énergie qu'il consomme en fonction du temps et la quantité de l'énergie disponible.
- La partie prédominante de l'énergie est consommée par un nœud capteur lors de la détection, communication puis traitement des données.

Il existe différentes définitions pour la durée de vie d'un réseau de capteurs (en fonction de la fonctionnalité souhaitée). Elle peut être définie par la durée qui s'écoule entre le déploiement du réseau et celle jusqu'à la mort du premier nœud. Elle peut également être définie par le temps jusqu'à ce qu'une proportion (pourcentage) de nœuds meurt. Si la proportion de nœuds morts dépasse un certain seuil [1]. C'est pourquoi, il est nécessaire d'optimiser la consommation d'énergie à tous les niveaux de conception de ce type de réseau. Par conséquent, la minimisation de la consommation d'énergie est un facteur de conception des plus importants dans les réseaux de capteurs.

1.2 Généralités sur les réseaux de capteurs sans fils (RCSFs)

1.2.1 Définition

Les réseaux de capteurs sans fil sont des réseaux spontanés constitués de nœuds déployés en grand nombre pour collecter et transmettre des données à un ou plusieurs points de collecte de

manière autonome. Les nœuds sont des capteurs intelligents capables d'accomplir trois tâches essentielles : la mesure d'une quantité physique, le traitement eventuel de ces informations, et communication avec d'autres capteurs. Tous ces capteurs sont déployés de manière aléatoire pour une application, formant un réseau de capteurs [2].

1.2.2 Nœud capteur

Un nœud capteur est un petit appareil autonome avec des ressources limitées [3], il est capable d'effectuer de simples mesures et recueille des informations sur l'environnement dans lequel il est placé telles que la température, les vibrations et la pression [4] et de les transmettre à d'autres dispositifs grâce aux ondes radios sur une distance limitée [3]. La *Figure 1* montre 3 capteurs physiques disponibles sur le marché.

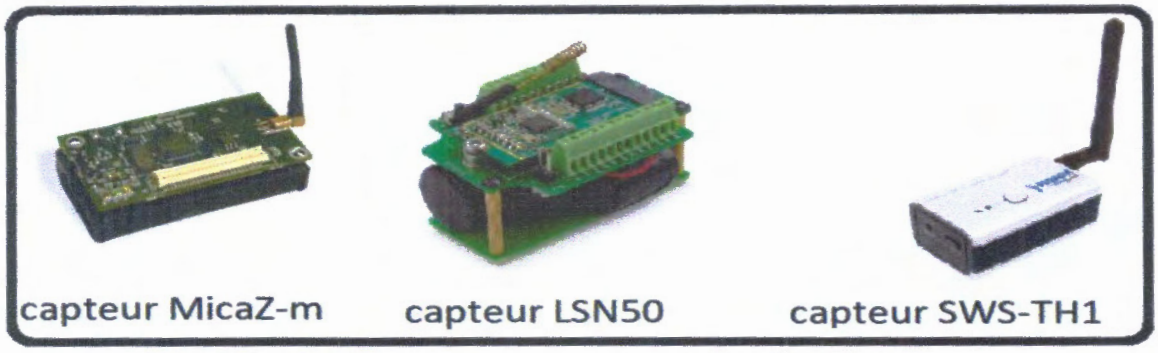


Figure 1 : Exemples de capteur sans fils

1.2.3 Architecture d'un nœud capteur

Un nœud capteur est composé de quatre composants de base comme le montre la *Figure 2* qui sont : l'unité de capture, l'unité de traitemment, l'unité de communication et l'unité d'énergie.

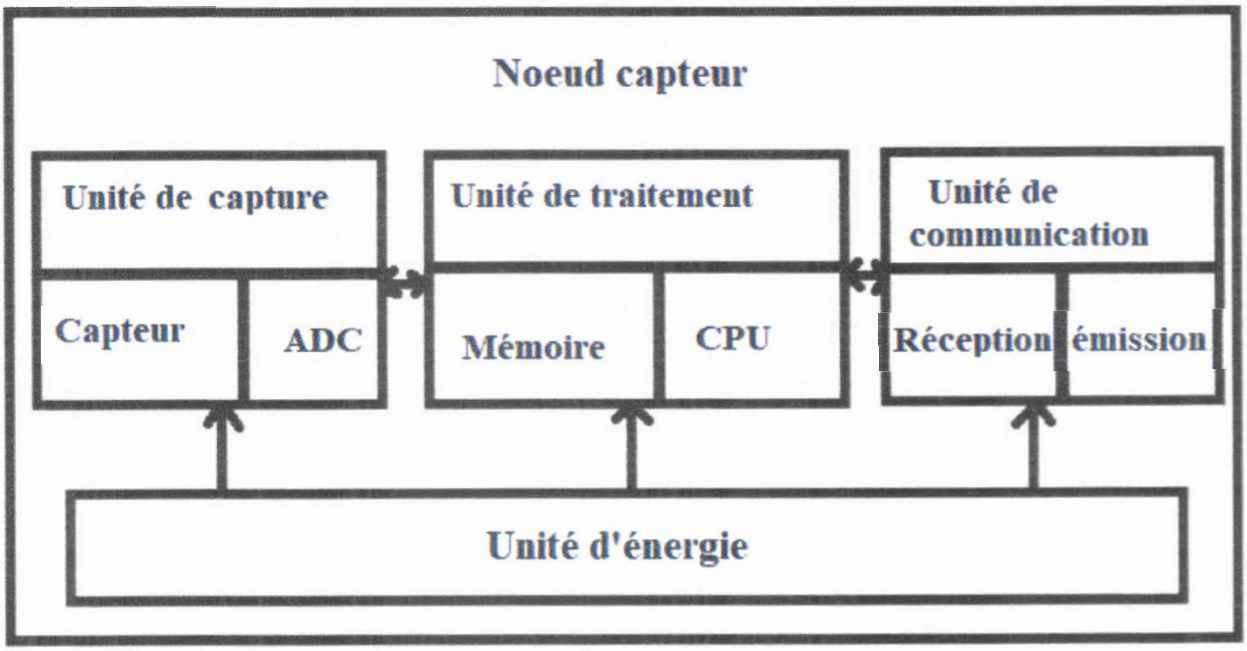


Figure 2: Architecture d'un nœud capteur

1.2.3.1.Unité de traitement

Mote, processeur, RAM et Flash : On appelle généralement Mote la carte physique utilisant le système d'exploitation pour fonctionner. Celle-ci a pour cœur le bloc constitué du processeur et des mémoires RAM et Flash. Cet ensemble est à la base du calcul binaire et du stockage, temporaire pour les données et définitif pour le système d'exploitation. Cette unité est chargée d'exécuter les protocoles de communications qui permettent de faire collaborer le nœud avec les autres nœuds du réseau. Elle peut aussi analyser les données captées pour alléger la tâche du nœud puits [5].

1.2.3.2 Unité de capture

On trouve des équipements de différents types de détecteur et d'autre entrée. Le capteur est généralement composé de deux sous-unités : le récepteur (reconnaissant l'analyse) et le transducteur (convertissant le signal du récepteur en signal électrique). Le capteur est responsable de fournir des signaux analogiques, basés sur le phénomène observé, au convertisseur Analogique/Numérique. Ce dernier transforme ces signaux en un signal numérique compréhensible par l'unité de traitement [3].

1.2.3.3 L'unité de communication

Elle connecte le nœud à l'ensemble du réseau. Elle est responsable d'effectuer toutes les émissions et réceptions de données sur un médium sans fil, afin de permettre l'échange d'informations entre le nœud capteur et son environnement extérieur. En fonction du domaine

d'application, trois modes de communication sans fil peuvent être utilisés : optique, infrarouge, radiofréquence [6].

1.2.3.4 Unité d'énergie

C'est la batterie qui, n'est généralement ni rechargeable ni remplaçable. La capacité d'énergie limitée au niveau des capteurs représente la contrainte principale lors de conception de protocoles pour les réseaux de capteurs. Les unités d'énergie peuvent être supportées par des photopiles qui permettent de convertir l'énergie lumineuse en courant électrique [7].

1.2.4 Architecture d'un RCSF

Un RCSF est un ensemble de nœuds capteurs déployés dans des champs de capteurs qui [8] dispersés a la capacité de collecter des données et envoie des requêtes ou des commandes aux noeuds capteurs dans la région de détection. Les nœuds capteurs collaborent pour accomplir la tâche de détection et rediriger les données collectées vers la station de base [9], et en même temps la station de base servent également de passerelle vers les utilisateurs finaux. Elle collecte les données saisies via les nœuds capteurs, puis les traite et envoie les informations nécessaires à l'utilisateur final.

Pour envoyer les données collectées, chaque capteur peut utiliser une liaison longue distance à saut unique. Ce qui conduit à l'architecture de réseau à un seul saut [9], ou par une communication multi-sauts, le capteur transmet ses données via un ou plusieurs nœuds intermédiaires. Cette dernière option permet de réduire la consommation d'énergie et d'augmenter considérablement la durée de vie du réseau.

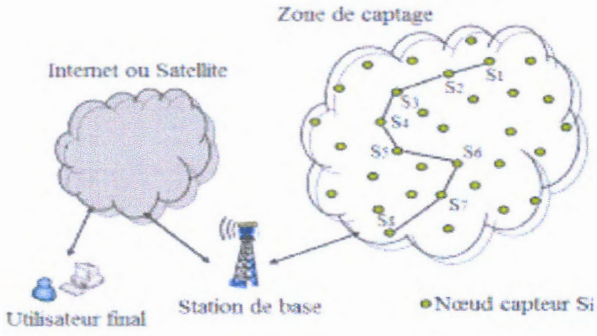


Figure 3 : Architecture d'un RCSF

Dans l'exemple illustré par la *Figure 3*, les nœuds capteurs sont déployés de manière aléatoire ou déterministe dans une zone d'intérêt donnée (zone de captage). Une station de base (BS) située en dehors de cette zone d'intérêt est chargée de récupérer les données collectées et envoyées par les différents nœuds capteurs. Ainsi, lorsqu'un nœud capteur donné détecte un

événement pertinent, il capture une mesure sur cet événement et l'envoie à la BS par le biais d'un routage multi-sauts.

1.2.5 Caractéristiques des réseaux de capteurs sans fil

Les RCSFs sont souvent caractérisés par un déploiement dense et à grande échelle dans des environnements distants, dangereux et inaccessibles. Nous citons quelques caractéristiques de ces résaux [2] :

1.2.5.1 Forte densité des nœuds

Les RCSFs peuvent contenir un très grand nombre de capteurs en raison de leurs petites tailles et s'adaptent à tous les environnements où la densité de capteurs peut aller jusqu'à 20 capteurs/m^3.

1.2.5.2 Ressources limitées

Les capteurs sont généralement faibles en termes de ressources et limités en termes de bande passante, puissance de calcul, de la mémoire et de l'énergie disponible.

1.2.5.3 Contrainte d'énergie

L'énergie est une contrainte clé dans les réseaux de capteurs dont la durée de vie des capteurs en dépend. Il est nécessaire de s'assurer que la distribution d'énergie consommée soit équitable au sein du réseau afin de prolonger la durée de vie aussi longtemps que possible.

1.2.5.4 Déploiement de nœuds

Comme les autres réseaux sans fil, les réseaux de capteurs sont indépendants de l'infrastructure physique, de sorte que l'emplacement des capteurs est associé à l'objectif du déploiement du réseau. Où ils sont placés manuellement en cas de besoin ou de manière aléatoire dans des environnements extrêmes ou difficiles.

1.2.5.5 Agrégation de données

Dans les réseaux de capteurs sans fil, en raison de la densité des noeuds, les données collectées par les capteurs dans une zone donnée sont redondantes et superflues et ont la même signification : elles peuvent donc être collectées et agrégées dans les noeuds intermédiaires avant leur transmission, ce qui contribue à réduire la consommation d'énergie pendant le transfert des données.

1.2.5.6 Médias de transmission

Dans un réseau de capteurs, les nœuds sont reliés par une architecture sans fil. Pour permettre des opérations sur ces réseaux dans le monde entier, le média de transmission doit etre standardisé. On utilise le plus souvent l'infrarouge, le Bluetooth , l'optique et les communications radio [10].

1.2.5.7 Couverture limitée

Dans les réseaux de capteurs, chaque nœud a une vision locale relativement limitée de son environnement, limitée par sa portée et sa précision. La couverture d'une grande surface nécessite l'union de nombreuses petites couvertures.

1.2.5.8 Auto organisation du réseau et la topologie dynamique

Ceci peut être nécessaire, vue le grand nombre de nœuds de micro-capteurs et leurs placements dans des endroits hostiles, où l'intervention humaine n'est pas faisable. D'ailleurs, les nœuds peuvent échouer (par manque d'énergie où destruction physique), comme de nouveaux nœuds peuvent rejoindre le réseau. Par conséquent, le réseau doit être capable de s'auto-organiser et de s'adapter périodiquement de sorte qu'il puisse continuer à fonctionner [11]

1.2.6 Classification des réseaux de capteurs sans fil

Les RCSFs peuvent être classés selon plusieurs critères comme l'homogénieté, la mobilité, le mode de fonctionnement, ...

1.2.6.1 Réseaux de capteurs homogènes vs hétérogènes

Les réseaux de capteurs sont divisés en deux catégories en fonction du type de capteurs qui les compose : réseaux homogènes et réseaux hétérogènes [12].

Dans les réseaux de capteurs homogènes, tous les nœuds sont identiques en termes de contraintes d'alimentation de la batterie, de détection et de ressources de communication.

Par contre, dans un réseau de capteurs hétérogènes certains, nœuds peuvent avoir plus de capacité de batterie, de capacité de traitement et certaines données peuvent être collectées et agrégées au niveau de ces nœuds. Le déploiement de ce type de réseau est relativement complexe et son application est limitée, car elle nécessite une distribution en douceur de différents types de nœuds dans des emplacements spécifiques. Ces réseaux ont fait leurs preuves dans le cadre de déploiements réels en raison de leur capacité à augmenter la durée de vie du réseau sans augmentation significative du coût [13].

1.2.6.2 Réseaux de capteurs fixes vs mobiles

Dans les réseaux de capteurs fixes, les capteurs restent immobiles, ils ne se déplacent pas. Ils sont utilisés par exemple pour surveiller les séismes ou les changements de température.

Dans les réseaux de capteurs mobiles, les nœuds de capteurs ou les phénomènes observés se déplacent, par exemple, dans la zone surveillée. Les capteurs mobiles peuvent améliorer la sécurité du réseau et la couverture faible du réseau, mais souffrent de certaines exigences d'alimentation et de la reconfiguration du réseau.

1.2.6.3 Réseaux de capteurs basés sur des événements ou basés sur des requêtes

Dans les applications de réseaux de capteurs basées sur des événements, telles que la détection des incendies de forêt, un ou plusieurs capteurs détectent un événement et le signalent à une station de base ou à une station de surveillance [12].

Cependant, dans les réseaux de capteurs basés sur des requêtes tels que le suivi des stocks dans des entrepôts d'une usine, les capteurs restent silencieux jusqu'à ce qu'ils reçoivent une demande de la station de surveillance [12].

1.2.6.4 Réseaux de capteurs à structure plate vs structure hiérarchique

Il existe deux structures de réseau qui détermine la manière dont les capteurs sont assemblés et comment les informations des capteurs sont acheminées sur le réseau : une structure plate et une structure hiérarchique.

Avec une structure plate, un grand nombre de nœuds, en général, homogènes sont déployés dans la zone de routage et les données sont acheminées par les nœuds un à un vers la station de base.

Dans les réseaux de capteurs hiérarchiques, les nœuds sont organisés en groupes (clusters) et pour chaque groupe un nœud est selectionné en tant que tête de groupe. Les nœuds membres d'un groupe envoient les données capturées vers la tête de groupe (Cluster-Head), qui collecte les données et les envoit à la station de base.

1.2.7 Domaines d'applications des réseaux de capteurs sans fil

Les réseaux de capteurs sans fil sont l'une des technologies les plus importantes de ce ciècle et sont utilisés dans de nombreux domaines. Ils se caractérisent par une modification de la structure du réseau en fonction du domaine d'application. Dans cette section, nous discutons des domaines d'utilisation les plus importants des RCSFs.

1.2.7.1 Applications militaires

Le domaine militaire a été une motivation majeure pour le développement de réseaux de capteurs en raison des différents avantages apportés par ces réseaux : autorégulation, tolérance aux fautes, faible coût des équipements, étendue sur une vaste zone, ce qui le rend très utile dans le domaine militaire et permet de surveiller tous les mouvements de l'ennemi et d'analyser le terrain et les mines ...

La *Figure 4* suivant montre un exemple d'application des réseaux de capteurs sans fil dans le domaine militaire.

Figure 4 : Les RCSFs dans le domaine militaire

1.2.7.2 Applications médicales

Le domaine de la médecine est devenu plus sophistiqué grâce aux capteurs sensibles qui peuvent maintenant être avalés ou implantés sous la peau, permettant ainsi une surveillance continue des patients et la possibilité de collecter des informations et des caractéristiques physiologiques, facilitant ainsi le diagnostic de certaines maladies [4]. Les capsules de capture ont permis de surveiller de l'intérieur les fonctions vitales des êtres humains et de réaliser une imagerie sans recourir à la chirurgie. La *Figure 5* montre un exemple d'application des réseaux de capteurs sans fil dans le domaine médicale.

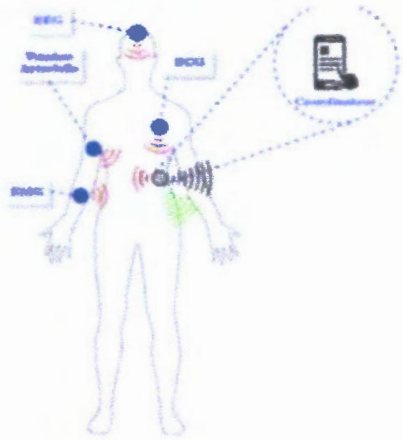


Figure 5 : Les RCSFs pour les applications médicales.

1.2.7.3 Applications de surveillance

Les réseaux de capteurs sans fil permettent d'étendre la surveillance en termes de sécurisation des lieux et des personnes, tout en permettant de suivre les structures d'aéronefs, les navires, les voitures, le métro et les lieux publics en temps réel. De même que les réseaux de

distribution d'énergie ou les réseaux de distribution pouvant être surveillés et contrôlés en temps réel par les RCSFs. De plus, l'intégration de capteurs dans de grandes structures telles que des ponts, des routes, des plates-formes, des voies ferrées et des bâtiments peut aider à détecter toute fissure ou modification de la structure. En cas de séisme ou de vieillissement de ces structures, un réseau de capteurs de mouvement peut être un système d'alarme distribué utilisé pour détecter des anomalies [12].

1.2.7.4 Applications environnementales

Les réseaux de capteurs peuvent être utilisés dans les champs agricoles, sur les sites industriels, dans les centrales nucléaires, dans les champs pétroliers, dans les forêts ou pour la surveillance de l'environnement marin [4], par exemple, le déploiement de capteurs de température peut aider, dans la forêt, à la découverte d'un potentiel début d'incendie ou le déploiement de capteurs chimiques en milieu urbain peut aider à la détection de la pollution et à l'analyse de la qualité de l'air [12].

1.2.7.5 Applications commerciales

Des RCSFs peuvent être utilisés dans le domaine commercial afin d'aider les commerçants à améliorer le processus de stockage et la livraison des marchandises. Pour les entreprises de production les RCSFs permettent de suivre le procédé de production à partir des matières premières jusqu'au produit final [14].

1.3 Consommation et conservation d'énergie dans les RCSFs

1.3.1 Introduction

La durée de vie est probablement la mesure la plus importante pour évaluer les performances des réseaux sans fil. L'âge du réseau est principalement lié à la batterie du capteur. Les capteurs sont conçus pour fonctionner pendant des mois, voire des années [15], dans des environnements difficiles avec des ressources limitées. A noter qu'une fois un nœud capteur a épuisé son énergie, il est considéré comme défaillant. Ainsi il y a une forte probabilité de perdre la connectivité du réseau [7] et les économies d'énergie sont l'un des principaux défis des capteurs car la recharge des sources d'énergie est souvent trop coûteuse et parfois impossible ; augmenter la durée de vie du réseau est associé à une consommation d'énergie réduite des capteurs, et toute ressource limitée doit être prise en compte. Malgré les progrès réalisés, la durée de vie de ces dispositifs reste un défi majeur et un facteur clé qui nécessite des recherches supplémentaires sur l'efficacité énergétique et les protocoles de communication dans les réseaux de capteurs sans fil.

1.3.2 La durée de vie d'un réseau de capteurs

La durée de vie du réseau de capteurs correspond à la période de temps pendant laquelle le réseau peut maintenir une communication adéquate ou maintenir le taux de perte d'informations en dessous d'un certain seuil. L'épuisemet de l'énergie d'un certain pourcentage de nœuds entraîne la non-couverture de la zone et/ou d'une partie du réseau [4]. Par conséquent, la durée de vie du système est liée à la durée de vie des capteurs et la durée de vie du nœud dépend principalement de deux facteurs fondamentaux : L'énergie qu'il consomme en fonction du temps et de la quantité d'énergie disponible.

Voici quelques-unes des définitions proposées dans la littérature [6] :

- Durée jusqu'à ce que le premier nœud épuise toutes ses énergies.
- Durée jusqu'à ce qu'un certain pourcentage de capteurs épuisent leurs énergies.
- Durée jusqu'à ce que tous les capteurs épuisent leurs énergies.

1.3.3 Consommation d'énergie dans les RCSFs

Pour identifier les problèmes du système d'alimentation, nous analysons les caractéristiques de consommation d'énergie du nœud de capteur sans fil. Cette analyse systématique de l'énergie des capteurs est extrêmement importante pour permettre une amélioration efficace. La puissance consommée par le capteur provient principalement des opérations suivantes : capture, traitement et transmission de données.

1.3.3.1 Énergie de capture

La capture est effectuée par les composants d'acquisition qui traduisent les phénomènes physique en signal électrique. La consommation d'énergie du module de détection dépend de la spécificité du capteur. Dans de nombreux cas, elle est négligeable par rapport à l'énergie consommée par les modules de traitement et communication. L'énergie consommée lors de la capture peut être réduite en utilisant des composants à faible consommation d'énergie. Une autre façon de réduire l'énergie consommée lors de la capture consiste à réduire la durée de capture et supprimer les captures jugées redondantes et inutiles [2].

1.3.3.2 Energie de traitement

L'énergie consommée durant le traitement des données peut être divisée en deux parties : l'énergie de commutation et l'énergie de fuite. L'énergie de commutation est déterminée par la tension d'alimentation et la capacité totale commutée au niveau logiciel (en exécutant un programme). Par contre, l'énergie de fuite correspond à l'énergie consommée lorsque l'unité de

calcul n'effectue aucun traitement. En général, l'énergie de traitement est faible par rapport à celle nécessaire pour la communication [2].

1.3.3.3 Energie de communication

L'énergie de communication (réception / transmission) représente la plus grande proportion de l'énergie totale consommée par un capteur [4]. L'énergie de communication est divisée en deux parties : la puissance de réception et l'énergie d'émission. Cette énergie est déterminée par la quantité de données transmises et la distance de transmission, ainsi que par les caractéristiques physiques de l'unité radio [7]. Lorsque la puissance d'émission est élevée, le signal aura une grande portée et l'énergie consommée sera plus élevée.

1.3.4 Facteurs intervenants dans la consommation d'énergie

La consommation d'énergie dépend de plusieurs facteurs qui sont :

1.3.4.1 Etat du module radio

Pour assurer la communication entre les éléments du réseau, les capteurs utilisent leurs unités radio [7]. Cette unité est responsable de la consommation d'une grande partie de l'énergie des capteurs. La radio fonctionne en quatre modes : actif, envoi, réception et veille.

- Mode actif : le nœud ne reçoit ni ne transmet. Cette condition entraîne une perte de puissance due à une écoute inutile du canal de transmission,
- Mode envoi : la radio transmet le paquet.
- Mode réception : la radio reçoit un paquet.
- Mode veille : la radio est éteinte.

Il est aussi à noter que le passage fréquent de l'état actif à l'état veille peut avoir comme conséquence une consommation d'énergie plus importante que de laisser le module radio en mode actif. Ceci est dû à la puissance nécessaire pour la mise sous tension du module radio. Cette énergie est appelée l'énergie de transition [7].

1.3.4.2 Accès au medium de transmission

La sous couche MAC assure l'accès au support de transmission, la fiabilité de transmission, le contrôle de flux, la détection d'erreurs et la retransmission des paquets. Puisque les nœuds partagent le même médium de transmission, la sous-couche MAC joue un rôle important pour la coordination entre les nœuds et la minimisation de la consommation d'énergie. En effet, minimiser les collisions entre les nœuds permet de réduire la perte d'énergie. Dans ce qui suit, nous analyserons les principales causes de consummation d'énergie au niveau de la couche MAC [17]:

a) La retransmission

Les capteurs partagent, généralement, le même canal de transmission, la transmission simultanée de données entraîne de multiples collisions entre les paquets envoyés, entraînant la perte de certaines informations transmises. Le renvoi de paquets perdus peut entraîner une perte de puissance importante.

b) L'écoute active

L'écoute active (idle listening) du canal pour une éventuelle réception de paquet qui ne sera pas reçu peut engendrer une perte importante de la capacité des nœuds en énergie. Pour éviter ce problème, il faut basculer les nœuds dans le mode veille le plus longtemps possible.

c) La surécoute

Une surécoute se produit lorsque le nœud reçoit des paquets qui ne sont pas lui destinés. Le coût d'une surécoute peut être élevé si l'opération est répétée plusieurs fois dans le cas d'un réseau dense et d'une charge importante.

La *Figure 6* montre un exemple de surécoute, lorsque le message est envoyé à **B** mais il atteint également **C**. Il en résulte une consommation d'énergie en **C**, même si ce n'est pas un message important en **C**.

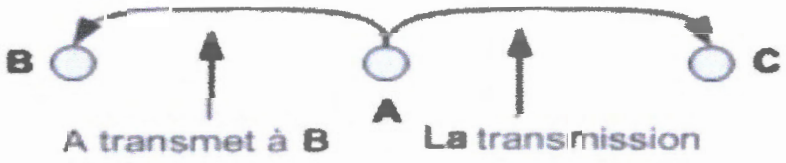


Figure 6 : La surécoute dans une transmission

d) La surcharge

Plusieurs protocoles de la couche MAC fonctionnent par échange de messages de contrôle pour maintenir une bonne communication entre les nœuds afin d'assurer différentes fonctionnalités : signalisation, connectivité, établissement de plan d'accès et évitement de collisions. Tous ces messages nécessitent une énergie additionnelle.

e) La surémission

Le phénomène de diffusion excessive se produit lorsque les nœuds envoient des messages à des destinations qui ne sont pas prêtes à les recevoir. En effet, ces messages sont considérés comme inutiles et consomment beaucoup d'énergie.

f) La taille des paquets

La taille des paquets a un impact sur la consommation d'énergie. En fait, si la taille des paquets est trop petite, le nombre de paquets augmente, ce qui augmente la charge. Cependant, si la taille du paquet est importante, elle nécessite l'utilisation d'une grande capacité de transport et consomme donc plus d'énergie.

1.3.4.3 Modèle de propagation radio

Le modèle de propagation représente une estimation de la puissance moyenne reçue du signal radio à une distance donnée d'un émetteur. La propagation du signal radio est généralement soumise à différents phénomènes : la réflexion, la diffraction et la dispersion par divers objets. Généralement, la puissance du signal reçue est de l'ordre de $\mathbf{1/d^n}$, où **d** est la distance entre l'émetteur et le récepteur, **n** un exposant de perte d'un chemin (Exemple : n = 2 dans le vide, de 4 à 6 dans un immeuble) [17].

1.3.4.4 Routage des données

Le routage des données peut avoir un impact sur la consommation d'énergie dans les réseaux de capteurs à sauts multiples. Lorsque des paquets sont acheminés de la source à la station de base via plusieurs capteurs intermédiaires, ce qui leur fait consommer de l'énergie, soit pour transmettre leurs données, soit pour transmettre les données des autres capteurs, et doit choisir une politique de routage appropriée et efficace pour réduire la consommation d'énergie.

1.3.5 Techniques de minimisation de la consommation d'énergie

Après avoir décrit les principales raisons de la consommation d'énergie dans le réseau [7], dans cette section, nous allons essayer de présenter quelques mécanismes importants pour réduire la consommation d'énergie.

Afin de prolonger la vie du réseau le plus longtemps possible, les mesures expérimentales montrent que le transfert de données consomme le plus d'énergie par rapport aux calculs et à l'unité de détection (capture). Dans de nombreux cas, l'unité de détection n'est guère négligeable par rapport à l'énergie consommée par l'unité de communication. Dans d'autres cas, l'énergie consommée pour la détection peut être comparable à l'énergie nécessaire pour transmettre les données.

Le schéma de la *Figure 7* donne un aperçu global de ces mécanismes qui visent à minimiser la consommation d'énergie dans les réseaux de capteurs sans fil :

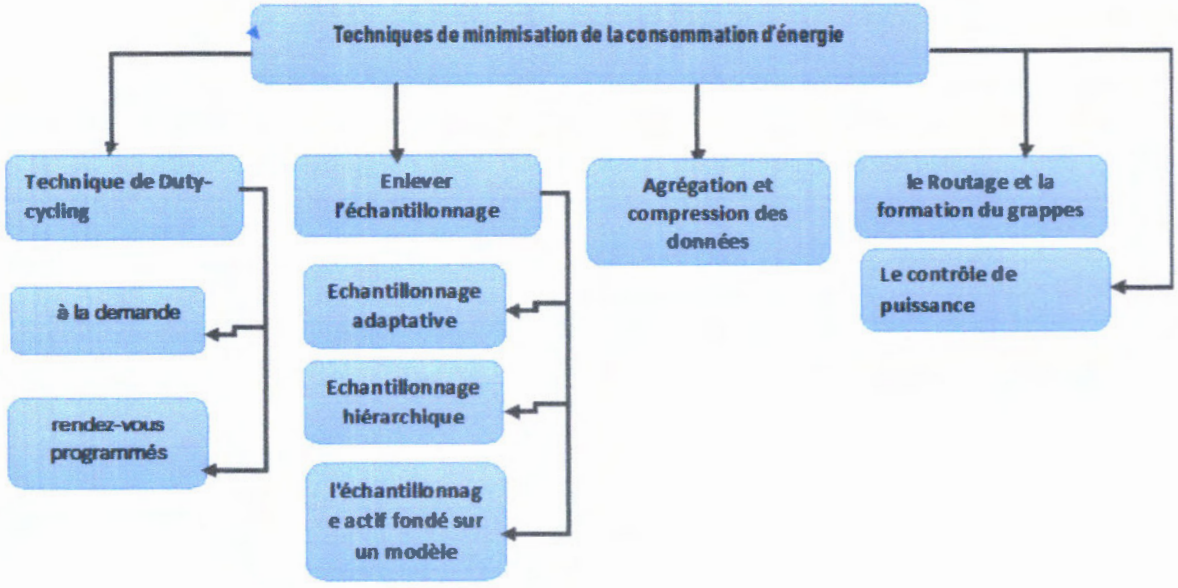


Figure 7 : Techniques de minimisation de la consommation d'énergie

1.3.5.1 Technique de Duty-cycling

Cette technique est un moyen très efficace d'économiser de l'énergie. Elle consiste à mettre la transmission radio en mode basse consommation lorsque la communication n'est pas nécessaire. Idéalement, la radio doit être éteinte lorsqu'il n'y a plus de données à envoyer ou à recevoir, et être prête lorsque le capteur a besoin d'envoyez ou de recevoir des données. Ainsi, le capteur alterne entre des périodes actives et des périodes de repos en fonction de l'activité du réseau. Ce comportement est généralement appelé devoir cycliste. Le cycle est défini comme faisant partie du temps pendant lequel les nœuds sont actifs. Comme indiqué précédemment, le système veille/réveil d'un composant particulier (l'unité radio) peut être défini dans le nœud capteur [18].

Les protocoles de veille/réveil appartiennent à deux catégories principales :

a) **à la demande**

Les protocoles à la demande utilisent l'approche la plus intuitive pour la gestion d'énergie. L'idée de base est qu'un noeud devrait se réveiller seulement quand un autre noeud veut communiquer avec lui. Le problème principal associé aux régimes à la demande est de savoir comment informer un noeud en sommeil qu'un autre noeud est disposé à communiquer avec lui. À cet effet, ces systèmes utilisent généralement plusieurs radios avec différents compromis entre énergie et performances (i.e. une radio à faible débit et à faible consommation pour la signalisation, et une radio à haut débit mais à plus forte consommation pour la communication de données). Le protocole STEM (Sparse Topology and Energy Management) par exemple utilise deux radios.

b) rendez-vous programmés

La deuxième catégorie dépend de l'utilisation d'une approche par date spécifique. L'idée est que chaque nœud doit se réveiller en même temps avec ses voisins. Habituellement, les nœuds se réveillent à un horaire de réveil et restent actifs pendant un court instant pour communiquer avec leurs voisins. Ensuite, ils reviennent au prochain rendez-vous.

1.3.5.2 Enlever l'échantillonnage inutile

Vu la densité du réseau de capteurs, les données échantillonnées ont souvent de fortes corrélations spatiales et/ou temporelles. Il est donc inutile de communiquer les informations redondantes à la station de base. En fait, même si le coût de la détection est négligeable, la communication des données redondantes implique une surconsommation d'énergie. Il existe trois techniques principales pour enlever l'échantillonnage inutile :

a) Echantillonnage adaptative

Les approches proposées pour l'échantillonnage adaptatif exploitent les corrélations spatiales et temporelles des données pour réduire la quantité de données à acquérir par le module de détection.

b) Echantillonnage hiérarchique

Le but de ce type d'approches consiste à trouver un compromis entre la consommation d'énergie et la précision. Étant donné que les capteurs qui donnent de très bonne précision sont gourmant en énergie, l'idée est d'utiliser différents types de capteurs avec des caractéristiques de puissance différentes. Où de simples capteurs sont utilisés pour détecter des données suspectes. Plus précisément, des capteurs plus puissants sont utilisés. Pour un échantillonnage en série précis, cette technique peut associer conservation de l'énergie et precision.

c) l'échantillonnage actif fondé sur un modèle

Un modèle du phénomène mesuré est construit sur la base de données échantillées. Ce modèle est ensuite utilisé pour prédire les données futures avec un niveau élevé de précision, ce qui permet au capteur de saisir la relation spatio-temporelle entre ses mesures et peut ainsi limiter et réduire le nombre d'échantillons prélevés du modèle construit.

1.3.5.3 Agrégation et compression des données

Les nœuds voisins dans les réseaux de capteurs, étant donné qu'ils sont très corrélés spatialement et temporellement, peuvent générer les mêmes données qui seront ensuite transférées à la station de base, ce qui implique l'existence de redondances de données. La réduction des données en termes de taille ou de nombre de paquets dans les réseaux peut avoir un impact

significatif sur la consommation d'énergie due aux communications. L'un des moyens de réduire les données est le traitement en réseau, qui consiste à implémenter l'agrégation de données (par exemple, calculer la moyenne de certaines valeurs) dans le capteur intermédiaire entre la source et la station de base. Ainsi, la quantité de données est réduite pendant le routage du capteur vers la station de base. La compression des données peut également être appliquée pour réduire la quantité d'informations envoyées par les nœuds sources. Ce schéma implique le cryptage des informations dans les nœuds générant des données, et décryptage au niveau de la station de base. Il y a différentes façons de compresser les données [1].

1.3.5.4 Le routage et la formation de clusters (clustering)

Les algorithmes de regroupement (clustering) organisent le réseau dans des structures à plusieurs niveaux. Les nœuds sont regroupés en groupes contrôlés par la tête du cluster (CH) qui est responsable du transfert des données de chaque capteur à la station de base . Les algorithmes de clustering permettent l'organisation du réseau en sous-réseaux (groupes) plus homogènes selon une métrique ou une combinaison de métriques, formant ainsi une topologie virtuelle. Chaque cluster identifie un nœud particulier appelé cluster-head. Le cluster-head, assure la coordination entre les nœuds membres de son cluster, d'agréger leurs données collectées et de les transmettre par la suite à la station de base. De ce fait, seulement les CHs seront responsables de l'acheminement de l'information collectée vers la station de base. Minimisant ainsi l'énergie consommée par les nœuds capteurs. Il existe plusieurs méthodes de formation de clusters. La méthode la plus répandue est LEACH. D'une part, les têtes de clusters changent dans le temps selon un certain algorithme lié à la quantité d'énergie restante pour maintenir l'équilibre énergétique du réseau et utiliser des protocoles pour acheminer les données entre les têtes de clusters via un chemin à sauts multiples afin de réduire le coût du transport des données et de réaliser une économie d'énergie.

1.3.5.5Le contrôle de puissance

Les techniques de contrôle de puissance visent essentiellement à optimiser la puissance utilisée de la radio ; elles permettent aux nœuds d'ajuster leurs puissances radio dans le but de minimiser la consommation d'énergie. Les techniques de contrôle de puissance n'ont pas seulement un effet sur la consommation énergétique des nœuds capteurs mais aussi influent la connectivité du réseau. En outre, ils doivent assurer une transmission avec succès des paquets de données vers une destination. Les protocoles ATPC (Adaptive Transmission Power for Wireless

Sensor Networks) LMA et LMN (Local Mean Algorithm and Local Mean of Neighbors Algorithm) font partie de cette catégorie [19].

1.4 Conclusion

La durée de vie d'un réseau de capteurs est étroitement liée à la vie d'un nœud. Cette dernière, quant à elle, dépend essentiellement de la consommation d'énergie du nœud. Dans ce chapitre, nous avons étudié les réseaux de capteurs sans fil en général, leur utilisation et le problème de l'énergie limitée, ainsi que certaines techniques permettant de maximiser la durée de vie du réseau grâce à une consommation efficace et d'éviter toute surconsommation d'énergie en réduisant la durée de l'activité sans fil du capture et en limitant les communications parce que les communications sont responsables de la consommation d'énergie la plus importante. Le routage hiérarchique est l'une des technologies d'économie d'énergie les plus importantes dans les réseaux de capteurs sans fil. c'est la meilleure solution pour réduire la consommation d'énergie et une approche efficace pour augmenter la durée de vie du réseau.

Dans le chapitre suivant, nous nous intéressons aux méthodes de routage hiérarchique et aux techniques de clustering dans les réseaux de capteurs sans fil.

Chapitre 2 : Etude des protocoles de routage hiérarchique et des techniques de clustering dans les réseaux de capteurs sans fil (RCSF)

2.1 Inroduction

Les réseaux de capteurs sans fil (RCSFs) sont des réseaux sans infrastructure, composés de plusieurs capteurs autonomes répartis sur une vaste zone, avec une station de base puissante (BS) collectant des informations à partir de ces nœuds capteurs. Comme nous l'avons mentionné précédemment, ces capteurs souffrent d'un manque de ressources, les contraintes énergétiques restent les plus importantes car elles affectent directement la vie du réseau. Le routage est une solution importante pour améliorer la durée de vie du réseau de capteurs. Les protocoles de routage dans les RCSFs peuvent être divisés en deux catégories : routage à plat et routage hiérarchique. Dans une topologie plate, tous les nœuds exécutent les mêmes tâches et ont la même configuration [21]. L'acheminement des paquets de la part d'un nœud dépend de sa position dans le réseau. Afin d'augmenter la scalabilité (Facteur d'échelle) du système et réduire la consommation d'énergie, les topologies hiérarchiques ont été introduites en divisant les noeuds en plusieurs niveaux de responsabilité. L'une des méthodes les plus employées est le clustering, où le réseau est partitionné en groupes appelés "clusters". Un cluster est constitué d'un chef (cluster-head) et des nœuds membres. La mise en cluster augmente l'efficacité de la transmission de données en réduisant le nombre de capteurs essayant de transmettre des données à la station de base. L'agrégation des données au niveau des CHs via la communication intra-cluster contribue également à éliminer la duplication des données [21]. Le routage hiérarchique ou le routage par groupement (cluster-based), est une technique bien connue avec des fonctionnalités qui résolvent les problèmes liés à la surcharge de la station de base due à la densité du réseau. Les protocoles du routage hiérarchiques sont chargés, généralement, de confier des rôles à des nœuds du réseau, d'établir des groupes de capteurs (clusters), et de définir la manière comment les noeuds choisissent les chefs de groupe (cluster-head). Les noeuds choisis comme étant des cluster-head sont des noeuds à énergie élevée. Ils peuvent être utilisés pour traiter et envoyer l'information. Les noeuds à faible énergie peuvent être employés pour exécuter la tâche de la capture à proximité de la cible [23].

Dans ce chapitre, nous nous intéressons aux approches de routage hiérarchique et nous étudions les techniques de clustering dans les réseaux de capteurs sans fil. Dans la première partie, nous discuterons des caractéristiques, objectifs et des avantages du clustering et nous parlerons dans la seconde partie de certaines techniques de routage hiérarchique.

2.2 Généralités sur les protocoles de routage hiérarchique

2.2.1 Définitions

a) Clustering

Le clustering est une technique pour partitionner le réseau en groupes (Clusters), sachant que pour chaque groupe est désigné un leader (CH), ce dernier communique avec les membres de son groupe et les Cluster Heads des autres groupes. De cette manière, l'opération de clustering contribue considérablement à l'économie de l'énergie, à la réduction de la complexité des protocoles de routage, et à la résistance au facteur d'échelle, en plus de l'agrégation de données qui permet d'éliminer la redondance de données et de n'envoyer que les informations utiles [24].

b) Cluster

Le cluster est un ensemble de nœud, qui forme l'unité d'organisation d'un réseau de capteurs, la nature dense de ces réseaux exige la décomposition en cellules afin de simplifier les tâches de communication et répondre aux difiérentes contraintes [24].

c) Cluster Head

La tête de groupe (CH) est le leader d'un groupe. Les CHs sont souvent tenus d'organiser des activités dans leurs clusters. Ces tâches incluent l'agrégation de données, l'organisation de la communication inter-cluster et intra-cluster. Le chef de cluster est choisi pour jouer ce rôle soit par les autres noeuds ou bien pré-assignés par le concepteur de réseau, il peut être ordinaire comme les autres noeuds ou bien doté de plus d'énergie.

d) Station de Base

La station de base (BS) se situe au niveau supérieur de la hiérarchique d'un réseau de capteurs et assure la liaison de communication entre le réseau de capteurs et l'utilisateur final.

La *Figure 8* montre un RCSFs hiérarchique, où certains capteurs sont déclarés chef de leur cluster (CH). Il est responsable du traitement des données capturées à partir des nœuds membres et de la gestion du routage vers les stations de base.

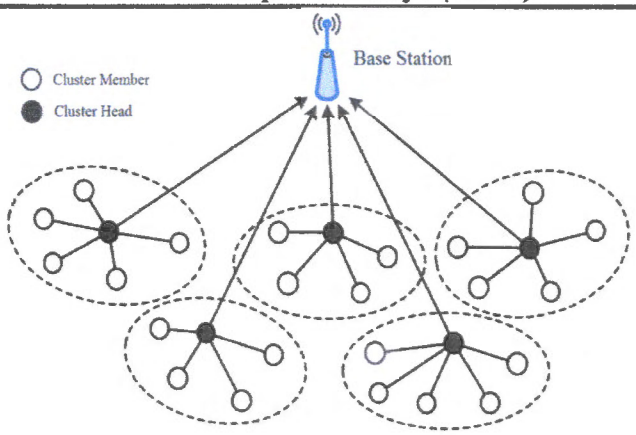


Figure 8 : Hiérarchie d'un réseau de capteurs sans fil.

2.2.2 Caractéristiques de Clustering

2.2.2.1. L'algorithme de clustering utilisé

Plusieurs algorithmes sont proposés dans la littérature , il existe deux types :

•Centralisé :

L'algorithme est exécuté sur le nœud qui a une vue globale du réseau, généralement, au niveau de la station de base. Ce type d'algorithme est peu utilisé à cause de la surcharge (overhead) généré suite aux transmissions exécutées pour pouvoir garder la vue globale du réseau et la dynamique de la topologie qui fait que cette vue soit très variable [24].

•Distribué :

L'algorithme est exécuté en coopération au niveau de chaque nœud du réseau. La synchronisation des tâches de contrôle est obtenue en échangeant des messages de contrôle. Ce type d'algorithmes minimise la communication relative à la sauvegarde de la vue globale du réseau, car chaque nœud décide, indépendamment des autres, de son rôle de faire connaître sa décision par l'envoi de message. Cependant, l'efficacité de ces algorithmes dépend de la taille et du nombre de ces messages de synchronisation [24].

2.2.2.2 L' élection des Cluster Heads

Le nœud Cluster-Head consomme plus d'énergie que les autres nœuds du réseau. Le Cluster Head coordonne le fonctionnement des nœuds membres de son cluster et agrège leurs données, de ce fait, il dissipe plus d'énergie créant un déséquilibre de la distribution de l'énergie sur le réseau. Pour pallier à ce problème, une rotation de ce rôle de Cluster-Head est organisée au sein du cluster ou bien au sein du réseau entier. La rotation est effectuée périodiquement ou bien en fonction de la consommation de l'énergie du nœud Cluster Head [25].

2.2.2.3 Communication intra-cluster

La communication entre noeuds Cluster-Head et les autres noeuds membres du cluster peut se faire, soit en un seul saut soit, en plusieurs sauts. Dans le cas d'une communication directe (en un seul saut), les paquets de données sont envoyés directement au Cluster-Head. Cela suppose que les noeuds membres soient capables d'atteindre le CH en utilisant une transmission assez puissante pour une bonne réception de données. Ce type de communications engendre une consommation importante d'énergie si la distance entre le CH et les autres noeuds est grande. Pour réduire la consommation de l'énergie, une communication en plusieurs sauts, de petites distances, est utilisée, dans ce cas chaque membre du cluster envoie ses données au plus proche membre de son cluster jusqu'à l'aboutissement au CH. Ce type de communication est souvent utilisé pour réduire le nombre de collisions.

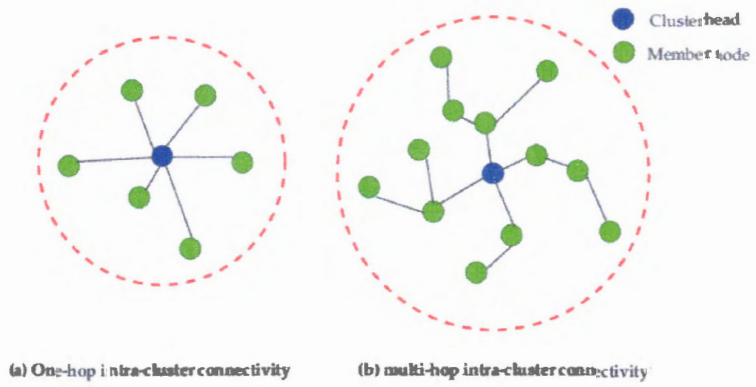


Figure 9 : Communication intra-cluster dans RCSF.

La Figure 9 montre la nature de la communication au sein du cluster, où dans (a) la communication par un seul saut, dans (b) la communication par un ou plusieurs sauts.

2.2.2.4 Communication inter-cluster

Les Cluster-Heads communiquent avec la station de base soit directement, soit en deux ou plusieurs sauts via des nœuds appelés généralement des " Nœuds intermédiares". Ces nœuds peuvent être des CHs ou bien des nœuds membres d'un cluster. L'utilisation de la communication en multi-sauts permet de réduire la consommation d'énergie et d'augmenter la scalabilité du réseau.

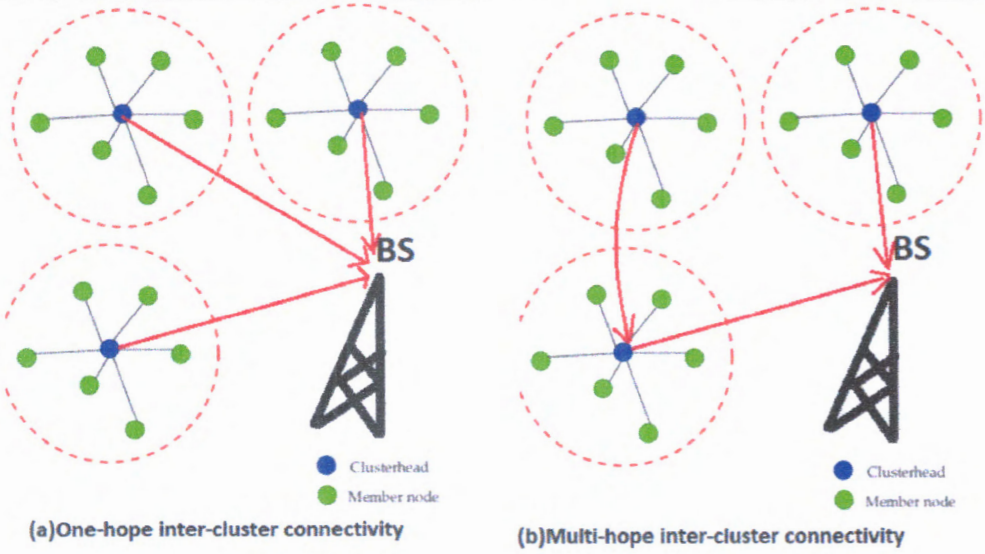


Figure 10 : Communication inter-cluster dans RCSF

La Figure 10 montre la nature de la communication inter-cluster, où dans (a) la communication inter-cluster par un seul saut, dans (b) la communication par un ou plusieurs sauts.

2.2.2.5 Le niveau d'agrégation de données

Selon le type des capteurs utilisés, l'agrégation de données peut se faire à chaque noeud du réseau ou bien uniquement au niveau des Cluster-Heads. L'agrégation de données permet de réduire la taille des données échangées entre les noeuds, et par conséquence réduire l'énergie dépensée. Plusieurs techniques d'agrégation sont utilisées à savoir : des fonctions élémentaires comme la somme, la moyenne, l'écart type, etc. ou bien des fonctions plus complexes spécifiques aux applications utilisées [24].

2.2.2.6 La nature des clusters générés

Les algorithmes de clustering utilisés peuvent générer deux types de clusters : des clusters disjoints et des clusters interconnectés. Dans le premier type, un nœud ne peut appartenir qu'à un et un seul cluster à la fois (le cas le plus fréquent); sauf que pour des applications spécifiques telles que le routage inter-cluster, la localisation et la synchronisation des nœuds; les clusters interconnectés sont utilisés. Ce type de clustering permet aux nœuds d'appartenir à un ou plusieurs clusters à la fois [26].

2.2.2.7 Variabilité du nombre de clusters

Sur la base de la variabilité du nombre de clusters, les schémas de classification peuvent être classés en deux types : fixe et variable. Dans le premier cas, le nombre de Cluster-heads est

déterminé par avance et le nombre de clusters est une constante. Dans le second cas, le nombre de clusters est variable, les CHs étant sélectionnés, de manière aléatoire ou selon certaines règles.

2.2.2.8 Uniformité des tailles de cluster

Les protocoles de routage de clusters dans les RCSFs peuvent être classés en deux catégories : des classes égales et inégales en termes de nombre de capteurs appartenant à chaque cluster. Ce qui correspond à la taille du cluster. En général, les groupes de tailles différentes permettent d'obtenir une consommation d'énergie plus uniforme et d'éviter les pertes d'énergie.

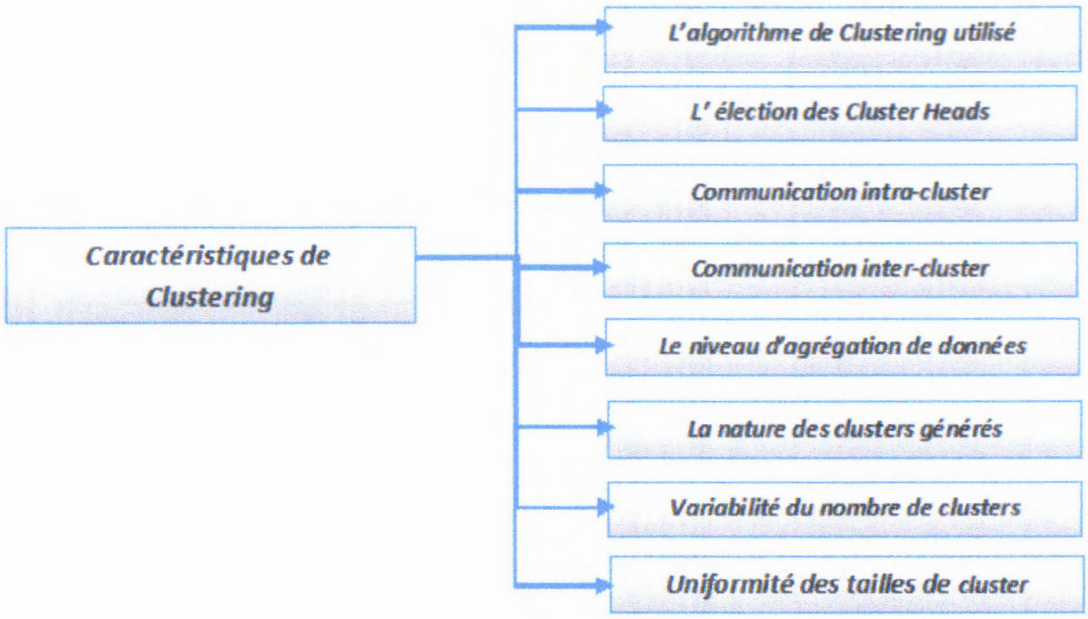


Figure 11 : Les caractéristiques de clustering dans les RCSF

Le schéma de la *Figure 11* illustre les caractéristiques du clustering dans les RCSFs décrites précédemment.

2.2.3 Taxonomie des attributs de clustering

Dans cette section nous allons énumérer l'ensemble des attributs qui peuvent être utilisés pour catégoriser et différencier les algorithmes de clustering dans les RCSFs, nous pouvons identifier les attributs suivants :

2.2.3.1 Propriétés de cluster

- Nombre de clusters

Cela signifie combien de clusters sont formés en un seul tour. Un grand nombre de clusters conduit à une distribution de taille réduite, meilleure en termes de consommation d'énergie. les CHs sont prédéterminés dans certaines des approches publiées, les algorithmes de sélection de

CHs sélectionnent généralement de manière aléatoire des CHs à partir des capteurs déployés, ce qui donne un nombre variable de clusters.

• Stabilité

La topologie du réseau sera dite adaptative quand le nombre de clusters varie et les nœuds membres d'un cluster évolue dans le temps, autrement, il est considéré fixe puisque les nœuds ne changent pas de clusters et le nombre de clusters reste invariant tout au long de la durée de vie du réseau.

•Topologie intra-cluster

Certains capteurs communiquent directement avec leur CH désigné, mais parfois, une connectivité capteur multi-saut vers CH est requise particulièrement quand la portée de communication est limitée.

•Connectivité inter-cluster

Quand un CH n'a pas de possibilité de communication à longue portée, sa connectivité à la station de base doit être assurée. Dans ce cas, l'approche de clustering doit assurer la praticabilité d'établir un itinéraire inter-cluster entre chaque CH et à la station de base. Cependant, certains travaux supposent que les CHs pourraient atteindre directement la station de base [19].

•Connectivité de CH à la station de base

La connexion peut être directe ou indirecte (lien simple ou multi-saut). Elle fait référence à l'emplacement de la station de base et à la stratégie de routage.

2.2.3.2 Capacités Cluster-Head

Les capacités des CHs dans les schémas de groupage influencent le processus de regroupement en termes de stabilité et de durée de vie du réseau de capteurs. Voici quelques attributs permettant de différencier les schémas de classification [28].

•Mobilité

La mobilité des CHs dans les réseaux de capteurs peut être attribuée en fonction des objectifs définis dans les schémas de clusters. Si les CHs sont mobiles, nous pouvons utiliser ceci pour créer un cluster équilibré afin d'améliorer les performances du réseau. Les CHs mobiles peuvent également être déplacées si le réseau de capteurs en avait besoin [28].

•Type de nœud

Au moment du déploiement, certains noeuds sont prédéfinis en tant que CH sur la base de davantage d'énergie, les CHs sont équipés de ressources de calcul et de communication plus significatives.

•Le Rôle

Le rôle des CHs dans les réseaux de capteurs peut servir de relais pour les informations générées par les membres du cluster ou effectuer la tâche d'agrégation ou de fusion de données.

2.2.3.3 Sélection du CH en fonction de :

•Energie initiale

Il s'agit d'un paramètre important pour sélectionner le CH. Au début, la plupart des algorithmes considèrent généralement l'énergie initiale comme principale référence pour la sélection de CH.

•Énergie résiduelle

Une fois les premiers tours terminés, la sélection de CH doit être en fonction de l'énergie restante dans les capteurs [28].

•Energie moyenne du réseau

L'énergie moyenne est utilisée comme énergie de référence pour chaque nœud. C'est l'énergie idéale que chaque nœud devrait posséder dans le cycle actuel pour maintenir le réseau en vie [28].

•Taux de consommation d'énergie

C'est un autre paramètre important qui considère le taux de consommation de l'énergie [28].

2.2.3.4 Processus de clustering

•Méthodologie

Quand les CHs sont des capteurs homogènes, le clustering doit être exécuté de façon distribuée sans coordination. Dans certaines approches, une autorité centrale, la station de base par exemple, est responsable de la configuration des clusters, de la détermination de l'attribution du rôle de chaque nœud.

Dans le cas de cluster-heads riches en ressources, des techniques hybrides peuvent être utilisées.

Cependant, la coordination inter-CHs est assurée d'une manière distribuée, alors que chaque CH se charge individuellement de la formation de son propre cluster.

•Objectifs de regroupement des nœuds

Les algorithmes de formation de clusters et de choix des cluster-heads visent plusieurs objectifs tels que : la tolérance aux pannes, l'équilibrage de charge, la connectivité réseau, la communication sociale, le guidage. Nous présentons ces objectifs en détail ultérieurement.

•Complexité d'algorithme

Selon l'objectif et la méthodologie, de nombreux algorithmes de clustering ont été proposés. La complexité et le taux de convergence de ces algorithmes peuvent être constants ou dépendants du nombre de CHs et/ou de nœuds capteurs [19].

•Sélection du cluster-head

En ce qui concerne la sélection des cluster-heads, ces derniers peuvent être configurés à l'avance ou choisis au hasard parmi un ensemble de nœuds.

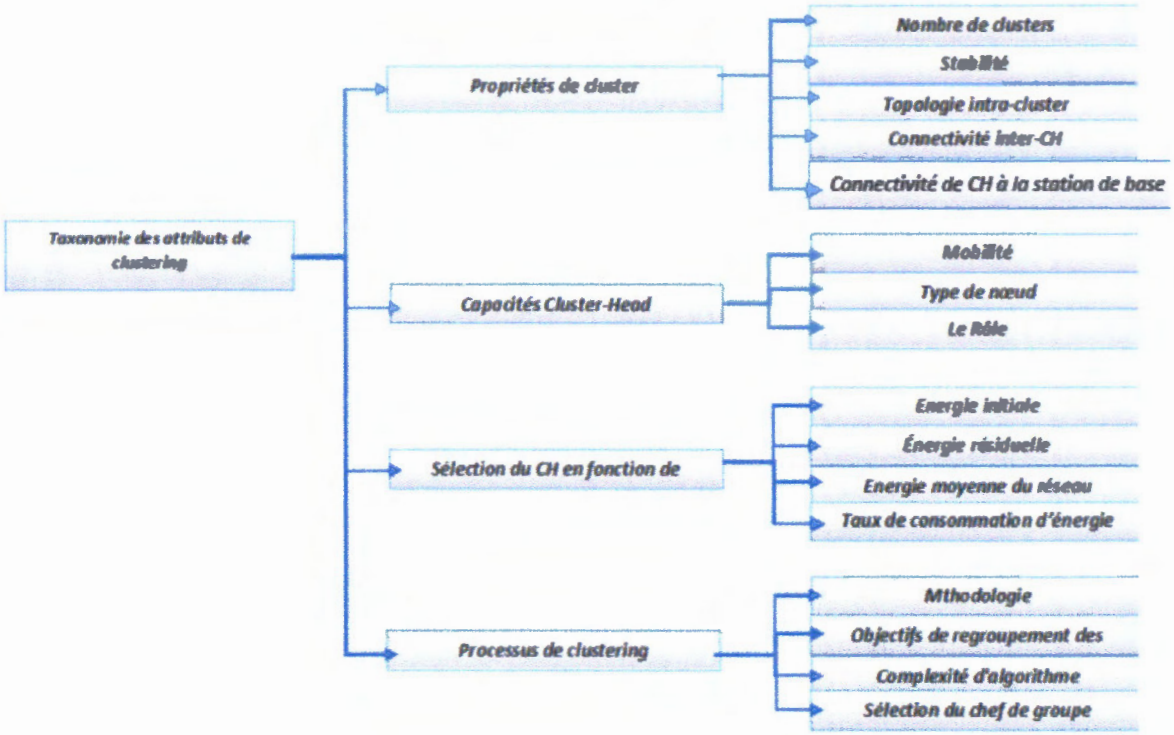


Figure 12 : La taxonomie des attributs de clustering dans les RCSFs

Le schéma de la Figure 12 illustre La taxonomie des attributs du clustering dans les RCSFs décrites précédemment.

2.2.4 Avantage et objectifs du clustering

Comparé aux protocoles de routage à plat dans les RCSFs, la structuration du réseau en groupes vise à réduire les communications et les coûts. Cet objectif global comprend plusieurs sous-objectifs, nous allons énumérer les plus importants. Les protocoles de routage en cluster aussi présentent de nombreux avantages, tels que plus d'évolutivité, moins de charge, une consommation d'énergie moindre et plus de robustesse. Nous résumons ces avantages ainsi que les objectifs [21] du clustering dans les RCSFs comme suit :

2.2.4.1 Plus d'évolutivité

Dans le schéma de routage en clustering, les nœuds sont divisés en une variété de clusters avec différents niveaux d'affectation. Les CHs sont responsables de la communication, et les nœuds membres (MN) sont chargés de la détection d'événements et de la collecte d'informations dans leur environnement. Comparé à une topologie plate, ce type de topologie réseau est plus facile à gérer et plus évolutif pour répondre aux événements de l'environnement [29].

2.2.4.2 Agrégation / fusion de données

Le processus d'agrégation des données à partir de plusieurs nœuds permet d'éliminer les transmissions redondantes et fournir des données fusionnées à la station de base (BS). La plus importante méthode d'agrégation-fusion de données dans un capteur sans fil réseau est une agrégation de données en cluster, dans laquelle chaque CH combine les données collectées à partir des autres nœuds membres de son cluster et transmet les données fusionnées à la BS.

2.2.4.3 Moins de consommation d'énergie

Dans le schéma de routage en clusters, la collecte de données réduit la taille des données envoyées, ce qui contribue à réduire la consommation d'énergie. En outre, les communications intra-cluster et inter-cluster peuvent réduire le nombre de nœuds effectuant une communication ainsi que les distances entre nœuds permettant ainsi moins d'énergie consommée pour l'ensemble du réseau.

2.2.4.4 Plus de robustesse

Le système de routage en clusters convient mieux à la topologie du réseau et permet de réagir aux changements de réseau, notamment en augmentant le nombre de noeuds, en mobilité et en imprévus échecs, etc. Il permet également à faire face à des changements au sein de groupes individuels, de sorte que l'ensemble du réseau soit plus puissant et plus pratique à gérer.

2.2.4.5 Prévention des collisions

Dans le modèle plat à sauts multiples, le support sans fil est partagé et géré par des nœuds individuels. Ce modèle peut donc réduire l'efficacité de l'utilisation des ressources. Dans le modèle de clustering à sauts multiples, un RCSF est divisé en clusters et les communications de données entre nœuds comprennent deux modes, à savoir intra-cluster et inter-cluster, respectivement pour la collecte de données et pour la transmission de données. En conséquence, les ressources peuvent être allouées orthogonalement à chaque cluster pour réduire les collisions entre les clusters et être réutilisées cluster par cluster. En conséquence, le modèle de clustering à sauts multiples convient aux RCSFs de grande taille [21].

2.2.4.6 Moins de charge

Les capteurs pouvant générer des données significatives redondantes, l'agrégation ou la fusion de données est devenue un principe et un objectif important dans les réseaux de capteurs. L'idée principale de l'agrégation ou de la fusion de données est de combiner des données provenant de différentes sources pour éliminer les transmissions de données redondantes et fournir une vue riche et multidimensionnelle des cibles surveillées. De nombreux schémas de routage en clusters avec des capacités de données nécessitent une sélection minutieuse pour l'approche de clustering.

Pour la topologie en clusters, tous les membres du cluster envoient uniquement des données aux CH, et l'agrégation des données est effectuée au niveau des CHs, qui aide à réduire considérablement les données transmises et économiser de l'énergie. De plus, les routes sont établies dans les clusters, ce qui réduit ainsi la taille de la table de routage stockée au niveau du capteur individuel. [30].

2.2.4.7 Tolérance aux pannes

En raison de l'applicabilité des RCSFs dans de nombreux scénarios dynamiques, les nœuds capteurs peuvent souffrir d'épuisement de l'énergie, d'erreurs de transmission, de dysfonctionnements matériels et d'attaques malveillantes etc. Avec des applications telles que la modélisation et le suivi des ouragans envisagées pour utiliser un grand nombre de petits noeuds capteurs, le coût de chaque noeud capteurs est limité. En raison de contraintes importantes sur le coût, et donc sur la qualité des capteurs, ainsi que des environnements souvent hostiles dans lesquels ils sont déployés, les réseaux de capteurs sont sujets à des pannes. Ainsi, la tolérance aux pannes est un défi important dans les réseaux de capteurs sans fil [31]. Afin d'éviter la perte de données significatives en provenance de nœuds clés, la tolérance aux pannes de cluster head est généralement requise dans ce type d'applications. Par conséquent, des approches efficaces à tolérance de pannes doivent être conçues dans les RCSFs. Le regroupement en clusters est la méthode la plus intuitive pour récupérer d'une défaillance de cluster, bien que cela perturbe généralement l'opération en cours. L'affectation d'un CH de sauvegarde est un schéma viable pour la reprise après une défaillance CH [21].

2.2.4.8 Maximisation de la durée de vie du réseau

La durée de vie du réseau est une considération inévitable dans les RCSFs, car les nœuds capteurs sont limités en termes d'alimentation, de capacité de traitement et de bande passante de transmission. L'idée d'économie d'énergie dépend du déploiement de capteurs de manière à

pouvoir couvrir efficacement la zone sélectionnée et le transfert de données à l'aide d'algorithmes de clustering permettant d'identifier les moyens d'allonger la durée de vie du réseau dans la communication entre groupes.

2.2.4.9 Qualité de service

Les applications de RCSFs imposent une exigence de qualité de service (QoS). Généralement, un échantillonage éfficace, moins de retard et une précision temporaire sont requis. Il est difficile pour tous les protocoles de routage de satisfaire toutes les exigences de la qualité de service, car certaines demandes peuvent enfreindre un ou plusieurs principes de protocole. Les approches de routage en clusters existantes dans les RCSFs se concentrent principalement sur l'augmentation de l'efficacité énergétique plutôt que sur la prise en charge de la qualité de service. Les métriques de QoS doivent être prises en compte dans de nombreuses applications en temps réel, telles que le suivi des cibles de bataille, la surveillance des événements émergents, etc [21].

2.2.4.10 L'équilibrage de charge

L'équilibrage de charge est un facteur clé pour prolonger la durée de vie d'un RCSF, où des groupes de tailles égales contribuent efficacement à la longévité du réseau et prévenir le dysfonctionnement prématuré, les CHs assurant le traitement des données et la gestion intra-cluster.

2.3 Les protocoles de routage hiérarchique

Dans les protocoles de routage hiérarchique, le réseau est divisé en clusters, chaque cluster étant constitué de capteurs ordinaires et d'une tête de cluster, qui est responsable du routage du groupe vers les autres clusters ou vers la station de base, soit par connexion directe, soit par liaison multi-sauts. En effet, ce type de protocole contribue à l'évolutivité globale du réseau, à réduire la consommation d'énergie et à prolonger la durée de vie du réseau.

Dans ce mémoire, nous aborderons certains protocoles hiérarchiques ou basés sur des clusters, dans lesquels nous présentons une étude générale de ces protocoles, et nous essaierons de les diviser en trois catégories comme suit :

2.3.1 Protocoles de routage de mise en clusters basés sur la construction de cluster

2.3.1.1 LEACH (Low Energy Adaptive Clustering Hierarchy)

LEACH est l'une des approches de clustering les plus populaires, forme des clusters en utilisant un algorithme distribué, où les nœuds prennent des décisions autonomes sans aucune centralisation.

Dans LEACH, les nœuds s'organisent en clusters. Dans chaque cluster, un nœud agit en tant que cluster-head et tous les autres transmettent leurs données à ce dernier. Par ailleurs, chaque cluster-head qui reçoit les données transmises par les membres du groupe, effectue des agrégations sur ces données, et envoie les résultats à la sation de base [32].

Le but principal de LEACH est de parvenir à un équilibre entre la consommation énergétique de tous les nœuds, les cluster-heads sont sélectionnés par rotation. De cette manière, la charge d'énergie d'un cluster-head est répartie uniformément entre tous les nœuds, et la consommation d'énergie élevée lors de la connexion à la station de base est distribuée et commune à tous les capteurs à un rythme proche.

Le fonctionnement de LEACH est divisé en tours. Chaque tour commence par une phase de configuration qui consiste à organiser les nœuds en clusters ; suivie d'une phase d'état stable.

Pendant la phase de configuration, chaque nœud génère un nombre aléatoire compris entre 0 et 1 et le compare à un seuil prédéfini T (n). Si nombre aléatoire < T (n), le nœud devient CH dans ce tour, sinon il s'agit d'un nœud membre [33].

Lorsqu'un nœud est élu CH avec succès, il diffuse un message de publication aux autres nœuds. En fonction de la puissance du signal reçu de la publicité, d'autres nœuds décident du groupe auquel ils se joindront pour ce tour et enverront un message d'appartenance à leur CH [21].

Dans la deuxiéme phase (l'état stable), les données des nœuds membres sont transférées au cluster-head et ensuite vers la station de base.

Parmi les avantages de LEACH est qu'il garantit l'égale probabilité de chaque nœud d'être CH. Le mécanisme de clusters permet aux nœuds d'effectuer des communications sur des petites distances avec leurs CH afin d'optimiser l'utilisation du média de communication en la faisant gérer localement par un CH pour minimiser les interférences et les collisions. L'agrégation de données permet de réduire la quantité d'informations transmise. Cela permet de réduire la complexité des algorithmes de routage, de simplifier la gestion du réseau, d'optimiser les dépenses d' énergie.

Mais d'un autre côté, LEACH souffre toujours de certains inconvénients, où nous notons que, les nœuds les plus éloignés du CH meurent rapidement par rapport aux plus proches. L'utilisation d'une communication à un seul saut au lieu d'une communication multi-sauts diminue l'énergie rapidement.

La Figure 13 présente la topologie de base de LEACH.

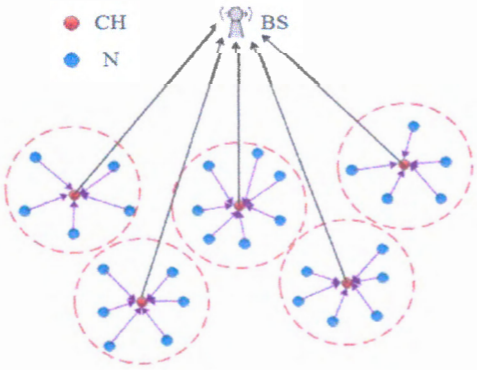


Figure 13 : La topologie de base de LEACH.

2.3.1.2 HEED (Hybrid, Energy-Efficient, Distributed approach)

HEED est un algorithme de mise en clusters à sauts multiples qui apporte un routage de mise en cluster économe en énergie, qui pallie les inconvénients des CHs inégalement distribués comme celui de l'algorithme LEACH. L'un de ses aspects le plus important, c'est la méthode avec laquelle sont sélectionnés les CHs [28]. Dans le but d'assurer un équilibrage de puissance énergétique, HEED ne sélectionne pas les nœuds en tant que CH au hasard, la construction des clusters se fait selon une combinaison de deux paramètres. L'un des paramètres dépend de l'énergie résiduelle des nœuds, le second dépend du coût des communications intra-cluster. Initialement, dans HEED, un pourcentage de CH parmi tous les nœuds, C_{prob}, est défini pour supposer qu'un pourcentage optimal ne peut pas être calculé a priori. La probabilité qu'un nœud devienne un CH est [21]:

$$CH_{prob}=C_{prob}\frac{Eresidual}{Emax}$$

où $E_{residual}$ est l'énergie actuelle estimée du nœud et E_{max} est une énergie maximale de référence, qui est généralement identique pour tous les nœuds du réseau. La valeur de CH_{prob} n'est toutefois pas autorisée à tomber en dessous d'un certain seuil choisi inversement proportionnel avec E_{max}. Ensuite, chaque nœud effectue plusieurs itérations jusqu'à ce qu'il trouve le CH. S'il n'entend aucun canal, le nœud se choisit lui-même et envoie un message d'annonce à ses voisins. Chaque nœud double sa valeur CH_{prob} et passe à l'itération suivante jusqu'à ce que son CH_{prob} atteigne 1. Par conséquent, un nœud peut annoncer à ses voisins: l'état provisoire et l'état final . Si son CH_{prob} est inférieur à 1, le nœud devient un CH provisoire et peut changer son statut en un nœud normal à une itération ultérieure s'il trouve un CH moins coûteux. Si son CH_{prob} a atteint 1, le nœud devient définitivement un CH [21].

Parmi les avantages de HEED, il vise à réaliser une distribution uniforme des clusters heads dans le réseau et à générer des clusters équilibrés en taille [24]. Il soutient également l'économies de l'énergie et l'évolutivité. Il est considéré aussi comme une méthode de clustering entièrement distribuée.

D'autre part HEED présente certaines limitations : Par exemple, la concurrence de cluster-heads peut empêcher certains nœuds de se joindre à n'importe quel cluster. De plus HEED a besoin de plusieurs itérations pour former des clustesrs générant de nombreux paquets broadcast. Aussi les capteurs proches de la station de base consomment également trop d'énergie par rapport aux autres nœuds en raison de leur rôle de nœuds intermédiaires dans le routage à sauts multiples. La *Figure 14* représente la topologie du protocole HEED.

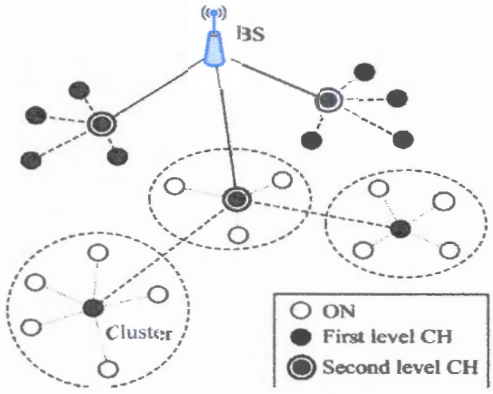


Figure 14 : La topologie du protocole HEED.

2.3.1.3 EEUC (Energy-Efficient Unequal Clustering)

L'algorithme EEUC (Energy-Efficient Unequal Clustering), est un algorithme de clustering et de concurrence distribuée [21]. Il divise l'ensemble du réseau en clusters de tailles différentes, dans lesquels les CHs sont choisis par compétition localisée, les clusters plus proches de la station de base ont des tailles plus petites que ceux plus éloignés de la station de base. Ainsi, les cluster-heads proches de la station de base consomment donc moins d'énergie pendant le traitement des données intra-clusters et peuvent conserver plus d'énergie pour le trafic de relais inter-clusters.

Comparé à l'algorithme de routage par regroupement uniforme (par exemple, LEACH, HEED), le protocole EEUC peut résoudre efficacement le problème des "points chauds" et

prolonger la durée de vie du réseau [34]. Le protocole EEUC est divisé en phase de clustering et phase de transmission de données.

Dans la phase de clustering, chaque nœud génère un nombre aléatoire compris entre 0 et 1, et seul le nœud dont le nombre est supérieur à un seuil sera activé pour l'élection de CH par un message de concurrence diffusé par la radiodiffusion dans un rayon de concurrence déterminé par sa distance au BS. Le nœud devient candidat à être cluster-head, sinon il passe en état de veille jusqu'à la fin de la compétition finale des cluster-heads.

Dans EEUC, le routage multi-sauts est utilisé pour la communication inter-cluster. Les CHs choisissent des nœuds de relais pour la transmission de données en fonction de l'énergie résiduelle des nœuds et de la distance à la BS, En d'autres termes, un CH choisirait celui qui avait le coût d'énergie résiduelle comme nœud relais parmi les deux dont les coûts de communication sont le moins élevé parmi tous ses CH voisins [21]. La *Figure 15* représente la topologie du protocole EEUC.

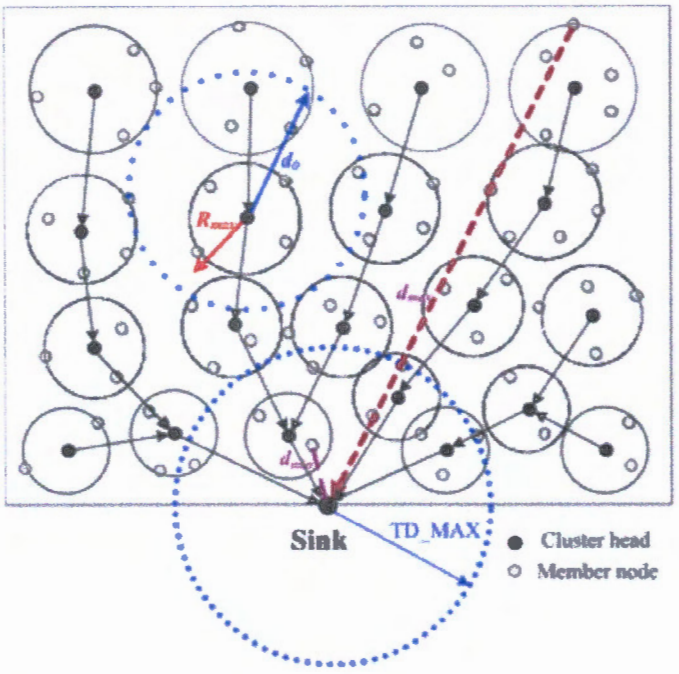


Figure 15 : La topologie du protocole EEUC.

2.3.1.4 LEACH-C (Low Energy Adaptive Clustering Hierarchy Centralized)

LEACH-C est une variante de LEACH utilise une architecteur centralisée, a été conçue pour répondre au problème de sélection aléatoire du CH dans LEACH, dans laquelle la formation de

cluster est effectuée par la station de base et considère uniquement l'énergie résiduelle comme fonction objective pour décider de la sélection des CHS.

Parmi les avantages de LEACH-C est qu'il permet une diminution remarquable de la consommation énergétique.

D'autre part, l'inconvénient de ce protocole est qu'il ne convient pas aux réseaux à grande échelle car il est problématique d'envoyer l'état d'un nœud éloigné de la station de base. Le rôle de cluster-head tourne à chaque fois, il est donc impossible d'envoyer des informations à chaque fois rapidement. Cela augmente la latence et le délai [29].

2.3.2 Protocoles de routage de mise en cluster basés sur la transmission de données

2.3.2.1 PEGASIS (Power-Efficient Gathering in Sensor Information Systems)

L'idée principale de PEGASIS est de former une chaine entre les nœuds de sorte que chaque nœud communique uniquement avec ses voisins proches. Les données collectées sont transmises, et agrégées, d'un noeud à un autre jusqu'à ce qu'elles arrivent à un cluster-head. Ce dernier les transmet, à son tour. Les emplacements des nœuds sont aléatoires et chaque noeud a la capacité de capturer des données, de communiquer en sans fil, de fusionner des données et de se localiser. Contrairement aux protocoles vus précédemment, PEGASIS évite la formation des clusters et celui-ci forme des chaines de nœuds. La construction de ces chaines constitue la première phase du protocole qui se fait en deux étapes :

- Dans la première étape, la station de base (SB) et les nœuds capteurs sont auto-organisés en utilisant un algorithme de types glouton (algorithme permettant de choisir localement la meilleure solution).
- Dans la deuxième étape, la station de base diffuse auprès des nœuds capteurs des informations concernant la chaine à établir, puis vient la formation de la chaine qui débute par le nœud le plus éloigné de la SB jusqu'à ce que tous les nœuds soient inclus dans la chaine [35].

Lorsqu'un nœud de la chaîne meurt, la chaîne sera reconstruite de la même manière pour contourner le nœud mort. La sélection de ce cluster-head se base sur plusieurs paramètres prédéterminés, tel que le rapport (Signal)/(Bruit). Le cluster-head élu demeure seulement pour une période de temps, ensuite dans la prochaine période, un autre noeud sera choisi pour un autre tour [36].

Parmi les points positifs, nous voyons que PEGASIS prolonge la durée de vie du réseau et cela en diminuant le surcout causé par le processus de formation de clusters dans LEACH, et en diminuant le nombre de transmissions et de réceptions en employant l'agrégation de données, et disperse uniformément la charge d'énergie dans le réseau. Pour s'assurer que le nœud capteur fixe n'est pas sélectionné en tant que CH et empêcher ainsi la mort précoce ultérieure de ce nœud capteur, tous les nœuds agissent tour à tour en tant que CH [37].

Cependant, ce protocole est adapté seulement aux nœuds qui sont immobiles car il est nécessaire de disposer d'une vue complète de la topologie du réseau sur chaque nœud pour la construction de la chaîne et que tous les nœuds doivent pouvoir transmettre directement vers la station de base. Ainsi, ce schéma ne convient pas aux réseaux dont la topologie varie dans le temps, et il ne gère pas la scalabilité en plus du problème du point chaud (Les points chauds apparaissent souvent dans le routage multi-sauts et constitue les nœuds voisins de la station de base et à travers lequel la majeure partie du trafic passe avant d'atteindre la station de base.). La *Figure 16* représente une chaine PEGASIS.

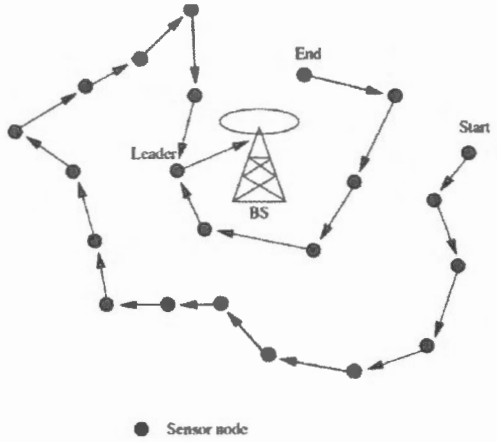


Figure 16 : La collecte de données avec PEGASIS.

2.3.2.2 TEEN (Threshold sensitive Energy Efficient sensor Network)

TEEN est un protocole hiérarchique dont le but principal est de faire face aux changements soudains des attributs détectés tels que la température. Le protocole combine la technique hiérarchique en ligne avec une approche centrée sur les données. Les nœuds détectent leur environnement en permanence, mais la consommation d'énergie de cet algorithme peut potentiellement être bien inférieure à celle du réseau proactif, car la transmission des données est moins fréquente [21]. TEEN utilise la même stratégie que LEACH pour l'étape de formation de clusters en plus des niveaux, mais adopte une approche différente pour la phase de transmission

des données où chaque CH envoie à ses membres deux seuils : un seuil Fort HT (hard threshold), qui est la valeur seuil du paramètre contrôlé (surveillé) et un seuil Faible ST (soft threshold) représentant une petite variation ou changement de la valeur du paramètre contrôlé. Le premier seuil (HT)est une valeur de seuil pour l'attribut détecté. C'est la valeur absolue de l'attribut au-delà de laquelle le nœud qui détecte cette valeur doit activer sa valeur (doit allumer son émetteur et le signaler à son CH) [38]. Si la valeur captée dépasse le seuil HT pour la première fois elle est stocké dans une variable SV et elle est transmise par le CH au nœud concerné, par la suite si la valeur captée dépasse la valeur stockée par une magnitude ST, le nœud décide de la transmettre et stocke cette nouvelle valeur dans son cache pour les comparaisons ultérieures [24]. Ainsi, le seuil critique tente de réduire les communications de données en permettant aux nœuds de transmettre uniquement lorsque l'attribut détecté se trouve dans la plage d'intérêt. Le deuxième seuil réduit davantage les communications de données qui auraient pu se produire autrement lorsque l'attribut détecté n'a que peu ou pas changé. Aux dépens d'une consommation d'énergie accrue, une valeur inférieure du seuil variable génère des informations plus précises sur le réseau [21].

L'un des avantages de ce protocole est que l'utilisation de seuils permet de contrôler le transfert des données de manière proportionnée, car seules les données sensibles demandées sont transférées, ce qui réduit la consommation de puissance de transmission et améliore les performances tout en prenant en charge des applications importantes.

D'autre part, ce protocole souffre toujours d'inconvénients, car il ne convient pas aux applications de rapports périodiques, car s'il n'atteint pas le seuil, aucune donnée ne sera transmise à l'utilisateur en tant que tel. La station de base peut ne pas être en mesure de distinguer le nœud mort des nœuds actifs.

2.3.3 Protocoles de routage de mise en cluster basés sur la la population pour résoudre des problèmes discrets

2.3.3.1 ACO-C (Ant Colony Optimization for Clustering)

ACO-C est un protocole de clustering hiérarchique mis en œuvre au niveau de la station de base, basé sur l'algorithme ACO, qui utilise une approche basée sur la population pour résoudre des problèmes discrets. Il simule comment les vraies fourmis recherchent le chemin le plus court entre la source de nourriture et leur nid. Le modèle ACO-C est similaire au modèle utilisé dans le système LEACH et fonctionne sur plussieurs étapes. Chaque étape est divisée en deux phases : la phase de configuration des clusters et la phase à états fixes. Tous les nœuds ont la capacité d'agir en tant que cluster-head. Lors de la première étape avant la configuration des clusters, tous les

nœuds envoient à la station de base leurs propres informations sur les niveaux d'énergie et les localisations. La station de base sélectionne uniquement les nœuds dont le niveau d'énergie est supérieur à l'énergie moyenne du réseau et peutvent devenir des cluster-heads. ACO-C vise à résoudre le problème de la mise en clusters, en sélectionnant les cluster-heads de manière efficace pour atteindre une consommation d'énergie minimale, et en utilisant l'algorithme ACO pour trouver la meilleure solution, où sont sélectionnés K sur N nœuds comme cluster-heads. Pour ce faire, un groupe d'agents, appelé «fourmis logiciels», crée un ensemble de solutions basées sur un algorithme déterminé en fonction de la valeur des phéromones, de la distance et du coût [41]. Enfin, une recherche locale est effectuée pour choisir parmi les solutions qui ont été créées. Après avoir trouvé la meilleure solution, la station de base envoie un message à chaque nœud pour le prévenir des cluster-heads choisis.

En fin, les nœuds membres envoient leurs données au cluster-head, qui les compile, les fusionne et les renvoie à la station de base en un saut.

2.3.3.2 ACA-LEACH (Ant Colony Algorithm Low Energy Adaptive Clustering Hierarchy)

ACA-LEACH est une version améliorée de LEACH basée sur les colonies de fourmis. L'algorithme considère non seulement l'énergie résiduelle du nœud, mais également la distance entre les cluster-heads dans la sélection des cluster-heads [42]. Il utilise l'algorithme de colonie de fourmis dans la technique de routage inter-cluster qui réduit la consommation d'énergie des cluster-heads et optimise ainsi la durée de vie du réseau de capteurs [43]. Cet algorithme fonctionne sur des tours où chaque tour est divisé en phase de clustering et phase d'exploitation. Dans le nouveau tour qui commence, le cluster est divisé puis attend le transfert de données. Dans la phase de division du groupe, la valeur énergétique résiduelle des capteurs devrait être un facteur clé dans la sélection des cluster-heads, car les cluster-heads accomplissent des tâches supplémentaires, telles que l'agrégation de données et leur rôle de nœuds intermédiaires, leur imposant de consommer plus d'énergie que les autres nœuds. De plus, il peut y avoir une distribution irrégulière des cluster-heads en raison d'une sélection aléatoire, où il y a une forte concentration de cluster-heads dans certaines zones. L'algorithme utilise un moyen de résoudre ce problème en définissant une distance minimale entre les cluster-heads (min-dis), lorsque la distance entre deux cluster-heads est inférieure à la distance (min-dis), cet algorithme compare la valeur de l'énergie entre les deux, puis sélectionne le cluster-head avec le plus d'énergie en tant que nouveau cluster-head. Dans la deuxième étape, les données sont envoyées entre le cluster-head via une connexion multi-sauts,

afin de réduire la consommation d'énergie dans les cluster-heads éloignés de la station de base, ce qui contribue à prolonger la durée de vie du réseau. Le chemin le plus court entre le cluster-head et la station de base est déterminé par le comportement des fourmis. Les détails de l'algorithme sont fournis dans ce document [44], après quoi le processus est répété jusqu'à ce que chaque CH trouve le meilleur chemin vers la station de base et la consommation d'énergie la plus basse.

2.4 Comparaison entre les différents protocoles de routage et de clustering étudiés

Dans cette section, nous comparons les différents algorithmes de routage et de clustering pour les RCSFs. Pour comparer et analyser les différentes philosophies des protocoles de routage conçus pour les réseaux de capteurs sans fil, il est important d'utiliser des critères de classification appropriés pour pouvoir les distinguer. En effet, la classification permet aux concepteurs de mieux comprendre les caractéristiques de ces protocoles et de discerner les relations qui les relient. Nous résumons les catégories et les différences des protocoles de routage en clusters dans le tableau suivant :

	Classification	Mobilité	Agrégation de données	Multi saut	Scalabilité	Localisation	Qos
LEACH	Hiérarchique	Non	Distribué	Non	Limité	Distribué	Non
HEED	Hiérarchique	Non	Centralisé	Oui	Bonne	Centralisé	Non
EEUC	Hiérarchique	Non		Oui	Limité	Distribué	Non
LEACH-C	Hiérarchique	Non	Distribué	Non	Limité	Centralisé	Non
PEGASIS	Hiérarchique	Non	Non	Oui	Limitée	Distribué	Non
TEEN	Hiérarchique	Non	Centralisé	Oui	Limitée	Centralisé	Non
ACO-C	Hiérarchique	Non	Distribué	Non	Limité	Distribué	Non
ACA-LEACH	Hiérarchique	Non	Distribué	Oui	Limité	Distribué	Non

Tableau 1 : Tableau comparatif des différents protocoles de routage étudiés

2.5 Conclusion

Dans ce chapitre nous avons étudié quelques protocoles de routage hiérarchiques les plus courants, en ce qui concerne la durée de vie du réseau et la consommation d'énergie. Nous avons également abordé certains des avantages et inconvénients de chaque protocole.

Le Clustering permet aux capteurs de communiquer sur de longues distances avec leurs propres CHs et réduire les collisions entre les paquets transmis. Les CHs sont chargés de transmettre les données sensibles à la station de base. Dans un RCSFs, les contraintes énergétiques des nœuds jouent un rôle important dans la conception de toute application de protocole ; pour cela, il est difficile de concevoir un protocole de routage efficace, économe en énergie et évolutif pour les réseaux de capteurs sans fil. Des paramètres tels que l'efficacité énergétique, l'équilibrage de charge et l'évolutivité peuvent être pris en compte pour évaluer et déterminer le meilleur parmi ces protocoles.

Dans le prochain chapitre, nous étudierons en details les protocole implémentés ACO-C basé sur la simulation du comportement de colonies de fourmis, et LEACH-C basé sur le principe du recuit similé.

Chapitre 3 : Les protocoles de clustering implémentés

3.1 Introduction

Après avoir passer en revue, en chapitre 2, les techniques d'optimisation d'énergie nous avons selectioné deux protocoles de clustering que nous allons implémenter. La première est basée sur les algorithmes de colonies de fourmis (Ant Colony Optimization for Clustering). La deuxième est basée sur la technique de recuit simulé (LEACH-C).

Dans ce chapitre on commence par introduire les algorithmes de colonies de fourmis (ACO), puis on présentera les deux protocoles de clustering implémentées

3.2 Brève introduction à l'ACO

Compte tenu de l'importance croissante et le développement important des communications, des systèmes en réseau plus complexes sont en cours de conception et de développement. Les problèmes liés à la vaste complexité des réseau, tels que l'équilibrage de la charge, le routage et le contrôle de la congestion, accentuent le besoin de techniques plus sophistiquées et plus intelligentes pour résoudre ces problèmes. S'appuyant sur certaines techniques informatiques inspirées par le comportement des insectes, telles que les colonies de fourmis [46], plusieurs paradigmes basés sur des agents mobiles ont été conçus pour résoudre les problèmes de routage et de regroupement des données dans les reseau de capteurs sans fils. Bien que les fourmis, dans leur comportement individuel, soient une simple créature, une colonie de fourmis peut effectuer des tâches collectives utiles et intelligentes telles que la construction d'un nid et la recherche de nourriture.

Dans ce chapitre, nous évoquerons le principe de l'algorithme d'optimisation des colonies de fourmis(ACO) pour répondre au besoin de regroupement des données dans les reseau de capteurs sans fils. L'algorithme de colonies de fourmis (ACO) a été proposé par l'italien M. Dorigo dans les années 1990. C'est un algorithme d'optimisation intelligent distribué qui simule réellement le comportement de recherche de nourriture de la communauté naturelle. Il l'a utilisé pour résoudre le problème du voyageur de commerce [47] et a obtenu de bons résultats expérimentaux. Au cours de ces dernières années, grâce aux efforts continus d'experts et de spécialistes, l'algorithme de base des colonies de fourmis a été continuellement développé et amélioré. Dans les communications, le transport, la chimie et d'autres domaines doivent être optimisés, cela a été vérifié. À l'heure actuelle, l'algorithme de colonie de fourmis a été appliqué avec succès aux agents de voyages, aux calendriers de production, aux affectations, au routage du réseau de communication, [47] etc. L'algorithme d'optimisation des colonies de fourmis est une sorte

d'algorithme de simulation intelligente distribuée, qui s'inspire du comportement biologique de la colonie de fourmis dans la nature. Certaines des caractéristiques des problèmes de routage réseau, telles que les informations internes, l'informatique répartie, les dynamiques aléatoires et les mises à jour asynchrones de l'état du réseau, l'algorithme de colonie de fourmis fournit une nouvelle direction pour résoudre le problème du routage réseau. Afin de trouver le chemin optimal du nœud source au nœud cible avec un coût énergétique inférieur. L'algorithme prend en compte le facteur d'énergie dans la sélection de probabilité du chemin et l'amélioration du degré d'information, prolongeant ainsi la durée de vie de l'ensemble du réseau. Les biologistes ont constaté que le comportement des fourmis à la recherche de nourriture correspond au comportement du groupe et qu'il ne s'agit pas d'un comportement de fourmi unique. La caractéristique la plus importante de la colonie de fourmis est que les fourmis utilisent les phéromones comme moyen de communication.

3.3 L'approche de l'algorithme des colonies des fourmis (ACO)

3.3.1 L'origines de Ant Colony Optimization (ACO)

ACO simule *comment les vraies fourmis* trouvent le chemin le plus court entre la source de nourriture et leur nid. En fonction des effets de ces phéromones [48]. Où les fourmis vont laisser "la phéromone" sur le trajet qu'elles suivent, et la probabilité de sélectionner la prochaine fourmi pour le chemin riche en phéromones est plus grande grande que la probabilité de choisir le chemin sans la substance. Dans le même temps, ils libèrent une certaine phéromone pour améliorer la concentration de phéromone sur le trajet. Après ce cycle constant, plus de fourmis parcourent le chemin, plus il reste de phéromone et plus de fourmis le suivent. Ce processus se poursuivra jusqu'à ce que toute la colonie de fourmis sélectionne un chemin optimal. Afin de mieux expliquer le principe de l'algorithme de groupe de fourmis, nous présentons ci-dessous un modèle simplifié et discrétisé du phénomène expliqué à la *Figure 17* [46]. Notre modèle consiste en un graphe G = (V, E), où V est constitué de deux nœuds, à savoir V_s (représentant le nid des fourmis) et V_d (représentant la source de nourriture). De plus, E est constitué de deux liens, à savoir e_1 et e_2, entre V_s et V_d. Pour e_1, on attribue une longueur de l_1 et pour e_2 une longueur de l_2 telle que $l_2 > l_1$. En d'autres termes, e_1 représente le chemin court entre V_s et V_d et e_2 représente le chemin long. Les vraies fourmis déposent de la phéromone sur les chemins sur lesquels elles se déplacent. Ainsi, les traces de phéromones chimiques sont modélisées comme suit. Nous introduisons une valeur de phéromone artificielle τ_i pour chacun des deux liens e_i, i = 1, 2. Une telle valeur indique la force de la trace de phéromone sur le chemin correspondant. Enfin, nous présentons n_a fourmis

artificielles. Chaque fourmi se comporte comme suit : à partir de V_s (c'est-à-dire du nid), une fourmi choisit avec la probabilité :

$$P_i = \frac{\tau_i}{\tau_1+\tau_2}, \; i = 1, 2.$$

entre le chemin e_1 et le chemin e_2 pour atteindre la source de nourritur V_d. Evidemment, si $\tau_1 > \tau_2$, la probabilité de choisir e_1 est plus forte et vice versa. Pour revenir de V_d à V_s, une fourmi utilise le même chemin qu'elle a choisi pour atteindre V_d, et ça change la valeur de phéromone artificielle associée au chemin utilisé. Plus en détails, après avoir choisi le chemin e_i, une fourmi change la valeur de phéromone artificielle τ_i comme suit:

$$\tau_i \leftarrow \tau_i + \frac{\mathbb{Q}}{l_i}$$

où la constante positive $\mathbb{Q}$ est un paramètre du modèle. En d'autres termes, la quantité de phéromone artificielle ajoutée dépend de la longueur du chemin choisi : plus le chemin est court, plus la quantité de phéromone ajoutée est élevée. Dans ce modèle, l'alimentation d'une colonie de fourmis est simulée de manière itérative : à chaque étape (ou itération), toutes les fourmis sont initialement placées dans le nœud initial. Ensuite, chaque fourmi se déplace de V_s à V_d. comme indiqué ci-dessous. Comme mentionné dans la légende de la *Figure 17*(**d**), dans la nature, la phéromone déposée est soumise à une évaporation dans le temps. Nous simulons cette évaporation de phéromone dans le modèle artificiel comme suit :

$$\tau_i \leftarrow (1-p) * \tau_i \;, i=1,2.$$

Le paramètre $\rho \in]0, 1]$ est un paramètre qui régit l'évaporation de la phéromone. Enfin, toutes les fourmis effectuent leur trajet de retour et renforcent le chemin choisi.

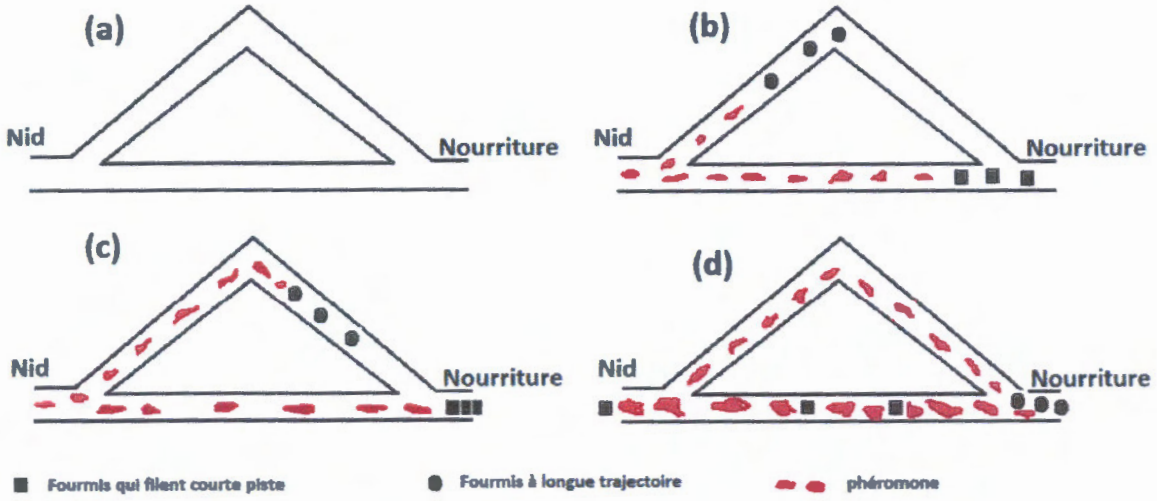


Figure 17 : Un cadre expérimental qui démontre la capacité des colonies de fourmis de trouver le chemin le plus court entre le nid des fourmis et l'unique source de nourriture s'il existe deux chemins de longueurs différentes.

Un tel processus de recherche coopérative régi par des phéromones mène au comportement intelligent des essaims. Ce comportement de recherche dans la vie réelle a été le principal facteur d'inspiration menant à la formulation d'algorithmes de fourmis artificiels pour résoudre plusieurs problèmes d'optimisation combinatoire et fonctionnelle. Les applications actuelles des algorithmes ACO entrent dans les deux classes de problèmes importants des problèmes d'optimisation combinatoire statique et dynamique. Les problèmes statiques sont ceux dont la topologie et les coûts ne changent pas tant que les problèmes sont en cours de résolution. C'est le cas, par exemple, du TSP classique [50], dans lequel les localisations des villes et les distances entre les villes ne changent pas pendant l'exécution de l'algorithme. En revanche, dans les problèmes dynamiques, la topologie et les coûts peuvent changer pendant que des solutions sont élaborées. Un exemple d'un tel problème est le routage dans les réseaux de télécommunications.

3.3.2 Les caractéristiques de l'algorithme de base d'optimisation des colonies de fourmis (ACO) :

L'algorithme d'optimisation de base des colonies de fourmis est caractérisé par [47] :

3.3.2.1 Choix du chemin :

Plus la concentration en phéromone est grande, plus les chances d'être choisi sont grandes.

3.3.2.2 Mise à jour des phéromones :

La concentration de phéromones au-dessus du trajet augmente avec le passage de la fourmi. Dans le même temps, la phéromone s'évapore progressivement.

3.3.2.3 Mécanisme de rétroaction positive :

Les fourmis utilisent la phéromone pour communiquer et travailler ensemble. Sur le trajet optimal, les fourmis laisseront plus de phéromone et la concentration plus forte en phéromone attirera plus de fourmis de la colonie, ce qui donnera plus d'informations sur le meilleur trajet. Cependant, la concentration en phéromone sur le chemin non-optimal est progressivement réduite en raison de l'évaporation. Ce processus converge vers la solution optimale. Le résultat final est que toutes les fourmis de la colonie de fourmis sont atirées par le même chemin optimal, ce qui est bénéfique pour la convergence de l'algorithme.

3.3.2.4 Calcul distribué parallèle

En utilisant la méthode de calcul distribué, plusieurs fourmis recherchent de nombreux points dans l'espace de la solution en fonction de la concentration en phéromone, ce qui est très approprié pour une mise en œuvre parallèle. Par conséquent, il s'agit essentiellement d'un algorithme de recherche parallèle et efficace. La distribution de l'algorithme de groupe de fourmis est caractérisée par deux aspects : premièrement, les phéromones sont distribuées sur chaque arc du graphe, et chaque fourmi construit la solution en fonction de la situation de phéromone du point actuel, et n'a pas besoin de contrôle centralisé. Deuxièmement, tout le système est capable de fonctionner normalement lorsqu'un ou plusieurs individus cessent de travailler. Par conséquent, l'algorithme a une forte robustesse.

3.3.2.5 Stagnation facile :

Après que l'algorithme ait recherché un certain degré, la phéromone située sur un chemin optimal local sera plus haute que l'autre. Toutes les fourmis sont atirées par ce chemin, de sorte que la solution ne peut plus être explorée. La distribution irrationnelle de la phéromone mène facilement à la stagnation. Dans ce cas, en ajustant dynamiquement la valeur de la phéromone, nous pouvons limiter la phéromone à une plage appropriée, de manière à éviter ces problèmes d'optima locaux.

3.3.3 Le principe de l'ACO :

3.3.3.1 La structure du système d'optimisation

Considérons un problème de minimisation $(\mathcal{S}, \mathcal{F})$, où $\mathcal{S}$ est l'ensemble des solutions possibles et $\mathcal{F}$est la fonction objectif, qui attribue à chaque solution $s \in \mathcal{S}$ une valeur de coût $\mathcal{F}(s)$. L'objectif est de trouver une solution optimale s^*, c'est-à-dire une solution réalisable d'un coût

minimal. L'ensemble de toutes les solutions optimales est noté S^*. L'optimisation des colonies de fourmis tente de résoudre ce problème de minimisation en répétant les deux étapes suivantes :

- **Les solutions candidates sont construites à l'aide d'un modèle probabiliste paramétré, c'est-à-dire une distribution de probabilité paramétrée sur l'espace de la solution.**
- Les solutions candidates sont utilisées pour modifier le modèle de manière à orienter l'échantillonnage futur en faveur de solutions à faible coût.

3.3.3.2 Le modèle probabiliste de l' ACO

Nous supposons que le problème d'optimisation combinatoire $(S, \mathcal{F})$, est mappé sur un problème qui peut être caractérisé par la liste d'items suivante :

• Un ensemble fini $C = c_1, c_2, \dots, c_N$ de composants, où N est le nombre de composants.

• Un ensemble fini X d'états du problème, où un état est une séquence $x = c_1, c_2, \dots, c_K, ..$ sur les éléments de C. La longueur d'une séquence $x$est exprimée par $|X|$. La longueur maximale d'une séquence est limitée par une constante positive $n < +\infty$.

• Un ensemble de solutions S, qui est un sous-ensemble de X (c'est-à-dire, $S \subseteq X$).

• Un ensemble d'états réalisables $\tilde{X}$, avec $\tilde{X} \subseteq X$, définis via un ensemble de contraintes Ω.

• Un ensemble non vide S^*de solutions optimales, avec $S^* \subseteq \tilde{X}$ et $S^* \subseteq S$.

Le principe de l'ACO peut être appliqué à tout problème d'optimisation combinatoire. des fourmis artificielles construisent des solutions candidates en effectuant des marches aléatoires sur le graphe pondéré complètement connecté $G = (C, \ell, T)$, Où c est l'ensemble des sommets, L est l'ensemble qui relie complètement les composants C, et T est un vecteur de trajets de phéromone τ. Les traces de phéromone peuvent être associées à des composants, des connexions ou les deux. Nous supposons ici que les traces de phéromones sont associées à des connexions, de sorte que τ_{ij} est la phéromone associée à la connexion entre les composants i et j. Il est facile d'étendre l'algorithme aux autres cas. Le graphe G s'appelle le graphe de construction. Pour construire des solutions candidates, chaque fourmi artificielle est d'abord placée sur un sommet du graphe choisi de manière aléatoire. ensuite il se déplaçant à chaque étape d'un sommet à l'autre du graphe, de telle sorte que le prochain sommet soit choisi de manière stochastique en fonction de la force de la phéromone se trouvant actuellement sur les arcs. Lors du déplacement d'un noeud à un autre du graphe G, les contraintes Ω peuvent être utilisées pour empêcher les fourmis de construire des

solutions irréalisables. Le comportement de construction d'une solution d'une fourmi générique peut être décrit comme suit :

Pour chaque fourmi : Sélectionnez un nœud de départ c_1 en fonction d'un critère dépendant du problème, Définisse K=1 et $x_k = < c_1 >$.

Tantque $x_k = \langle c_1, c_2, \dots, c_k \rangle \in \tilde{X}$, $x_k \notin S$, et l'ensemble J_{x_k} des composants pouvant être ajoutés à x_k n'est pas vide, sélectionnez le nœud suivant (composant) c_{k+1} en fonction de :

$$P_\tau(c_{k+1} = c | x_k) = \begin{cases} \dfrac{\mathcal{F}_{(c_k,c)}\left(\tau(c_k,c)\right)}{\sum(c_k,y) \in j_{x_k} \mathcal{F}_{(c_k,c)}\left(\tau(c_k,y)\right)} & , \ si\ (c_k,c) \in j_{x_k} \\ 0 & , \ autre \end{cases}$$

où une connexion (c_k,y) appartient à J_{x_k} si et seulement si la séquence $x_{k+1} = \langle c_1, c_2, \dots, c_k, y >$ satisfait aux contraintes Ω (c'est-à-dire, $x_{k+1} \in \tilde{X}$). et $\mathcal{F}_{(c_k,c)}(z)$ est une fonction monotone dont le choix commun est (z) $\alpha_{\eta}(i,j)^{\beta}$, où α,β>0 et η(i, j) sont des valeurs heuristiques mesurant l'opportunité d'ajouter un composant j après i. Si $x_k \notin S$ et $j_{x_k}=\emptyset$, c'est-à-dire si le processus de construction a atteint une impasse, l'état actuel x_k est ignoré. Cependant, cette situation peut être évitée en permettant à des fourmis artificielles de construire également des solutions irréalisables. Dans un tel cas, un terme de pénalité d'infaisabilité est généralement ajouté à la fonction de coût. Néanmoins, dans la plupart des paramètres dans lesquels ACO a été appliqué, la situation d'impasse ne se produit pas. Pour certains problèmes, il peut être utile d'utiliser un schéma plus général, dans lequel $\mathcal{F}$ dépend des valeurs de phéromone de plusieurs connexions «liées» plutôt que d'une seule. De plus, au lieu de la règle de proportionnalité aléatoire ci-dessus, différents schémas de sélection, tels que la règle de proportionnalité pseudo-aléatoire [51], peuvent être utilisés.

3.3.3.3 Mise à jour de phéromone dans ACO

La mise à jour des phéromones est une partie importante des algorithmes basés sur les colonie des fourmis. Par conséquent de nombreux schémas différents pour la mise à jour des phéromones ont été proposés dans l'ACO. Pour un aperçu complet, voir [52]. La plupart des mises à jour de phéromones peuvent être décrites à l'aide du schéma générique suivant :

- $\forall s \in \hat{S}_t, \forall (i,j) \in s: \tau(i,j) \leftarrow \tau(i,j) + \varphi_f(s|S_1, S_2 \dots, S_t)$
- $\forall (i,j): \tau(i,j) \leftarrow (1-\rho)\tau(i,j)$

où s_i est l'échantillon dans la i^{ème} itération, ρ, $0 \leq \rho < 1$, correspond au taux d'évaporation, et $\varphi_f(s|S_1, S_2 \ldots, S_t)$ correspond à une «fonction de qualité», qui doit normalement ne pas augmenter par rapport à f et est défini sur le «jeu de référence» $\widehat{S}_t$.

Différents algorithmes ACO peuvent utiliser différentes fonctions de qualité et ensembles de référence. Par exemple, dans le tout premier algorithme ACO Ant System [53]. la fonction de qualité était simplement $\frac{1}{f_{(s)}}$ et l'ensemble de référence $\widehat{S}_t = S_T$. Dans un schéma proposé plus récemment, appelé itération meilleure mise à jour [51], l'ensemble de référence était un singleton contenant la meilleure solution dans St (s'il y avait plusieurs solutions d'itération optimales, l'une d'entre elles était choisie au hasard). Pour la meilleure mise à jour globale [51] [54], l'ensemble de référence contenait la meilleure parmi toutes les meilleures solutions d'itération (et s'il existait plus d'une solution optimale, le meilleur est le plus ancien qui est choisi). Dans [53] une stratégie élitiste a été introduite, dans laquelle la mise à jour était une combinaison des deux précédentes. Dans le cas où une bonne limite inférieure sur le coût optimal de la solution est disponible, on peut utiliser la fonction de qualité suivante [55] :

$$\varphi_f(s|S_1, S_2 \ldots, S_t) = \tau_0\left(1 - \frac{f_{(s)} - LB}{\bar{f} - LB}\right) = \tau_0 \frac{\bar{f} - f_{(s)}}{\bar{f} - LB}$$

où $\bar{f}$ est la moyenne des coûts des k dernières solutions et LB est la limite inférieure du coût optimal de la solution. Avec cette fonction qualité, les solutions sont évaluées en comparant leur coût au coût moyen des autres solutions récentes, plutôt qu'en utilisant les valeurs de coût absolues. De plus, la fonction qualité est automatiquement mise à l'échelle en fonction de la proximité du coût moyen par rapport à la limite inférieure.

Une mise à jour de phéromone légèrement différente de la mise à jour générique décrite ci-dessus a été utilisée dans le système de colonie de fourmis (ACS) [51]. Là, la phéromone est évaporée par les fourmis en ligne pendant la construction de la solution, de sorte que seule la phéromone impliquée dans la construction s'évapore. Une autre modification de la mise à jour générique a été introduite dans MAX-MIN Ant System [54], qui utilise des limites maximales et minimales de traçage aux phéromones. Avec cette modification, la probabilité de générer une solution particulière est maintenue au-dessus d'un seuil positif. Cela permet d'éviter la stagnation de la recherche et la convergence prématurée vers des solutions sous-optimales.

3.3.4 Applications de l'ACO

Les bonnes performances des algorithmes de colonies de fourmis obtenues lors de leur application au problème du voyageur du commerce ont incité beaucoup de chercheurs à les utiliser dans d'autres domaines d'application [56]. En effet, plusieurs applications réussies d'ACO à un large éventail de problèmes d'optimisation discrets sont maintenant disponibles. La grande majorité de ces applications concernent des problèmes NP-difficiles; c'est-à-dire aux problèmes pour lesquels les algorithmes les plus connus garantissant l'identification d'une solution optimale présentent une complexité exponentielle dans le pire des cas. L'utilisation de tels algorithmes est souvent irréalisable en pratique, et les algorithmes ACO peuvent être utiles pour trouver rapidement des solutions de haute qualité. D'autres applications populaires concernent les problèmes dynamiques du plus court chemin apparus dans les problèmes de réseaux de télécommunications. Le nombre d'applications réussies pour des problèmes académiques a poussé les industriels à adopter l'ACO pour résoudre certains de leurs problèmes, ce qui prouve que cette technique de l'intelligence informatique est également utile dans les applications du monde réel [57].

3.3.4.1 Applications aux problèmes NP-complet

L'approche habituelle pour montrer l'utilité d'une nouvelle technique métaheuristique consiste à l'appliquer à un certain nombre de problèmes et à comparer ses performances à celles de techniques déjà disponibles. Dans le cas d'ACO, cette recherche a d'abord consisté à tester les algorithmes sur le problème du TSP. Par la suite, d'autres problèmes NP-difficile ont également été examinés. Jusqu'à présent, l'ACO a été testé sur probablement plus d'une centaine de problèmes NP-difficiles différents. Bon nombre des problèmes abordés peuvent être considérés comme relevant de l'une des catégories suivantes : problèmes d'acheminement au fur et à mesure qu'ils se posent, par exemple, dans la distribution de biens; problèmes d'affectation, où un ensemble d'éléments (objets, activités, etc.) doit être affecté à un nombre donné de ressources (emplacements, agents, etc.) soumis à certaines contraintes; les problèmes de planification, qui - au sens le plus large - concernent l'affectation de ressources limitées aux tâches dans le temps; et les problèmes de sous-ensemble, où une solution à un problème est considérée comme une sélection d'un sous-ensemble d'éléments disponibles. De plus, l'ACO a été appliqué avec succès à d'autres problèmes émergents dans des domaines tels que l'apprentissage automatique et la bioinformatique [57].

3.3.4.2 Applications dans les réseaux de télécommunication

Les algorithmes ACO se sont révélés être une approche très efficace pour résoudre les problèmes de routage dans les réseaux de télécommunication où les propriétés du système, telles que le coût d'utilisation des liaisons ou la disponibilité des noeuds, varient dans le temps. Les algorithmes ACO ont d'abord été appliqués aux problèmes de routage dans les réseaux à commutation de circuits (tels que les réseaux téléphoniques) [58], puis dans les réseaux à commutation par paquets (tels que les réseaux locaux ou Internet) [59]. À la suite de la validation de principe fournie par Schoonderwoerd et al., les stratégies de communication de réseau inspirées des fourmis se sont améliorées au point d'être à la pointe de la technologie dans les réseaux câblés. AntNet [59] est un exemple bien connu. AntNet a fait l'objet de tests approfondis, en simulation, sur différents réseaux et sous différents schémas de trafic, ce qui s'est avéré hautement adaptatif et robuste. Une comparaison avec des algorithmes de routage à la pointe de la technologie a montré que, dans la plupart des situations considérées, AntNet surpasse ses concurrents. Les algorithmes basés sur AntNet ont donné naissance à plusieurs autres algorithmes de routage, améliorant les performances dans une variété de scénarios de réseaux câblés; voir [60], [61] si besoin. Plus récemment, un algorithme ACO conçu pour la classe complexe de réseaux ad hoc mobiles s'est révélé compétitif par rapport aux algorithmes de routage à la pointe de la technologie [62], tout en offrant une meilleure évolutivité.

3.3.4.3. Applications aux problèmes industriels

Le succès rencontré par les problèmes académiques a attiré l'attention d'un certain nombre d'entreprises qui ont commencé à utiliser des algorithmes ACO pour des applications réelles. EuroBios (www.eurobios.com) est l'un des premiers à exploiter des algorithmes basés sur la métaheuristique ACO. Ils ont appliqué ACO à un certain nombre de problèmes d'ordonnancement différents, tels qu'un problème d'atelier d'écoulement continu à deux étages avec des réservoirs finis. Les problèmes modélisés incluaient diverses contraintes du monde réel telles que les temps d'installation, les restrictions de capacité, les compatibilités des ressources et les calendriers de maintenance. AntOptima (www.antoptima.com) est une autre société qui a joué et joue encore un rôle très important dans la promotion de l'application réelle d'ACO. Les chercheurs d'AntOptima ont développé un ensemble d'outils permettant de résoudre les problèmes de routage des véhicules dont les algorithmes d'optimisation sont basés sur ACO [57]. Les produits les plus performants basés sur ces outils sont DYVOIL, pour la gestion et l'optimisation de la distribution de mazout avec un parc de camions non homogène, utilisé pour la première fois par Pina Petroli en Suisse, et AntRoute, pour l'acheminement de centaines de véhicules de sociétés telles que Migros, la

principale chaîne de supermarchés suisse, ou Barilla, le principal fabricant de pâtes alimentaires italien. Une autre application de routage de véhicules a été développée par BiosGroup pour la société française Air Liquide. Gravel, Price et Gagné [63], qui ont appliqué l'ACO à un problème de planification industrielle dans un centre de coulée d'aluminium, sont d'autres applications intéressantes. Bautista et Pereira [64], qui ont appliqué avec succès ACO à la résolution d'un problème d'équilibrage de chaîne de montage comportant une fonction à objectifs multiples et des contraintes entre tâches pour une chaîne de montage de vélo.

Il existe d'autres domaines d'application pour les algorithmes ACO où de nombreux chercheurs se sont appuyés pour résoudre divers problèmes., on peut citer le problème d'affectation quadratique [65] [66], le problème de coloration de graphe [67], les problèmes de satisfaction de contraintes [68], la fouille de données [69], l'optimisation de site d'enseignement en ligne [70].

3.3.5 L'algorithme ACO (Ant System) pour le problème de TSP

En fait, les modèles utilisés dans la section précédente pour simuler le comportement de fourmis réelles ne peuvent pas être appliqués directement aux problèmes complexes. nous avons associé les valeurs de phéromone directement aux solutions au problème (c'est-à-dire un paramètre pour le chemin court et un paramètre pour le chemin long). Cette méthode de modélisation implique que les solutions au problème considéré sont déjà connues [46]. Cependant, dans l'optimisation combinatoire, nous essayons de trouver une solution optimale inconnue. Le problème du voyageur de commerce ("Travelling Salesman Problem", TSP) a fait l'objet de la première implémentation d'un algorithme de colonies de fourmis [71]. A titre d'exemple, nous présentons l'algorithme ACO, appelé Ant System (AS) [53], appliqué au TSP, que nous avons mentionné dans l'introduction et que nous définissons plus en détail dans ce qui suit: Dans le TSP, on donne un graphe complètement connecté, non orienté, G = (V, E) avec le poids des arêtes. Les nœuds V de ce graphe représentent les villes et les poids des arêtes représentent les distances entre les villes. Le but est de trouver dans G un chemin fermé (Hamiltonien) contenant chaque nœud une seule fois et dont la longueur est minimale. Ainsi, l'espace de recherche (S) est constitué de tous les tours de G. La valeur de la fonction objectif $f_{(s)}$ d'un tour $s \in S$ est définie comme la somme des poids des arêtes de (s). Le TSP peut être modélisé de nombreuses manières différentes en tant que problème d'optimisation discrète. Le modèle le plus courant consiste en une variable de décision binaire X_e pour chaque arête de G. Si, dans une solution, X_e=1, e est une partie du tour défini par la solution. En ce qui concerne l'approche Ant System, les arrêtes du graphe TSP donné

peuvent être considérés comme des composants de la solution, c'est-à-dire que pour chaque e_{ij} est introduite une valeur de phéromone τ_{ij}. La tâche de chaque fourmi consiste à construire une solution TSP réalisable, c'est-à-dire une visite réalisable. En d'autres termes, la notion de tâche d'une fourmi passe de «choisir un chemin du nid à la source de nourriture» pour «construire une solution réalisable au problème d'optimisation traité». Notez qu'avec ce changement de tâche, les notions de nid et de source de nourriture perdent leur sens.

Chaque fourmi construit une solution comme suit. Tout d'abord, l'un des nœuds du graphe TSP est choisi de manière aléatoire en tant que nœud de départ. Ensuite, la fourmi construit une visite dans le graphe TSP en passant à chaque étape de la construction de son nœud actuel (c'est-à-dire la ville dans laquelle elle se trouve) vers un autre nœud qu'elle n'a pas encore visité. A chaque étape, l'arrête traversée est ajouté à la solution en construction. Lorsqu'il ne reste plus aucun nœud non visité, la fourmi ferme la tournée en passant de son nœud actuel au nœud dans lequel elle a démarré la construction de la solution. Cette façon de construire une solution implique qu'une fourmi ait une mémoire T pour stocker les nœuds déjà visités. Chaque étape de construction de la solution est effectuée comme suit. En supposant que la fourmi soit dans le noeud V_i, l'étape de construction suivante est effectuée avec probabilité :

$$P(e_{ij}) = \frac{\tau_{ij}}{\sum_{k\epsilon 1....|V|/v_k \notin T} \tau_{ik}}, \forall j \epsilon 1 \ldots . |V|/v_j \notin T$$

Une fois que toutes les fourmis de la colonie ont terminé la construction de leur solution, l'évaporation de la phéromone est effectuée comme suit :

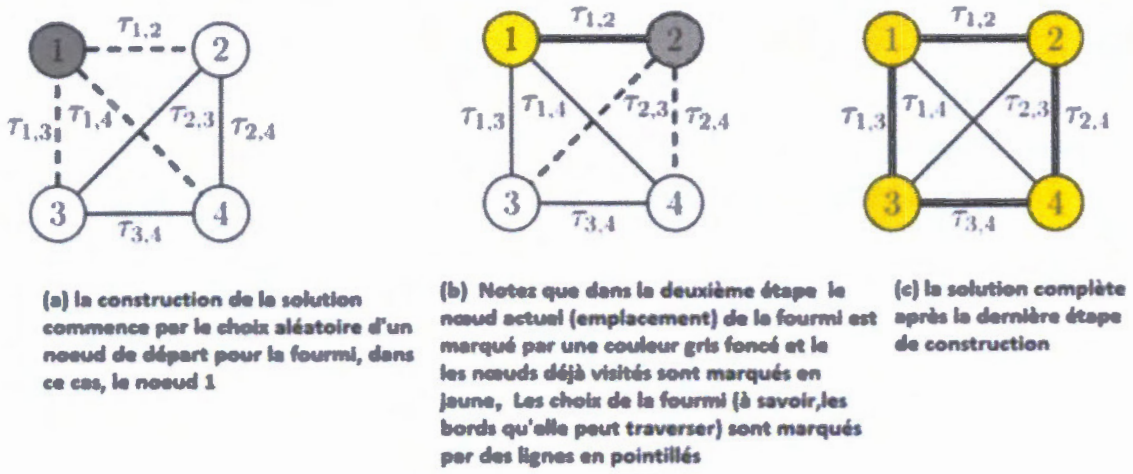
$$\tau_{ij} \leftarrow (1 - \rho) * \tau_{ij}, \ \forall \tau_{ij} \epsilon \mathcal{T}$$

où $\mathcal{T}$ est l'ensemble de toutes les valeurs de phéromone. Ensuite, les fourmis effectuent leur voyage de retour. Par la présente, une fourmi, ayant construit une solution (s) effectue pour chaque $e_{ij} \in$ (s) le dépôt de phéromone suivant :

$$\tau_{ij} \leftarrow \tau_{ij} * \frac{\varphi}{f_{(s)}}$$

où φ est encore une constante positive et $f_{(s)}$ est la valeur de la fonction objectif de la solution (s). Comme expliqué dans la section précédente, le système est répété sur n_a fourmis par itération jusqu'à ce qu'une condition d'arrêt (par exemple, une limite de temps) soit remplie. Bien que l'algorithme AS ait prouvé que le comportement d'alimentation des fourmis pouvait être

transféré dans un algorithme d'optimisation discrète, il s'est généralement révélé inférieur aux algorithmes les plus avancés. Par conséquent, au fil des ans, plusieurs extensions et améliorations de l'algorithme AS d'origine ont été introduites.



(a) la construction de la solution commence par le choix aléatoire d'un noeud de départ pour la fourmi, dans ce cas, le noeud 1

(b) Notez que dans la deuxième étape le noeud actuel (emplacement) de la fourmi est marqué par une couleur gris foncé et le les noeuds déjà visités sont marqués en jaune, Les choix de la fourmi (à savoir,les bords qu'elle peut traverser) sont marqués par des lignes en pointillés

(c) la solution complète après la dernière étape de construction

Figure 18 : Exemple de construction de solution pour un problème TSP composé de 4 villes

3.4 Implémentation et choix de protocoles

3.4.1 Introduction

Dans cette partie, nous allons détailler une méthodologie d'optimisation de colonies de fourmis (ACO-C) permettant de regrouper de manière optimale N objets dans K clusters.

ACO-C est un nouveau protocole de mise en cluster sensible à l'énergie pour les RCSF. Bien que les algorithmes de fourmis aient déjà été utilisés dans l'affectation d'objets à des clusters, la principale nouveauté provient de l'utilisation de l'algorithme ACO seulement pour trouver des cluster-head, avec l'incorporation des fonctions de coût appropriées. Ziyadi et al. dans [39], affirme que cet algorithme a été implémenté et testé sur plusieurs jeux de données simulés et réels. Les performances de cet algorithme sont comparées à celles d'autres méthodes stochastiques/heuristiques populaires et ils avaient vu qu'il donnait de bons résultats.

3.4.2 Ant Colony Optimization for clustering (ACO-C)

3.4.2.1 Modèle du système

A) Modèle de réseau

Dans cet méthode, nous considérons un modèle WSN, similaire à ceux utilisés dans LEACH et LEACH-C, avec les propriétés suivantes :

• Tous les nœuds capteurs sont immobiles et soumis à des contraintes d'énergie. Ils peuvent contrôler leur puissance d'émission pour différentes plages de transmission. En outre, tous les nœuds font des mesures de l'environnement à un débit fixe et ont toujours des données à envoyer à la station de base. De plus, tous les nœuds peuvent fonctionner en tant que cluster-head.

• L'emplacement de la station de base est fixe, que ce soit au milieu du réseau ou loin des capteurs.

• La fusion de données est utilisée pour réduire la taille totale des données envoyées, et la transmission directe d'un saut est utilisée pour envoyer les données fusionnées de chaque cluster-head à la station de base.

B) Modèle de consommation d'énergie

Nous supposons un modèle simple pour la consommation d'énergie du matériel radio dans lequel l'émetteur dissipe l'énergie nécessaire au fonctionnement de l'électronique radio et de l'amplificateur de puissance, et le récepteur dissipe l'énergie nécessaire au fonctionnement de l'électronique radio [72]. Si la distance est inférieure à un seuil, d_0, le modèle d'espace libre (perte de puissance d^2) est utilisé. Sinon, le modèle multi-trajet (perte de puissance d^4) est utilisé. Les modèles de consommation d'énergie de l'émetteur et du récepteur, séparés par la distance d pour un message à l bit, sont respectivement donnés par [42] :

$$E_{\mathrm{TX}}(l,d) = \begin{cases} lE_{elec} + l\epsilon_{fs}d^2 \text{ , } si\ d \leq d_0 \\ lE_{elec} + l\epsilon_{mp}d^4 \text{ , } si\ d > d_0 \end{cases}$$

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

où E_{elec} est l'énergie consommée par bit pour faire fonctionner les circuits de l'émetteur et du récepteur, ϵ_{fs} et ϵ_{mp} sont la perte de puissance dans l'espace libre et dans la cas du modèle multi-trajets utilisé, qui dépendent du taux d'erreur sur les bits acceptable choisi. Un choix approprié pour le seuil de la distance de transmission d_0 peut être [73] :

$$d_0 = \sqrt{\frac{\epsilon_{fs}}{\epsilon_{mp}}}$$

En outre, le coût de l'agrégation des données est modélisé par E_{DA}. Nous supposons que quel que soit le nombre de messages de données reçus dans un cluster-head, tous les messages peuvent être compressés en un seul message de même longueur.

3.4.2.2 Architecture du protocole

A) Méthode d'optimisation (Ant Colon Optimization)

Les algorithmes de fourmis font partie des classes de systèmes intelligents « Essaim » les plus réussies. En fait, ce sont des systèmes multi-agents dans lesquels chaque agent unique appelé fourmi artificielle imite le comportement de fourmis réelles. La méta-heuristique d'optimisation des colonies de fourmis (ACO), une classe particulière d'algorithmes de fourmis, est une approche basée sur la population pour résoudre des problèmes d'optimisation discrets [53]- [46]. Dans ce travail, nous avons utilisé l'algorithme ACO pour les problèmes de clustering dans lesquels N objets sont affectés à K clusters. La fonction objectif, qui doit être minimisée est la distance totale euclidienne entre chaque objet et le centre du groupe correspondant. Nous adaptons notre problème au problème de clustering dont l'objectif est de sélectionner K nœuds sur N comme cluster-heads.

Pour ce faire, R agents , appelés fourmis logicielles sont utilisés, pour construire les solutions. Chaque agent forme une solution S de longueur K, dans laquelle chaque élément représente le nœud sélectionné comme cluster-head. Les agents construisent leurs solutions à l'itération t en appliquant les informations fournies par le vecteur de phéromone mis à jour à la fin de l'itération, $t-1$. La valeur de la trace τ_i représente la concentration en phéromone du noeud i, $(\mathrm{i} = 1, 2, \ldots, \mathrm{N})$, chacun d'entre eux pouvant être sélectionné comme l'un des K cluster-heads. Le vecteur de traînée de phéromone évolue avec la répétition de l'algorithme. Enfin, une recherche locale est effectuée pour choisir parmi les meilleures solutions générées par R et, puis la valeur de phéromone est mise à jour.

B) Détails de l'algorithme

Un agent sélectionne le numéro de noeud pour chaque élément de la chaîne S de la manière suivante. Pour l'élément K de la chaîne S, l'un des nœuds appartenant à un ensemble N_K, est sélectionné avec une probabilité stochastique notée P_j donnée par [52] :

$$P_j = \frac{\tau_j}{\sum_{i=1}^{N_K} \tau_i} \quad , \quad j = 1, \ldots . N_k$$

où P_i est la probabilité normalisée de phéromone pour le noeud j, N_k est l'ensemble des noeuds distants d'au moins une distance R_0 des éléments déjà sélectionnés de S. La distance R_0 est modifiée pendant l'algorithme afin de garantir l'existence de K cluster-heads à chaque itération. Une procédure de recherche locale simple est mise en œuvre sur les L meilleures solutions qui ont les valeurs de fonction de coût les plus faibles. Dans les simulations, L. Est considéré comme 20%

des agents R. Avec un seuil de probabilité de recherche locale $P_{ls} \in [0,1]$ et une fonction de coût désignée par cost, un voisin de S_l, $l = 1 \ldots . L$ est généré comme suit :

I. $l = 1$

II. Soit S_t une solution temporaire et affecter $S_t(kk) = S_l(k)$, $k = 1, \ldots, K$.

III. Pour chaque élément k de S_t, tirer un nombre aléatoire, $r \epsilon [0, 1)]$. Si $r \leq pls$ un nœud n tel que $S_l(k) \neq n , k = 1, \ldots, K$ est choisi de manière aléatoire et on laisse $S_t(k) = n$.

IV. Calculez la fonction de coût pour la solution S_t .If $cost_t < cost_l$ alors $S_l = S_t$ et $cost_l = cost_t$.

V. $l = l + 1$; si $l \leq L$, passez à l'étape (II), sinon arrêtez.

À la fin de chaque itération, le vecteur des phéromones est mis à jour en fonction de la qualité des solutions (valeurs de la fonction de coût) produites par les fourmis. Cette mise à jour est effectuée, pour chaque nœud $i = 1, \ldots . N$ comme expliqué dans [52] :

$$\tau_i(t+1) \leftarrow (1-\rho) * \tau_i(t) + \sum_{l=1}^{R} \Delta\tau_i^l + e\Delta\tau_i^b$$

$$\Delta\tau_i^l = \begin{cases} g(1 - \dfrac{(cost_l - L_B)}{(L_{avg} - L_B)}) , & si\ l'agent\ l\ est\ utilisé \\ 0 , & sinon \end{cases}$$

$$\Delta\tau_i^b = \begin{cases} \dfrac{1}{cost_b} , & si\ le\ meilleur\ agent\ est\ utilisé \\ 0 , & sinon \end{cases}$$

où ρ est la persistance d'une trace comprise entre [0,1] et $(1-\rho)$ est le taux d'évaporation, L_B est le coût minimum pour que sa valeur soit inférieure au coût de la meilleure solution globale, L_{avg} est le coût moyen pour toutes les fourmis jusqu'à la dernière itération et les constantes g , e sont des constantes définies par l'utilisateur.

Notons que $\Delta\tau_i^l$ est égal à $g(1 - \frac{(cost_l - L_B)}{(L_{avg} - L_B)})$, si le nœud i est affecté à l'un des éléments de la solution construite par l'agent l et zéro sinon, et que $\Delta\tau_i^b$ est égal à $\frac{1}{cost_b}$ si le nœud i est affecté à l'un des éléments de la meilleure solution construite par l'agent b en tant que meilleur agent et

zéro sinon. Cette procédure est effectuée pour un nombre maximal d'itérations afin de trouver la meilleure solution. L'organigramme de cet algorithme est présenté à la *Figure 19*.

C. La Construction des clusters dans l'algorithme ACO-C

Le protocole étudié est mis en œuvre au niveau de la station de base, un nœud sans contrainte d'énergie. Cette protocole (ACO-C), comme LEACH et d'autres protocoles de clustering, fonctionne par tours. Chaque tour comprend une phase de configuration, dans laquelle des clusters sont formés, et une phase d'état stationnaire, pour laquelle nous avons considéré une procédure comme dans [74]. Dans la phase de configuration, tous les nœuds envoient les informations sur leurs niveaux d'énergie et leurs emplacements à la station de base. La station de base sélectionne uniquement les nœuds dont le niveau d'énergie est supérieur à l'énergie moyenne du réseau en tant que éventuels cluster-head et utilise l'algorithme ACO pour trouver la meilleure solution en fonction d'une fonction de coût. Une fonction de coût appropriée pour considérer à la fois les distances intra-cluster et le niveau d'énergie des nœuds est introduite dans [75]:

$$cost = \beta \times f_1 + (1-\beta) \times f_2 \quad ,$$

$$f_1 = \max_{k=1,\dots,K} \left\{ \sum_{\forall n_i \in C_{l,k}} d\,(n_i, CH_{l,k}) / |\,C_{l,k}| \right\} \quad ,$$

$$f_2 = \sum_{i=1}^{N} E(n_i) \Big/ \sum_{k=1}^{K} E\big(CH_{l,k}\big) \quad ,$$

Où la fonction f_1 est le maximum de la distance euclidienne moyenne des noeuds par rapport à leurs cluster-head associés et $|\,C_{l,k}|$ est le nombre de nœuds appartenant au cluster C_k de l'agent l.

La fonction f_2 est le rapport entre l'énergie initiale totale de tous les nœuds n_i, $i = 1, 2, \dots, N$ dans le réseau et l'énergie actuelle totale des cluster-heads dans l'itération en cours. La constante β est une constante définie par l'utilisateur.

Après avoir trouvé la meilleure solution, la station de base envoie un message à chaque nœud pour informer leurs cluster-headss. Les cluster-heads coordonnent la transmission des données dans leurs clusters correspondantes selon un calendrier TDMA. À la fin de chaque intervalle de temps, le cluster-head exécute la fusion de données sur les données reçues de la part de ses noeuds membres au cours de cet intervalle, puis envoie les données fusionnées à la station de base.

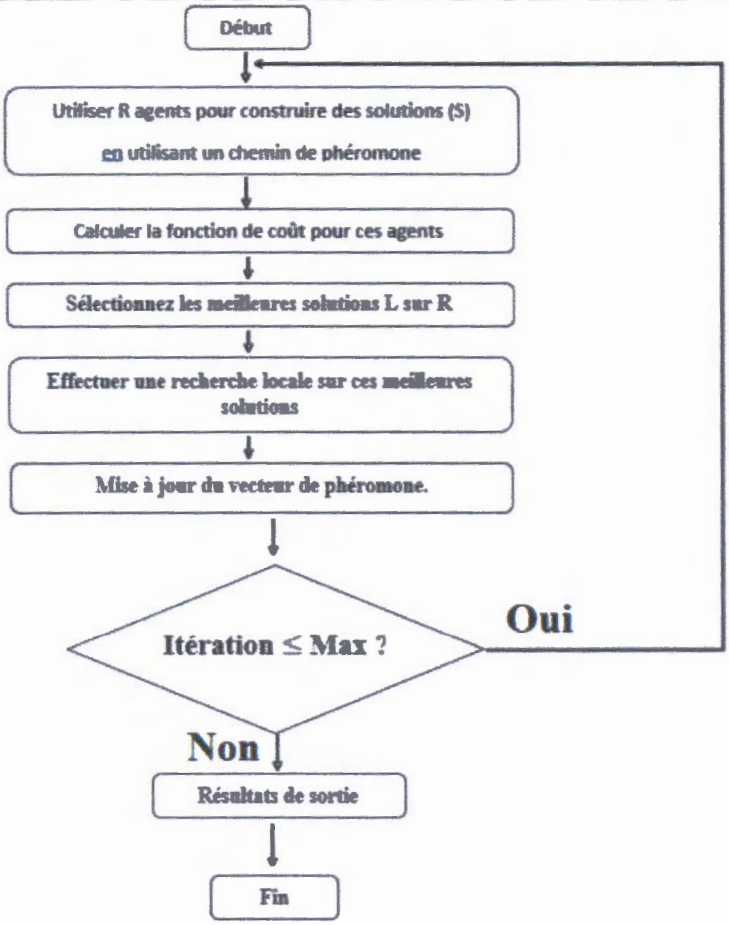


Figure 19 : L'algorithme ACO-C

3.4.2.2 Évaluation des performances

Pour évaluer les performances de ce protocole (ACO-C), ses auteurs ont simulé le réseau à l'aide du programme MATLAB, en utilisant les caractéristiques suivantes, comme indiqué dans [42] : 100 nœuds capteurs, d'énergies initiales inégales, sont répartis aléatoirement entre ($x = 0, y = 0$) et ($x = 500, y = 500$), l'énergie initiale de 20 nœuds est 5_J, et 2_J pour les autres nœuds (80 nœuds). Pour toutes les simulations, le nombre de clusters est défini à 5. de plus, les paramètres de consommation d'énergie et de modèle radio définis comme dans [72], rappelés dans le paragraphe 3.4.2.1 B) Modèle de consommation d'énergie.

$E_{elec} = 50_{nj}/bit$, $\epsilon_{fs} = \frac{10\frac{p_j}{bit}}{m^2}$, $\epsilon_{mp} = 0.0013\frac{p_j}{bit}/m^2$, $E_{DA} = 5_{nj}/bit$. et la taille de chaque message de données est de 500 octets avec 25 octets supplémentaires pour l'en-tête ud paquet. Les paramètres de l'algorithme ACO sont définis comme suit :$g = 0{,}4$, $e = 0{,}3$, $\rho = 0{,}1$ et $\beta = 0{,}5$.

ACO-C a été comparé à d'autres algorithmes de clustering connus tels que LEACH, LEACH-C et PSO-C (Particle Swarm Optimization-Clustering) [73] en termes de durée de vie du réseau et de transfert de données vers la station de base. Et a donné de bons résultats lorsque la station de base a été définie sur (x = 250, y = 250) [42], l'âge du réseau est meilleur que n'importe quel autre algorithme ayant la même fonction de coût. Cette performance est obtenue en optimisant la répartition des têtes de cluster-heads et en réduisant la distance entre les nœuds et leurs cluster-heads correspondants.

3.4.3 Approche du clustering LEACH-C

3.4.3.1 Le Principes de LEACH-C

Comme mentionné précédemment LEACH-C est une version améliorée de LEACH, divise également chaque tour en deux phases, la phase de configuration et la phase de d'état stable. Au cours de la phase de configuratio de LEACH-C, tous les nœuds envoient leur emplacement et leur énergie actuelle à la station de base. Après avoir obtenu des informations sur l'énergie de tous les nœuds, la BS calcule l'énergie moyenne du réseau, et ne marque que les nœuds dont l'énergie est supérieure que l'énergie moyenne, en tant que cluster head éligible [36]. Un nœud fonctionne comme un membre si son énergie est inférieure à l'énergie moyenne du réseau. La station de base détermine ensuite le nombre souhaité de CH à l'aide d'un algorithme de recuit simulé parmi l'ensemble de nœuds dont l'énergie est supérieure à l'énergie moyenne du réseau [37], puis difiuse un message contenant les identificateurs des cluster-heads. Chaque noeud membre est ensuite connecté au CH le plus proche pour former les clusters.

Dans LEACH-C, le nombre optimal des cluste-heads a été déterminé à l'avance. en plus, il n'y a pas de communication entre les nœuds lors de la création des clusters. Par conséquent, plus d'énergie peut être utilisée pour la transmission de données. Cela permet au réseau d'utiliser l'énergie de manière plus efficace par rapport à LEACH.

3.4.3.2 Le modèle de réseau

Dans cet article, nous supposons un modèle de réseau de capteurs avec les propriétés suivantes :

•Il y a une station de base fixe au centre de la zone où se trouvent les capteurs. Nous ne prenons pas en compte la consommation d'énergie dans la station de base et supposons qu'elle a une énergie infinie.

•Tous les nœuds ont la même configuration et une énergie limitée.

•Chaque nœud a la même énergie de démarrage dans le réseau.

•Tous les nœuds sont stationnaires.

•Le modèle de consommation d'énergie est similaire à celui utilisé dans ACO

3.4.4 La méta-heuristique du recuit simulé

3.4.4.1 Introduction

Le recuit simulé est une technique probabiliste permettant d'approcher l'optimum global pour un problème d'optimisation dans un grand espace de recherche. Le principe est de parcourir de manière itérative l'espace des solutions.

Pour les problèmes dans lesquels la recherche d'un optimum global approximatif est plus importante que la recherche d'un optimum local précis dans un laps de temps déterminé, un recuit simulé peut être préférable à des alternatives telles que la descente du gradient. Le nom et l'inspiration viennent du recuit en métallurgie, une technique impliquant le chauffage et le refroidissement contrôlé d'un matériau pour augmenter la taille de ses cristaux et réduire leurs défauts. Le chauffage et le refroidissement du matériau affectent à la fois la température et l'énergie libre thermodynamique. La simulation de recuit peut être utilisée pour trouver une approximation d'un minimum global pour une fonction comportant un grand nombre de variables.

3.4.4.2 Principes du recuit simulé

Le recuit simulé applique itérativement l'algorithme de Metropolis [76], pour engendrer une séquence de configurations qui tendent vers l'équilibre thermodynamique :

1) Choisir une température de départ T et une solution initiale $S = S_0$;
2) Générer une solution aléatoire dans le voisinage de la solution actuelle $S_0 \rightarrow S_t$, $S_t \in V_{(S)}$, au lieu de chercher la meilleur ou la première solution voisine améliorante comme dans une recherche locale classique ;

3) On calcule la variation de coût $\Delta f = f(S_t) - f(S_0)$;

4) Si $\Delta f \leq 0$, le coût diminue et on effectue la transformation améliorante comme dans une recherche locale ($s_0 := S_t$) ;

5) Si $\Delta f > 0$, le coût remonte, c'est un rebond, qu'on va pénaliser le plus si la température est basse et que Δf est grand. Une fonction exponentielle a les propriétés désirées. On calcule une probabilité d'acceptation $a = e^{-\Delta f/T}$, puis on tire au sort p

dans [0,1]. Si $p \leq a$, la transformation est acceptée, bien qu'elle dégrade le coût, et on fait S_0:=S_t. Sinon, la transformation est rejetée : on garde s pour l'itération suivante.

6) Pour assurer la convergence (analogie de la balle qui rebondit de moins en moins), T est diminuée lentement à chaque itération, par exemple $T := k * T$, $k = 0.999$ par exemple. On peut aussi décroître T par paliers. Pour être efficace, un recuit doit diminuer T assez lentement, en plusieurs milliers ou dizaines de milliers d'itérations. Il dure en tout cas beaucoup plus longtemps qu'une recherche locale, puisque les transformations améliorantes de cette dernière sont diluées parmi de nombreux rebonds.

7) On s'arrête quand T atteint un seuil fixe ε, proche de 0.

Présentation de l'algorithme en pseudo-code

Début

Générer la solution initiale S_0

$Min = S_0$

Répéter

Choisir $S_t \epsilon V(S_0)$

Calculer $\Delta^d = f_{(S_t)} - f_{(Min)}$

Si $\Delta^d < 0$ Alors

Min = S_t

$S_0 = S_t$

Sinon

tirer p dans [0,1] suivant une distribution uniforme

Si $p \leq e^{-\Delta f/T}$ Alors

$S_0 = S_t$

Sinon

S_t est rejetée

T = $g(T)$

Jusqu'à ce que T soit proche de 0

Fin.

3.4.4.3 Le principe de Metropolis :

Dans l'algorithme de Metropolis, on part d'une configuration donnée, et on lui fait subir une modification aléatoire. Si cette modification fait diminuer la fonction objectif (ou énergie du système), elle est directement acceptée; Sinon, elle n'est pas acceptée qu'avec une probabilité égale à $e^{(-\Delta E/T)}$ (avec E: énergie, et T: température), cette règle est appelé critère de Metropolis.

3.5 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base de l'algorithme d'optimisation basésur la colonie des fourmis (ACO) pour les problème d'optimisation combinatoire, ainsi que, nous avons présenté un nouveau protocole de clustering(ACO-C) qui améliore la consommation d'énergie dans les réseaux de capteurs sans fil, il utilise une

fonction de coût adaptée à l'algorithme ACO pour sélectionner l'ensemble optimal des cluster-heads. Nous avons également présenté l'algorithme de routage LEACH-C.

Dans le chapitre suivant, nous allons procéder à des simulations via notre application pour comparer ces deux algorithmes et évaluer leurs performances et leur efficacité, notamment en termes d'efficacité énergétique et de prolongation de la durée de vie du réseau.

Chapitre 4 : Implémentation et résultats de simulations

4.1 Introduction

Dans le cadre de ce projet, nous avons développé un outil appelé CLUSTERING_APPLICATION pour l'implémentation et la simulation de quelques algorithmes de clustering dans les réseaux de capteurs sans fil. En effet, nous avons implémenté deux protocoles de clustering : ACO-C [39], et LEACH-C [77].

Le premier est basé sur les algorithmes de colonies de fourmis, tandis que le deuxième est basé sur le principe de recuit simulé. Leurs principes respectifs ont été présentés en détails dans le chapitre précedent.

Dans ce chapitre, nous allons d'abord présenter l'outil CLUSTERING_APPLICATION en détails.

Ensuite, avec cet outil nous effectuons des simulations afin d'évaluer la performance de l'algorithme ACO-C amélioré, par rapport au protocole LEACH-C.

4.2 L'outil CLUSTERING_APPLICATION

Dans cette section, nous décrivons l'outil CLUSTERING_APPLICATION. Nous commençons d'abord par la présentation des objectifs de ce dernier. Par la suite, nous décrivons la structure du processus global que nous avons utilisé en phase d'implémentation. Enfin, nous terminons cette section par les différentes interfaces graphiques de notre application.

4.2.1 Objectifs

Nous avons développé l'outil CLUSTERING_APPLICATION afin de faire des simulations d'un algorithme de clustering basé sur les colonies de fourmis et d'évaluer ses performances en termes de l'efficacité énergétique notamment la durée de vie du réseau, et le comparer avec un autre algorithme de clustering par des représentations graphiques des résultats obtenus.

4.2.2 Environnement de développement

Nous avons choisi le langage de programmation Java pour développer notre application, car c'est un langage orienté objet qui permet la création d'interfaces graphiques grâce à sa bibliothèque JavaFX, et aussi parce que c'est un langage dont le code exécutable est portable. L'environnement de développement et d'exécution est le JDK 1.8, comme éditeur du code source, nous avons utilisé NetBeans 8.0.

4.2.3 Le langage Java

Java est un langage de programmation orienté objet, utilisé en informatique inspiré du langage C++, qui a été créé par Sun Microsystems en 1995 [76]. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut être utilisé sur internet pour des petites applications intégrées à la page web (applet). Parmi les avantages de Java on peut citer sa gratuité, et il est rigueureux (la plupart des erreurs se produisent à la compilation et non à l'exécution). Par ailleurs, le byte-code, qui assure à Java une portabilité complète vers de très nombreux systèmes d'exploitation.

4.2.4 La plate-forme Java FX

JavaFX est une plate-forme client riche qui permet de créer des applications fonctionnant sur plusieurs périphériques. L'un des principaux avantages de JavaFX est que le code écrit pour l'un des périphériques fonctionnera sur n'importe quel autre périphérique. Comme JavaFX est intégré à l'environnement d'exécution Java, il s'exécutera sur tous les postes de travail sur lesquels Java est installé et sur tout téléphone fournissant JavaME (Micro Edition). Il a été annoncé pour la première fois par Sun lors de JavaONE 2007. La version 1.0 a été publiée en décembre 2008, ciblée sur la plate-forme de bureau, tandis que la version 1.1 a été publiée récemment pour les appareils mobiles [77].

JavaFX permet à l'utilisateur de créer des interfaces graphiques dans un format de langage déclaratif qui définit les composants de la boîte à outils d'interface Swing de Java, ainsi que divers effets d'affichage, notamment des animations, des graphiques vectoriels, des pistes audios et des vidéos, le tout fonctionnant sur une machine virtuelle Java (JVM) [78].

4.2.5 Java Développent Kit (JDK)

Le Java Development Kit (JDK) est un composant de plate-forme clé pour la construction d'applications Java [79]. C'est un progiciel contenant une variété d'outils et d'utilitaires permettant de développer, d'empaqueter, de contrôler et de déployer des applications conçues pour toute plate-forme Java standard, notamment Java Platform Standard Edition (Java SE), Plate-forme Java Micro Edition (JavaME) ; et Java Platform Enterprise Edition (Java EE) [80]. Parmi les outils JDK qui aident au processus de développement logiciel on peut citer :

Javac : Cet utilitaire est utilisé pour compiler le code source Java en byte-code.

Rmic : Cet utilitaire crée des squelettes et des stubs à utiliser dans RMI (Remote Method Invocation).

Jar : Cet utilitaire de compression regroupe une multitude de fichiers dans un seul fichier Java Archive (JAR). L'utilitaire jar utilise un algorithme de compression standard utilisé par tous les utilitaires de compression les plus courants.

Javadoc : Cet utilitaire peut examiner les noms de classes et les méthodes contenues dans une classe, ainsi que consommer des annotations spéciales afin de créer une documentation d'interface de programmation d'application (API) pour le code Java.

4.2.6 NetBeans

NetBeans IDE est un environnement de développement intégré (IDE) gratuit et open source qui permet de développer des applications de bureau, mobiles et Web. L'IDE prend en charge le développement d'applications dans divers langages, notamment Java, HTML5, PHP et C++. L'IDE fournit une prise en charge intégrée pendant tout le cycle de développement, de la création du projet au débogage, au profilage et au déploiement. L'IDE fonctionne sous Windows, Linux, Mac OS X et d'autres systèmes UNIX.

4.2.7 Les interface de l'outil CLUSTERING_APPLICATION

L'utilisation de l'outil CLUSTERING_APPLICATION se fait à travers une fenêtre principale qui offre des raccourcis aux fonctionnalités essentielles de l'application.

4.2.7.1 La fenêtre principale

La fenêtre principale de cette application est illustrée dans la *Figure 20*. Elle est composée de quatre blocs :

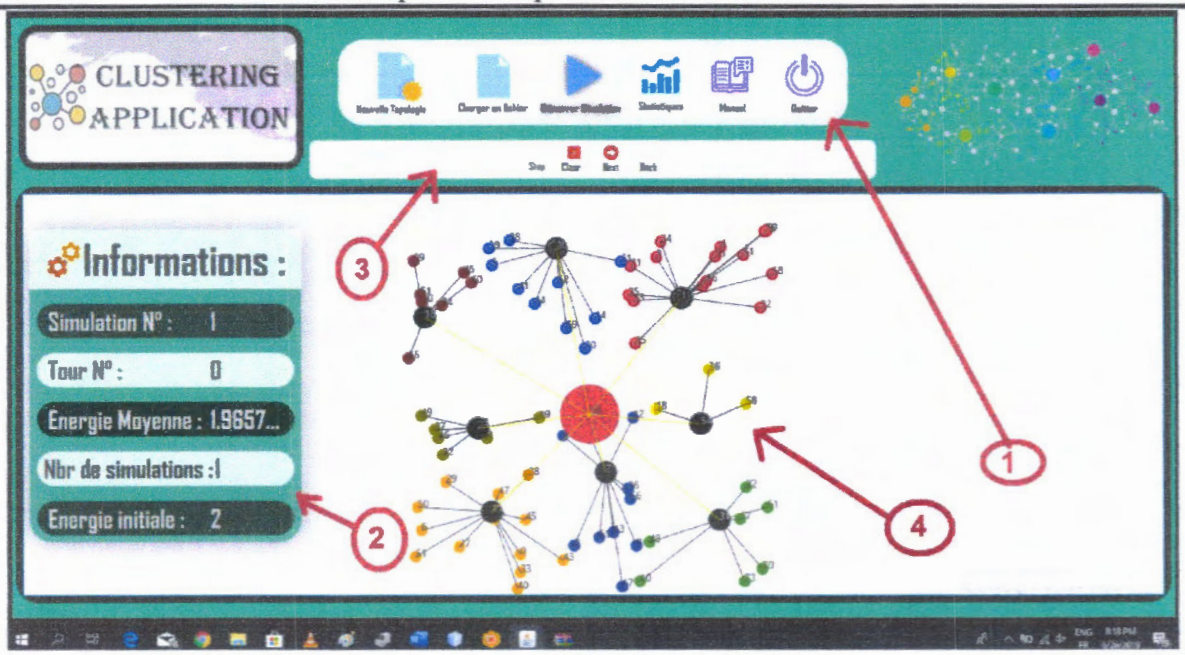


Figure 20 : Fenêtre principale

- Bloc N°1 :

Une barre d'outils (*Figure 21*) mettant à la disposition de l'utilisateur des raccourcis permettant d'accéder aux fonctionnalités principales de l'application :

Figure 21 : Barre d'outils de CLUSTERING_APPLICATION

- **Nouvelle topologie :** Permet à l'utilisateur de générer une nouvelle topologie et d'entrer les paramètres initiaux pour la simulation.
- **Charger un fichier :** Permet à l'utilisateur de charger une topologie à partir d'un fichier externe.
- **Démarrer Simulation :** Permet à l'utilisateur de lancer la simulation dont les paramètres ont été définis auparavant.
- **Statistiques :** Permet à l'utilisateur d'afficher les résultats de simulation sous forme de courbes.
- **Manuel :** Une explication simple de la façon d'utilisation de cette application.
- **Quitter :** Pour fermer l'application.

- Bloc N°2 :

La fenêtre qui contient des informations sur la simulation en cours. Elle est présentée sur la *Figure 22*.

Figure 22 : Fenêtre des informations de simulations

- **Simulation N° :** Permet d'afficher le numéro de la simulation en cours. En effet plusieurs simulations successives peuvent être effectuées.
- **Tour N° :** Permet à l'utilisateur de voir le nombre de périodes pendant le processus de simulation à un moment donné.
- **Energie Moyenne :** Permet à l'utilisateur de voir l'évolution de l'énergie moyenne actuelle dans le réseau lors de la simulation en cours.
- **Nbr de simulations :** Affiche le nombre de simulations successives que l'application va effectuer.
- **Energie initiale :** C'est la valeur de l'énergie initiale pour tous les nœuds.

- Bloc N°3 :

Une barre d'outils qui contient des raccourcis pour le déroulement la simulation :

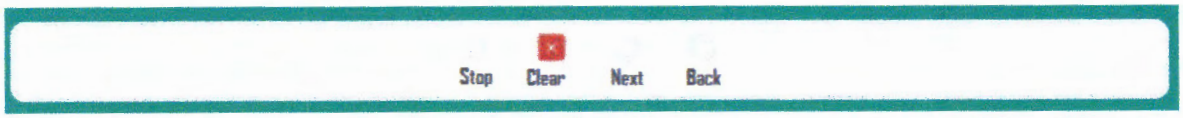


Figure 23 : La barre d'outils de déroulement de la simulation

- **Stop** : Permet à l'utilisateur d'arrêter la simulation à un moment donné, avec la possibilité de continuer la simulation plus tard.
- **Clear** : Permet à l'utilisateur de revenir à l'état initial de la simulation.
- **Next** : Permet à l'utilisateur de passer à la topologie de l'étape suivante.
- **Back** : Permet à l'utilisateur de revenir à la topologie de l'étape précédente.

- Bloc N°4 :

Une interface pour afficher une représentation graphique de la topologie. Elle est montrée sur la *Figure 24*.

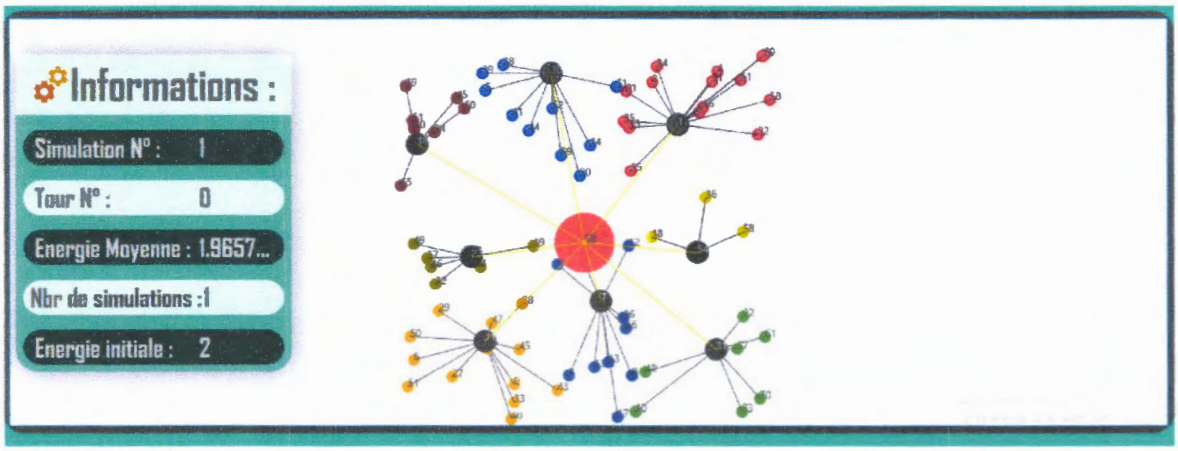


Figure 24 : La représentation graphique de la topologie

4.2.7.2 La fenêtre de configuration

Cette fenêtre (*Figure 25*) sert au paramétrage de la simulation. Elle permet également à l'utilisateur de définir les paramètres d'une nouvelle topologie.

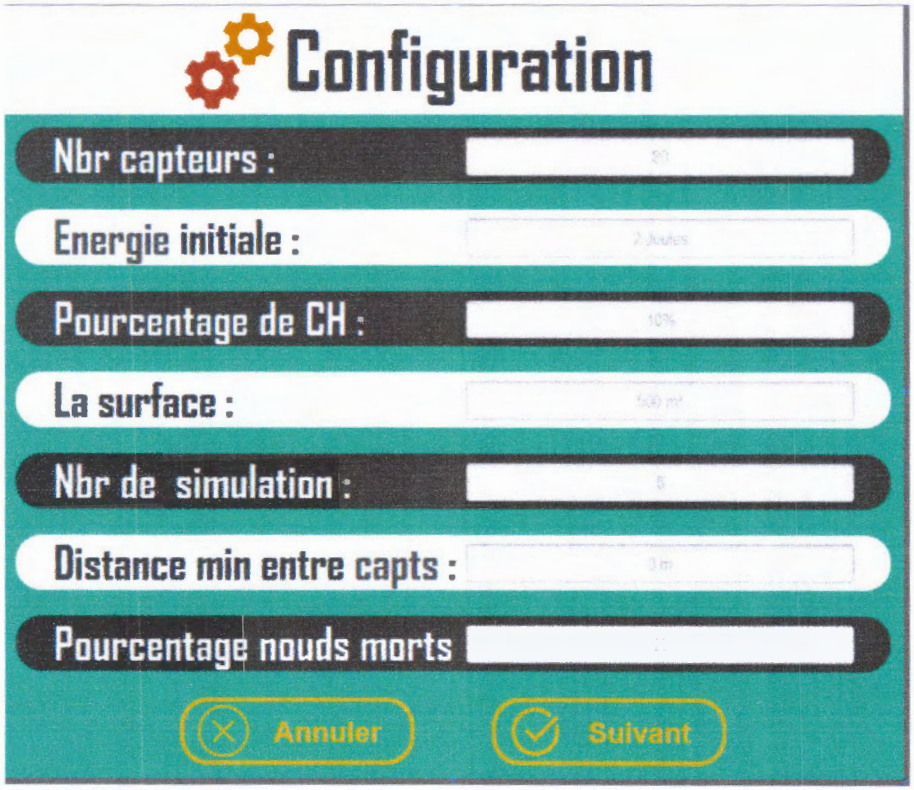


Figure 25 : La fenêtre de configuration

- **Nbr capteurs** : C'est le nombre total de capteurs utilisés en simulation.
- **Energie initiale :** Il s'agit de la valeur de l'énergie initiale et qui est identique pour tous les nœuds avant le début des simulations.
- **Pourcentage de CH :** C'est le pourcentage des cluster-heads parmi le nombre total des nœuds.
- **La surface :** C'est la surface de la zone où les capteurs sont déployés.
- **Nbr simulations :** Représente le nombre de fois où la simulation est répétée.
- **Distance min entre capts :** Représente la distance minimale entre deux capteurs.
- **Pourcentage nœuds mrts :** Représente le pourcentage des nœuds morts pour considérer que le réseau devient mort.

4.3 Hypothèses de base pour les algorithmes de clustering

Concernant les caractéristiques des nœuds nous avons adopté les propriétés suivantes :

- Les nœuds sont déployés aléatoirement.
- Tous les nœuds sont stationnaires.
- L'emplacement de la station de base est fixe au milieu du réseau.
- Tous les nœuds ont la même configuration et la même énergie initiale.
- Tous les nœuds peuvent contrôler leur puissance d'émission pour différentes plages de transmission.
- Tous les nœuds peuvent fonctionner en tant que cluster-head.
- La fusion de données est utilisée pour réduire la taille totale des données envoyées.
- Le pourcentage des cluster-heads utilisé dans notre étude est égale à 10% du nombre des nœuds.

4.4 Environnement et paramètres de simulation

Pour faire la simulation des algorithmes ACO-C et LEACH-C avec notre outil CLUSTERING_APPLICATION, nous avons imposé un modèle expérimental basé sur la distribution de 80 capteurs dans une zone carrée (500*500 m^2), où tous les nœuds ont la même énergie initiale, et ils sont candidats à devenir des CHs. la *Figure 26* suivante représente un exemple de la distribution aléatoire des capteurs dans le réseau.

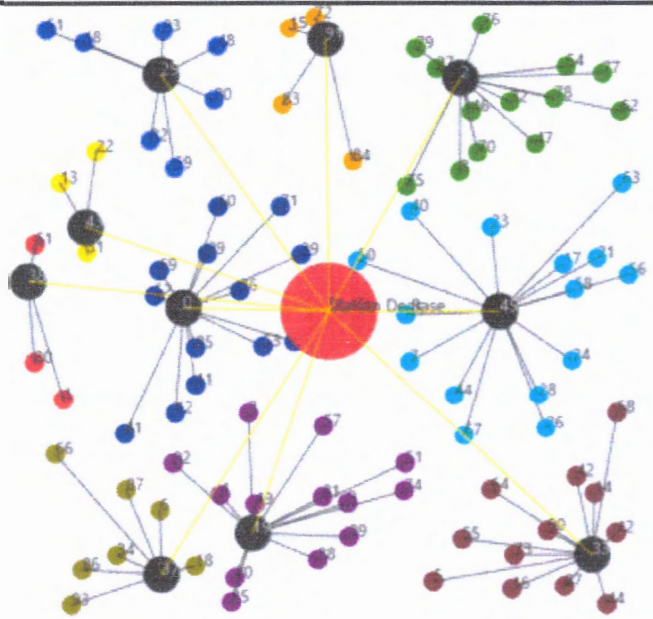


Figure 26 : Une distribution aléatoire des capteurs dans le réseau

Ainsi que, nous allons utilisé les paramètres montrés dans le tableau ci-après :

Paramètre	Valeur
La surface du réseau	500*500 m^2
La position de la station de base	250, 250
Le nombre de nœuds	80
Le pourcentage de Cluster-Heads	10 %
La taille de paquet	525ϕ
Le nombre de paquets envoyés par tour	4
L'énergie initiale	2 Joules

Tableau 2 : *Table des paramètres*

4.5 La structure de données

Dans le côté programmation, nous avons utilisé les structures de données suivantes :

Chaque capteur est representé par un objet de la classe « **Capteur** ».

L'ensemble de tous les capteurs de la topologie est representé par une liste java de types « Capteur » nommé « capts ».

Une solution représentée par une classe Java composeé d'une liste de listes de type capteur « ArrayList<ArrayList<Capteur>> » nommé «Content».

Le premier objet de type capteur dans les sous listes représente le cluster head et les autres objet suivants sont les neouds membres du cluster correspondant.

La classe « **solution** » possède d'autres paramètres et méthodes.

La topologie est représentée par une liste « capts ».

Au départ, une "solution" aléatoire est formée. Ensuite, la méthode « run() » de la classe « fourmi » retourne un ensemble de solutions (ensemble de clusters) candidates, puis la classe «best_clusters» retourne une solution qui représente la meilleure solution selon l'algorithme de colonies de fourmis. Cette solution est utilisée pour le transfert des informations collectées par les capteurs vers la station de base. Les figures (27, 28 , 29) suivanet décrient les structures de données utilisées dans le côté programmation.

```
public class solution {
    private int id;
    public ArrayList<Paint> liste_coleur = new ArrayList<Paint>();
    private ArrayList<ArrayList<capteur>> Content = new ArrayList<ArrayList<capteur>>();
    private double cost;
    public double getMoyen_eng() {
    public void setMoyen_eng(double moyen_eng) {
    private double cost(ArrayList<ArrayList<capteur>> dn) {
    public double eng_consom_env(capteur a, capteur b, double size_data) {
    public double eng_consom_rec(double size_data) {
    public void ameliorer() {
    public int nbr_capts_courant() {

}
```

Figure 27 : La classe solution

```
 *
 * @author user
 */
public class capteur {

    private ArrayList<capteur> bon_rout;
    private double charge //Energie;
    private double firomon;
    private int positionX;
    private int positionY;
    private int id;
    private boolean isClusterHeade = false;
    private capteur ClusterHead;

    double Eelec = 50;//nj
    double Efs = 0.01;//nj
    double Emp = 0.0000013;//nj
    double d0 = 16000;//Math.sqrt(Efs / Emp);//m

    public int getId() {

    public boolean isIsClusterHeade() {

    public void setId(int id) {

    public void setIsClusterHeade(boolean isClusterHeade) {

    public void setClusterHead(capteur ClusterHead) {

    public capteur(double charge, double firomon, int positionX, int positionY, int id) {

    public void setIsclusterhead(int id) {

    public void sende_data(double size_data) {

    private double distance_eqlid(capteur a, capteur b) {

    public void recived_data(double size_data) {
}
```

Figure 28 : La classe capteur

```
/**
 *
 * @author user
 */
public class fourmi {

    public fourmi(int id) {...4 lines }
    Random rnd;
    ArrayList<ArrayList<capteur>> dn;
    int id;
    double cost;

    public void setDn(ArrayList<ArrayList<capteur>> dn) {...3 lines }

    public ArrayList<ArrayList<capteur>> getDn() {...3 lines }

    public int getId() {...3 lines }

    public double getCost() {...3 lines }

    public void setId(int id) {...3 lines }

    public void setCost(double cost) {...3 lines }

    public int changer(int r0) {...4 lines }

    public solution run(int tour) {...107 lines }

    static public void trier(ArrayList< ArrayList<Double>> cost) {...19 lines }

}
```

Figure 29 : La classe fourmi

4.6 Résultats de simulation

Dans cette partie nous allons discuter les résultats obtenus avec notre application CLUSTERING_APPLICATION.

Après l'implémentation de l'algorithme ACO-C [39] original et constatant ses mauvais résultats, nous avons lui apporté plusieurs améliorations.

4.6.1 ACO-C amélioré

Dès le début et avant l'implémentation de l'algorithme ACO-C [39], nous avons remarqué que la fonction objectif qui est la somme de deux fonction *f1* et *f2* n'est pas cohérente car la première fonction est une distance en mètre et la deuxième fonction est un rapport sans unité.

Aussi, la partie principale de la fonction qui met à jour les phéromones dans les capteurs n'est pas adaptée. Cette fonction est un rapport entre la différence entre le coût de la solution courante (agent *l*) et le coût de la solution strictement supérieur au coût de la meilleure solution, à la différence entre le coût moyen de toutes les solutions et le coût de la solution strictement supérieur au coût de la meilleure solution, tout ça dans l'itération courante de l'algorithme, ça signifie que chaque solution est évaluée indépendamment des solutions précédentes. C'est-à-dire elle n'exploite pas l'historique.

$$\Delta\tau_i^l = \begin{cases} g\left(1 - \dfrac{(cost_l - L_B)}{L_{avg} - L_B}\right) & \text{if agent l is used} \\ 0 & \text{otherwise} \end{cases} \qquad (6)$$

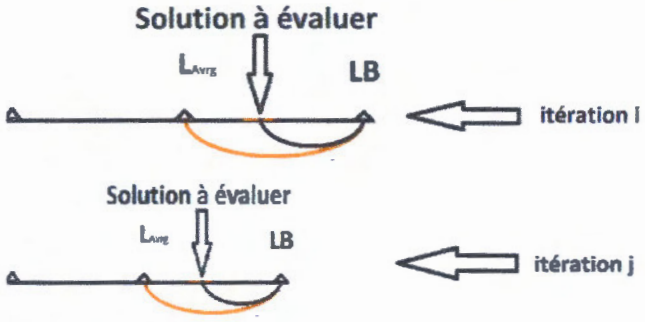


Figure 30 : Exemple 1

Nous avons implémenté la proposition telle qu'elle, comme expliquée, dans [39], et elle a donné de mauvais résultats.

Nous avons travaillé sur la fonction objectif et la fonction de mise à jour des phéromones et nous avons apporté plusieurs améliorations.

Nous avons changé d'abord le LB de la fonction de mise à jour de phéromones, et nous l'avons remplacé par le meilleur coût trouvé par l'algorithme pour garder un historique et évaluer les solutions de façon dépendante entre elles, comme expliqué dans le chéma suivant :

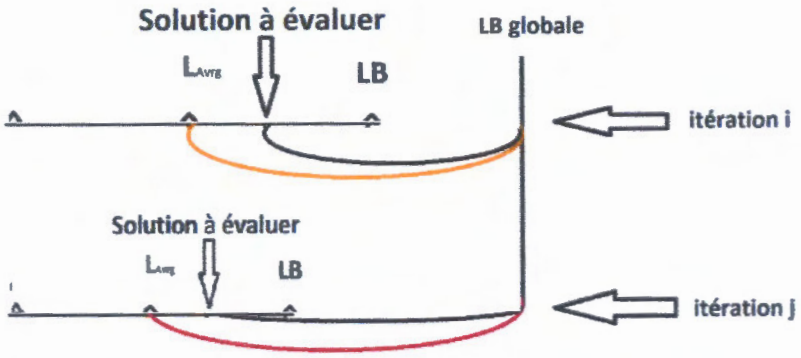


Figure 31 : Exemple 2

Aussi nous avons proposé la fonction objectif expliquée ci-dessous :

Soient :

- NC : Le nombre de clusters dans le réseau.
- C_i : Le cluster i.
- CH_i : Le cluster-head de cluster C_i.
- m_i : Le nombre de nœuds membres du cluster C_i.
- N_{ij} : Le nœud membre j du cluster C_i.

La fonction objectif est égale à la somme des énergies consommées par un cluster (la somme des énergies consommées par les nœuds N_{ij} pour le transfert des informations collectées vers leur cluster-head CH_i et l'énergie consommée par le cluster-head CH_i pour le transfert des informations vers la station de base) en fonction de l'exponentielle et l'énergie moyenne du réseau moins l'énergie du cluster-head CH_i en fonction de l'exponentielle.

Notre choix s'est porté sur la fonction Exponentielle car elle est positive et croissance sur l'intervalle $]-\infty, +\infty[$. Nous avons exploité cette propriété pour comparer l'énergie des cluster-heads et l'énergie moyenne. Et par conséquent garantir que le résultat de la comparaison est toujours positif.

$$cost(S) = \sum_{i=1}^{NC} \left(\begin{matrix} Exp(énergieConsoméePourL'envoi(C_i) + \\ Exp(EnergieMoyenne(Réseau) - EnergieActuelle(CH_i)) \end{matrix} \right)$$

$$énergieConsoméePourL'envoi(C_i) = \sum_{j=1}^{m} \begin{pmatrix} énergieConsoméePourL'envoi(N_{ij}, CH_i) + \\ énergieConsoméePourL'envoi(CH_i, SB) \end{pmatrix}$$

4.6.2 Etude de la durée de vie du réseau selon le pourcentage de nœuds morts

La batterie est un facteur clé dans les réseaux de capteurs sans fil où la durée de vie du réseau en dépend. Il est donc nécessaire d'équilibrer la consommation d'énergie entre les capteurs au sein du réseau. Il existe plusieurs définitions de la durée de vie du réseau : Il y a ceux qui considèrent la mort du réseau dès que le premier nœud est mort. Dans ce cas, la duré de vie du réseau se retrouve très limitée. D'autres considèrent que la mort du réseau est liée à la mort d'un certain pourcentage de capteurs du réseau.

Nous avons mené une étude pour obtenir ce pourcentage minimal des nœuds afin de donner une meilleure performance au réseau (une meilleure durée de vie).

Nous avons effectué plusieurs simulations en modifiant à chaque fois le pourcentage des capteurs morts et avons obtenu les résultats présentés dans la figure suivante :

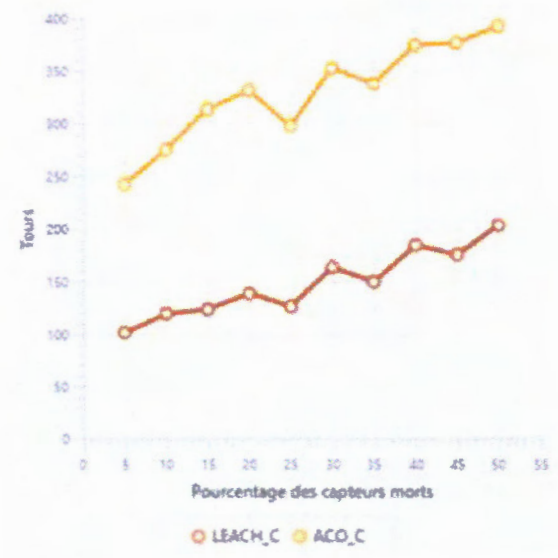


Figure 32 : L'évolution de la durée de vie du réseau selon le pourcentage de nœuds morts

Nbr-tours / Nœuds morts	ACO-C	LEACH-C
5%	240	100
15%	315	120
25%	300	130
35%	340	150
45%	375	175
50%	400	200

Tableau 3 : Durée de vie du réseau selon le pourcentage des nœuds morts

Les résultats obtenus montrent que la durée de vie du réseau augmente en fonction de l'augmentation du pourcentage des nœuds morts.

Dans la suite nous utiliserons le pourcentage de 20%. L'utilisateur peut modifier ce paramètre s'il y a besoin.

4.6.3 Durée de vie du réseau selon la densité

Pour étudier l'impact de la densité (Nombre de nœuds/Surface) sur les performances des protocoles ACO-C, LEACH-C, nous avons procédé successivement à plusieurs simulations en modifiant du nombre de nœuds entre 80 et 130 (sur une même surface de 500m*500m).

La *Figure 33* suivante illustre une courbe représentant l'évolution de la durée de vie du réseau selon la densité.

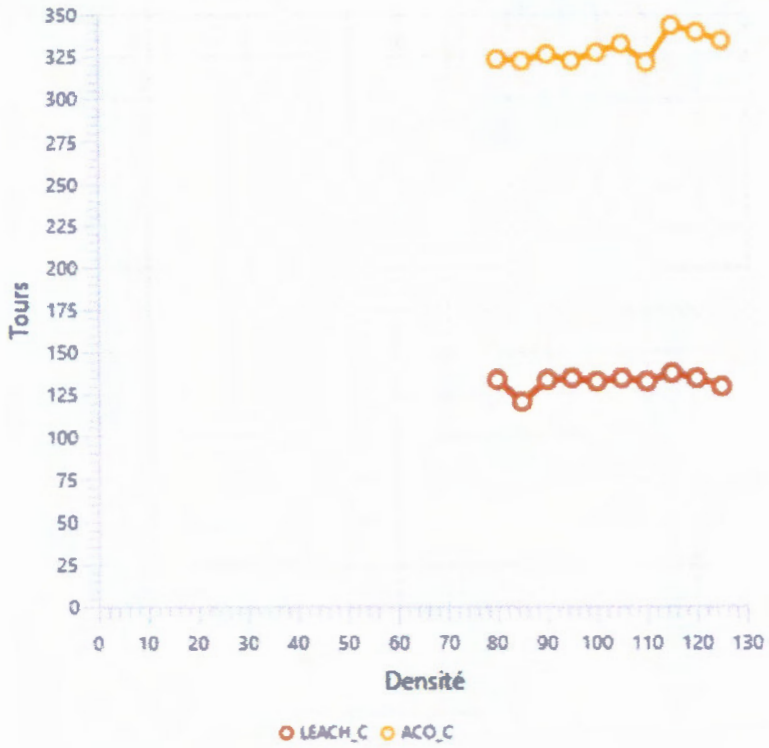


Figure 33 : L'évolution de la durée de vie du réseau selon la densité

Nous remarquons que la densité du réseau n'influence pas la durée de vie du réseau même si ACO-C donne un résultat supérieur par rapport à LEACH-C.

Ce resultat était attendu parce que dans notre application les communications sont assurées à 100%. Les collisions et la saturation du réseau ne sont pas pris en compte.

4.6.4 Durée de vie du réseau selon le nombre de paquets envoyés par tour

Dans cette section, nous avons étudié l'effet du nombre de paquets envoyés à chaque tour sur la vie du réseau. Afin de trouver le nombre optimal de paquets envoyés à chaque tour, ce qui permet de prolonger au maximum la durée de vie du réseau. Nous allons procédé successivement à plusieurs simulations en modifant le nombre de paquets envoyés par tour. La *Figure 34* suivante illustre une courbe représentant l'évolution de la durée de vie du réseau selon le nombre de paquets envoyés par tour. Le *Tableau 4* résume les résultats des simulations.

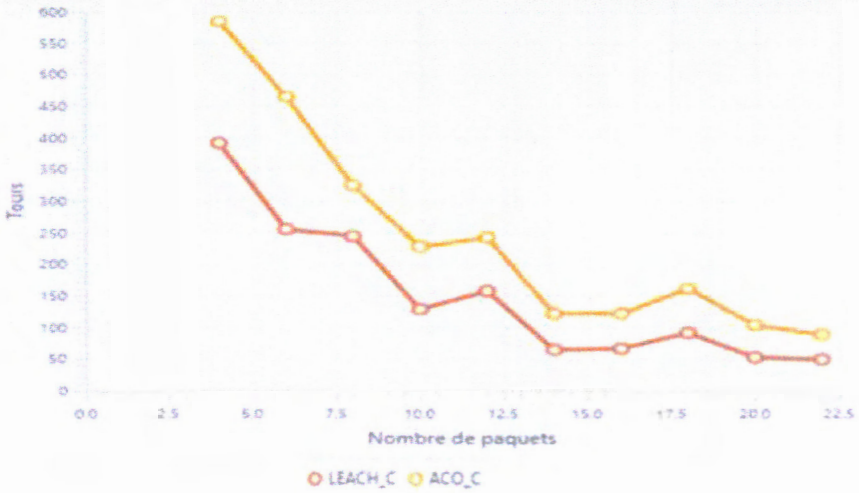


Figure 34 : L'évolution de la durée de vie du réseau selon le nombre de paquets envoyés par tour

Nbr tours / Nbr-paquets-Env	ACO-C	LEACH-C
4	580	390
6	450	260
8	330	240
10	230	130
14	115	60
16	120	70
18	160	90
22	90	45

Tableau 4 : Durée de vie du réseau selon le nombre de paquets envoyé par tours

Nous remarquons que la durée de vie du réseau diminue avec l'augmentation du nombre de paquets envoyés par tour dans les deux algorithmes, en plus nous notons que le protocole ACO-C donne la meilleure duréee de vie. Par contre, si on calcule le nombre de paquets reçus par la SB, on touve qui'il augmente avec l'augmentation du nombre de paquets envoyés par tour. Ce résultat est dû au fait que l'effet de la "période de configuration" diminue par rapport à la "période état-stable".

4.6.5 Comparaison entre les protocoles ACO-C et LEACH C

Dans cette section, nous allons comparer les protocoles ACO-C et LEACH-C en fonction de la durée de vie du réseau définie par le nombre de tours.

Nous allons donc faire 5 simulations consécutives puis nous calculons le nombre moyen du tours obtenu par chaque protocole. Nous utilisons les paramètres mentionnés dans le *Tableau 2* lors de simulations.

4.6.5.1 Comparaison en termes de nœuds en vie dans le réseau

Dans cette section, nous avons étudié l'evolution du nombre de capteurs en vie en fonction du nombre de tours pour les deux protocoles ACO-C et LEACH-C.

Nous avons effectué 5 simulations consécutives avec les mêmes conditions.

Le *Tableau 5* ci-dessous résume les résultats des simulations. et la *Figure 35* donne les courbes représentant le résultat d'une seule simulation.

Nbr de tours / N° simulation	ACO-C	LEACH-C
1	315	135
2	285	120
3	310	115
4	340	145
5	350	150
La moyenne	320	133

Tableau 5 : Résultat de simulation entre ACO-C et LEACH-C en termes du nœuds en vie

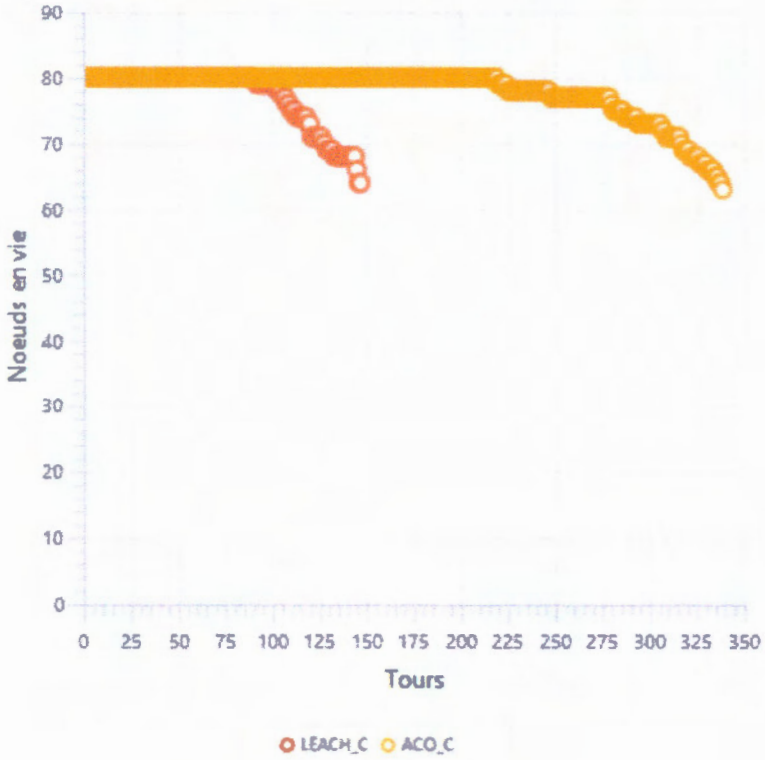


Figure 35 : Le résultat d'une simulation (Nœuds en vie/N° Tours)

Les résultats montrent que ACO-C donne un meilleur résultat que LEACH-C.

En effet, dans ACO-C, le premier nœud meurt après 215 tours, tandis que le premier nœud meurt après 85 tours seulement dans LEACH-C. Donc, ACO-C donne plus que le double de tours que ceux réalisés par LEACH-C.

Ce constat reste valable avec les autres pourcentages de nœuds morts.

Ce résultat est dû à une meilleure distribution des clusters que donne l'algorithme ACO-C.

4.6.5.2 Comparaison en termes d'énergie moyenne du réseau

Dans cette section, nous avons étudié l'evolution de l'énergie moyenne du réseau en fonction du nombre de tours pour les deux protocoles ACO-C et LEACH-C. Nous avons effectué 5 simulations consécutives avec les mêmes conditions.

Le *Tableau 6* résume les résultats des simulations et la *Figure 36* donne les courbes représentant le résultat d'une seule simulation.

Nbr de tours / N° simulation	ACO-C	LEACH-C
1	305	135
2	280	120
3	320	125
4	275	115
5	340	145
La moyenne	304	128

Tableau 6 : Résultat de simulation entre ACO-C et LEACH-C en termes d'énergie moyenne du réseau

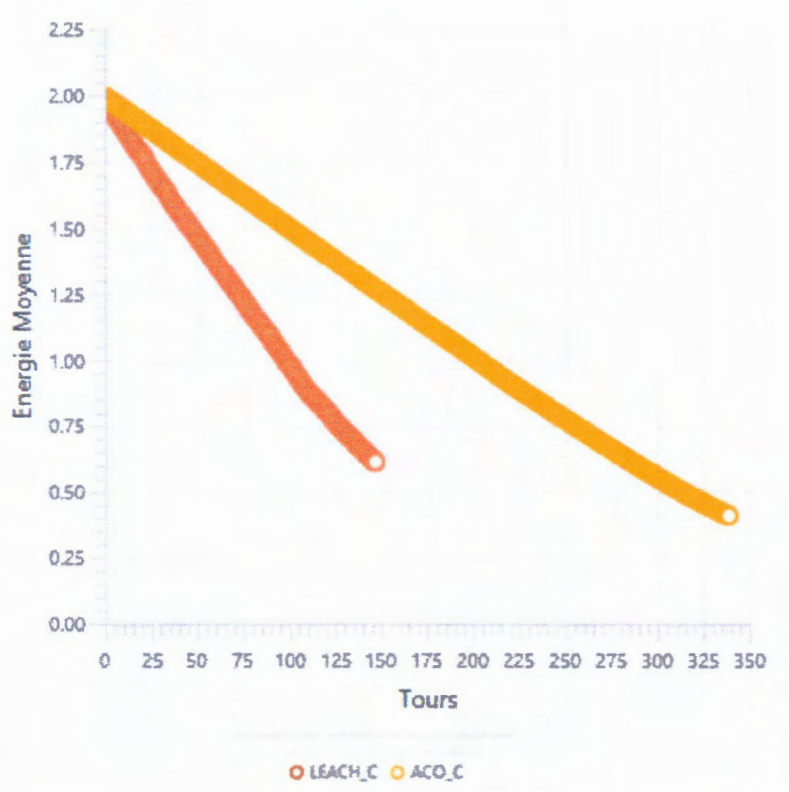


Figure 36 : Le résultat d'une simulation(Energie Moyenne/N° Tours)

Les résultats montrent que ACO-C donne un meilleur résultat que LEACH-C.

Dans ACO-C l'énergie moyenne diminue doucement car l'algorithme permet de choisir une meilleure distribution des clusters et par conséquence diminue l'énergie totale consommée.

4.7 Conclusion

Dans le cadre de ce travail, nous avons implémenté (dans notre outil CLUSTERING_APPLICATION) le protocole ACO-C afin d'évaluer l'effet du clustering basé sur le principe des colonies de fourmis sur la maximisation de la durée de la vie du réseau.

Pour cela nous avons implémenté également le protocole LEACH-C dans notre application afin de le comparer à ACO-C en termes d'efficacité énergétique.

Les résultats de simulation montrent que ACO-C est plus efficace que LEACH-C, où il a donné un nombre de tours plus que le double du nombre de tours donné par LEACH-C.

Cela est dû à l'efficacité de la stratégie de sélection des cluster-heads dans ACO-C où la consommation d'énergie est mieux équilibrée entre les nœuds du réseau.

Conclusion générale

Dans notre travail, nous sommes intéressés à la problématique de la conservation de l'énergie dans un réseau de capteurs sans fil. Dans ce dernier, les noeuds capteurs sont alimentés par des batteries à faible capacité, généralement irremplaçables car les noeuds capteurs sont déployés dans des zones difficilement accessibles.

Afin de prolonger la durée de vie du réseau de capteurs en minimisant la consommation d'énergie, les chercheurs ont proposé plusieurs idées sous des angles divers. Le clustering est la meilleure solution proposée par les chercheurs pour relever les défis de la conservation de l'énergie et de l'augmentation de la durée de vie du réseau.

Le but du clustering est d'organiser les nœuds du réseau dans des clusters équilibrés menés par des nœuds élus appelés cluster-heads, cela permet de réduire le nombre de communications au sein du réseau et économiser la précieuse énergie.

Notre contribution dans ce mémoire consiste à étudier une solution de clustering basée sur le principe de colonies de fourmis ACO-C dont nous avons proposé des améliorations.

Via notre application CLUSTERING_APPLICATION, nous avons mené une étude comparative par simulations des protocoles de routage ACO-C amélioré et LEACH-C afin d'évaluer leurs performances énergétiques

Les résultats de la simulation montrent que le clustering a un effet positif sur le réseau de capteurs et la suprématie de notre algorithme ACO-C modifié par rapport à LEACH-C. Cette évaluation est basée essentiellement sur la durée de vie du réseau.

Ce travail peut être amélioré par :

- ✓ L'implémentation du routage multi-sauts.
- ✓ Comparaison de l'ACO-C avec d'autres algorithmes de populations comme PSO-Particle swarm optimization (Optimisation par essaims particulaires).
- ✓ L'implémentation de ces algorithmes sous un vrai simulateur.

Bibliographie

[1] D. Ibrahima, "Optimisation de la consommation d'énergie par la prise en compte de la redondance de mesure dans les r éseaux de capteurs," 17 07 2014.

[2] M. Salima et D. MAKHMOUKH, «Approche de minimisation de la consommation d'énergie dans les réseaux de capteurs sans fil,» 2017.

[3] M. BOUALLEGUE, «Protocoles de communication et optimisation de l'énergie dans les réseaux de capteurs sans fil,» 2016.

[4] A. Djedjiga et K. Azamoum , «Optimisation de la consommation d'énergie dans les réseaux de capteurs sans fil,» 2014.

[5] F. Abdelfatah, «Développement d'une bibliothèque de capteurs,» 2008.

[6] M. August0, «Survey on Wireless Sensor Network Devices,» 2003.

[7] S. MOAD, «La consommation d'énergie dans les réseaux de capteurs sans fil,» 2008.

[8] R. Panigrahi et K. Sharma, «WIRELESS SENSOR NETWORKS –ARCHITECTURE SéCURITY REQUIREMENTS SECURITY THREATS AND ITS COUNTERMEASURES,» 2013.

[9] J. Zheng et A. Jamalipour, «WIRELESS SENSOR NETWORKS A Networking Perspective,» 2011.

[10] Y. Yousef, «Routage pour la gestion de l'énergie dans les réseaux de capteurs sans fil,» 2011.

[11] L. KHELLADI et N. BADACHE, «Les réseaux de capteurs : état de l'art,» 2004.

[12] M. Asim, «Self-Organization and Management of Wireless Sensor Network,» 2010.

[13] S. Gupta et N. Parveen, «Optimum Node Deployment Strategy for Heterogeneous Wireless Sensor Network by Estimating Network Lifetime,» 2009.

[14] T. Nieberg, S. Dulman, P. Havinga, L. v. Hoesel et J. Wu, «Collaborative algorithms for communication in wireless sensor networks,» 2003.

[15] R. Kacimi, «Techniques de conservation d" energie pour les ŕ eseaux,» 2009.

[16] M. Mohammad et I. Imad, «handbook of sensor networks Compact wireless and wired Sensing Systems,» 2005.

[17] H. Namgoong et D. Lee, «Energy efficient topology for wireless microsensor networks,» n° %1ETRI, Daejeon, Korea, 2005.

[18] H. GUYENNET et J.-N. D. SOUZA, «Optimisation de la durée de vie dans les réseaux de capteurs sans fil sous contraintes de couverture et de connectivité,» n° %1UNIVERSITE CHEIKH ANTA DIOP DE DAKAR, 2016.

[19] G. Chirihane, «Igorithme de routage pour les réseaux de capteurs avec prise en charge de la consommation d'énergie,» 2017.

[20] X.-X. Liu, «A Survey on Clustering Routing Protocols in Wireless Sensor Networks,» 2012.

[21] B. Olutayo, L. Hanh, M. Audrey et T. Makoto , «A Survey on Clustering Algorithms for Wireless Sensor Networks,» 2010.

[22] A. COSTANZO, T. V. LUONG et G. MARILL, «Optimisation par colonies de fourmis,» 2006.

[23] A. A. Abbasi et Y. Mohamed , «A survey on clustering algorithms for wireless sensor networks,» 2007.

[24] B. Ferhat et H. Lamia, «Le routage hiérarchique sous contrainte d'énergie dans les réseaux de capteurs sans fil,» 2014.

[25] A. Makhoul, «localisation, couverture et fusion de données,» 2008.

[26] M. Amina, «Etude comparative entre les deux protocoles de routage LEACH et,» 2015.

[27] S. K. Gupta, n. jain et p. sinha, «Clustering Protocols in Wireless Sensor Networks,» 2013.

[28] Y. K. Tan et S. Winston , «Sustainable Wireless Sensor Networks,» 2010.

[29] A. Kemal et Y. Mohamed, «A survey on routing protocols for wireless sensor networks,» 2005.

[30] C. Laukik, D. Alin et R. Sanjay, «Fault tolerant aggregation in heterogeneous sensor networks,» 2009.

[31] Y. Youcef, «Routage hiérarchique dans les R ́eseaux Capteurs Sans Fil (RCSF) : LEACH et ses variantes,» 2016.

[32] R. Elankavi, R. Udayakumar et Kalaiprasath.R, «A Review on Clustering Algorithms in wirless sensor networks,» 2013.

[33] B.-J. JIAN, J. FAN, D.-C. HUANG et X.-D. CHEN, «An Uneven Clustering Routing Algorithm Based on Energy Load Balancing,» n° %1University, Kunming, China , 2016.

[34] G. Sofiane, «Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs sans fil,» 2016.

[35] B. Juba et M. Adel, «Routage Hiérarchique avec Optimisation de Consommation d'Energie dans les Réseaux de Capteurs Sans-Fil,» 2016.

[36] Y.-L. Chen et J.-S. Lin, «Energy efficiency analysis of a chain-based scheme via intra-grid for wireless sensor networks,» 2012.

[37] A. P. Singh et N. Sharma, «The Comparative Study Of Hierarchical Or Cluster Based Routing Protocol For Wireless Sensor Network,» 2013.

[38] M. Ziyadi, K. Yasami et B. Abolhassani, «Adaptive Clustering for Energy Efficient Wireless Sensor Networks based on Ant Colony Optimization,» 2009.

[39] P. Jalandhar et P. Ferozepur, «Classical and Computational Intelligence Based Routing Protocols for Wireless Sensor Networks,» 2013.

[40] S. Kumar et M.Tech Student, «Optimization of Ant based Cluster Head Election Algorithm in Wireless Sensor Networks,» 2016.

[41] W. Yong et T. Xiaoling, «An Ant Colony Clustering Routing Algorithm for Wireless Sensor Networks,» 2009.

[42] E. Bonabeau et M. Dorigo, «"Inspiration for optimization from social insect behavior,» 2000.

[43] L. Liping, «An Improved Ant Colony Algorithm in,» 2017.

[44] S. PS, J. VK et K. BD, «An ant colony approach for clustering,» 2004.

[45] C. Blum, «Ant colony optimization: Introduction and recent trends,» 2005.

[46] C. Andrea, L. T. Van et M. Guillaume, «Optimisation par colonies de fourmis,» 2006.

[47] M. Dorigo et L. M. Gambardella, «Ant colony system: A cooperative learning approach to the traveling salesman problem,» 1997.

[48] M. Dorigo et t. Stützle, «Ant colony optimization,» 2004.

[49] M. Dorigo, V. Maniezzo et A. Colorni, «The Ant System: Optimization by a colony of cooperating agents,» 1996.

[50] T. Stutzle et H. Hoos, «AX-MZN Ant System and Local Search for the Traveling Salesman Problem,» 1997.

[51] V. Maniezzo, «Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem,» 1999.

[52] B. Ahmed, «Application des algorithmes de colonies de fourmis pour l'optimisation et la classification des images,» 2013.

[53] Marco Dorigo, Mauro Birattari et T. Stutzle, «Ant Colony Optimization,» 2006.

[54] R. Schoonderwoerd, O. Holland et J. Bruten, «Antbased load balancing in telecommunications networks,» 1996.

[55] G. D. Caro et M. Dorigo, «AntNet: Distributed stigmergetic control for Communications Networks,» 1998.

[56] K. M. Sim et W. H. Sun, «Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions,» 2003.

[57] G. D. Caro, «Ant colony optimization and its application to adaptive routing in telecommunication networks,» 2004.

[58] FREDERICK DUCATELLE, G. D. CARO et L. M. GAMBARDELLA, «Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks,» 2005.

[59] M. Gravela, W. L.Priceb et C. Gagnéa, «MarcGravelaWilson L.PricebCarolineGagnéa using a multiple objective ant colony optimization metaheuristic,» 2002.

[60] J. Bautista et J. Pereira, «Ant Algorithms for Assembly Line Balancing,» 2002.

[61] E. TAILLARD, «Programmation à mémoire adaptative et algorithmes pseudogloutons: nouvelles perspectives pour les méta-heuristiques,» 1999.

[62] V. Maniezzo, «The Ant System Applied to the Quadratic Assignment Problem. Technical Report,» 1994.

[63] D. COSTA, «Ants Can Color Graphs, Journal of the Operational,» 1997.

[64] C. Solnon, «Solving Permutation Constraint Satisfaction Problems with Artificial Ants,» 2000.

[65] R. S. Parpinelli, «a Heuristic Approach, chapter An Ant Colony Algorithm for Classification Rule Discovery,» 2002.

[66] Y. Semet, E. Lutton et P. Collet, «Ant Colony Optimisation for E-Leaming : Observing the Emergence of Pedagogic Suggestions,» 2003.

[67] J. Dréo, «Adaptation de la métaheuristique des colonies de fourmis pour l'optimisation difficile en variables continues. Application en génie biologique et médical.,» 2006.

[68] W. B. Heinzelman, A. P. Chandrakasan et H. Balakrishnan, «An Application-Specific Protocol Architecture forWireless Microsensor Networks,» 2002.

[69] A. S. Zahmati, B. Abolhassani, A. A. B. Shirazi et A. S. Bakhtiari, «an-energy-efficient-protocol-with-static-clustering-for-wireless-sensor-networks,» 2007.

[70] W. R. Heinzelman, A. Chandrakasan et H. Balakrishna, «Energy-Efficient Communication Protocol for Wireless Microsensor Networks,» 2000.

[71] N. M. A. Latiff, C. C. Tsimenidis et B. S. Sharif, «ENERGY-AWARE CLUSTERING FOR WIRELESS SENSOR NETWORKS USING PARTICLE SWARM OPTIMIZATION,» 2007.

[72] J. Tillet, R. Rao et F.Sahin, «Cluster-head identification in ad hoc sensor networks using particle swarm optimization,» chez *IEEE International Conference on Personal Wireless Communications*, 2002.

[73] M. Tripathi, M. Gaur, V. Laxmi et R. Battula, «ENERGY EFFICIENT LEACH-C PROTOCOL FOR WIRELESS SENSOR NETWOTK,» 2001.

[74] A. Parmar et A. Thakkar, «An improved modified LEACH-C algorithm for energy efficient routing in Wireless Sensor networks,» 2015.

[75] N. Metropolis, A. .. Rosenbluth, M. .. Rosenbluth, A. Teller et E. Teller, «Equation of State Calculations by Fast Computing Machines,» 1953.

[76] W. B. Heinzelman, A. P. Chandrakasan et H. Balakrishnan, «an application-specific protocol architecture for wireless micro sensor Networks,» 2002.

[77] «java,» [En ligne]. Available: https://www.java.com/fr/download/faq/whatis_java.xml.

[78] «DZone/Java Zone,» [En ligne]. Available: https://dzone.com/articles/what-does-javafx-mean-you.

[79] «TheServerSide/javaFx,» 2019. [En ligne]. Available: https://www.theserverside.com/definition/JavaFX.

[80] «javaworld,» [En ligne]. Available: https://www.javaworld.com/article/3296360/what-is-the-jdk-introduction-to-the-java-development-kit.html.

[81] «TheServerSide,» [En ligne]. Available: https://www.theserverside.com/definition/Java-Development-Kit-JDK.

[82] B. Amira, «DOCZZ,» 2015. [En ligne]. Available: http://doczz.fr/doc/910481/prolongation-de-la-durée-de-vie-des-batteries-dans-les-ré.

[83] S. Kabou, «état de l'art sur les réseaux de capteurs sans fil,» 2010.

[84] H. d. N. Alves, B. A. d. Souza et H. A. Ferreira, «Banks of automatic capacitors in electrical distribution systems a hybrid algorithm of control,» 2005.

[85] D. M. M. Middendorf et a. H. Schmeck, «Ant Colony Optimization for Resource-Constrained Project Scheduling,» 2002.

[86] M. BOUALLEGUE, «Protocoles de communication et optimisation de l'énergie dans les réseaux de capteurs sans fil,» 2016.

[87] A. Djedjiga et K. Azamoum , «Optimisation de la consommation d'énergie dans les réseaux de capteurs sans fil,» 2014.

[88] A. Baptiste, «Les métaheuristiques en optimisation combinatoire,» 2006.

[89] B. Amine, «Le recuit simulé,» 2011.

[90] M. Salima et D. MAKHMOUKH, «Approche de minimisation de la consommation d'énergie dans les réseaux de capteurs sans fil,» n° %1Université A/Mira de Béjaïa, 2017.

[91] D. Ibrahima, "Optimisation de la consommation d'énergie par la prise en compte de la redondance de mesure dans les r éseaux de capteurs," no. L'UNIVERSIT DE TOULOUSE, 17 07 2014.