République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la recherche scientifique

Université de l'Hadj Lakhdar-Batna
Faculté des sciences de l'ingénieur
Département d'informatique

MEMOIRE DE MAGISTERE EN INFORMATIQUE

Option : Informatique industrielle

Intitulé

Protocole de routage pour les réseaux de capteurs sans fil

Travail réalisé par : ***Mr Boubiche Djallel Eddine***

Sous la direction de : ***Dr Bilami Azeddine***

Membre de Jury composé de *:*

- Pr. Benmohamed Mohamed : Président des jurys (Université de Constantine).
- Dr. Babahenini Mohamed Chaouki : Examinateur (Université de Biskra).
- Dr. Zidani Abdelmadjid : Examinateur (Université de Batna).
- Dr. Bilami Azeddine : Rapporteur (Université de Batna).

Promotion 2007-2008

Résumé-

Le réseau de capteurs sans fil (RCSF) est une technologie émergente qui vise à offrir des capacités innovantes. Leur utilisation ne devrait cesser d'augmenter et ceci dans de nombreux domaines qu'ils soient scientifiques, logistiques, militaires ou encore sanitaires. Cependant, la taille des capteurs constitue une limitation importante, principalement en terme d'autonomie d'énergie et donc de durée de vie car la batterie doit être très petite, c'est pourquoi de nombreux travaux portent aujourd'hui sur la gestion de l'énergie consommée par les capteurs dans un réseau en prenant en considération, en premier lieu, les communications et les algorithmes de routage des données. C'est dans ce but que nous avons proposé un algorithme de routage adaptatif qui est une combinaison de deux grandes approches de routage à savoir (Cluster-based approach) et (Chain-based approach). Afin de confirmer les améliorations apportées par notre algorithme nous avons conduit une simulation à laide du simulateur réseau NS2, dans la quelle les performances de notre algorithme sont évaluées et comparée avec les protocoles de clustering existant (LEACH et LEACH-C) ainsi que des protocoles de chaînage (PEGASIS) utilisés dans les réseaux de capteurs.

Mots clés: WSN, efficacité d'énergie, LEACH, LEACH-C, PEGASIS, approche basée cluster, approche basée chaîne, protocole de routage hiérarchique

Abstract-

Wireless sensors network (WSN) is an emergent technology which aims at offering innovating capacities. Their use should not cease increasing and this in many fields such as scientist, logistic, military or medical. One of the weaknesses of wireless sensors networks is the limit of energy which affects network's lifetime. This is why many works concern today the energy management consumed by the sensors in a network while taking into account, initially, the communications and the data routing algorithms. It is to this end that we proposed a new routing approach which is a combination of two great approaches of routing to knowing (Cluster-based approach) and (Chain-based approach). In our proposal, nodes belonging to the same cluster form a chain where nodes communicate only with their closest neighbors, so that energy dissipation within

clusters can be minimized and consequently, lifetime of the network can be improved. Simulations using Network Simulator NS2 have been conducted in order to confirm this assumption. Performances of the routing algorithm are evaluated and compared with the existing clustering protocols (LEACH and LEACH-C) as well as chaining protocols (PEGASIS) used in the sensors networks.

Keywords: WSN, energy efficiency, LEACH, LEACH-C, PEGASIS, Cluster based approach, chain based approach, hierarchical routing protocol.

Table des matières

- Introduction générale…….3

Chapitre 1 : «***Généralités sur les réseaux de capteurs sans fil*** »

1. Introduction…….4
2. Architecture d'un micro capteur …..……5
   - 2.1 L'unité de captage ……...……6
   - 2.2 L'unité de traitement ……...……6
   - 2.3 L'unité de transmission ……...6
   - 2.4 L'unité de contrôle d'énergie ……...7
3. Applications des réseaux de capteurs …….....7
   - 3.1 Applications militaires ……....7
   - 3.2 Applications à la sécurité ……8
   - 3.3 Applications environnementales …… 9
   - 3.4 Applications médicales ……10
   - 3.5 Applications commerciales ……...11
4. Besoins et facteurs de conception dans un réseau de capteurs sans fil ……..….12
   - 4.1 Tolérance aux fautes, adaptabilité et fiabilité……..…....12
   - 4.2 Gestion et consommation d'énergie ……..12
   - 4.3 Efficacité du réseau et agrégation de données ……..14
   - 4.4 Routage Intelligent….. ……...14
   - 4.5 Le défi de gestion…….. ……....14
5. Conclusion ……. 15

Chapitre 2 : «***Etat de l'art*** »

1. Introduction……16
2. Les algorithmes de routage dans RCSFs ……...16
   - 2.1 Les protocoles données centrales (Data- centric protocols) ……....17
   - 2.2 Les protocoles basés sur la localisation (géographique)…….……..…..19
   - 2.3 Les protocoles hiérarchiques ……..…….... 20
     - 2.3.1 Cluster-based approach (approche à grappe) ……..…..21
       - 2.3.1.1 LEACH (Low Energy Adaptive Clustering Hierarchy)……22
     - 2.3.2 Chain-based approach (approche à chaîne)…….. 27
       - 2.3.2.1 PEGASIS (Power-Efficient Gathering in Sensor Information Systems)...27
     - 2.3.3 Comparaison des protocoles (LEACH et PEGASIS) à l'aide de la simulation….30
3. Motivation et objectives ……35
4. Conclusion ……..…… 37

Chapitre 3 : «***Algorithme proposé*** »

1. Introduction ...……38
2. Concept de base de notre protocole .....……...38
3. Approche de formation de grappes à chaînes ……..…...40
4. Les grandes étapes de notre algorithme……. ……..……...45
   - 4.1 Etape d'initialisation ……..……...46
   - 4.2 Etape de transmission ……..……....48
5. Conclusion ……...…… 51

Chapitre 4 : «***Implémentation***»

1. Introduction ..…….. 52
2. Choix du langage et de l'environnement d'implémentation ..…….. 52
3. Etapes d'implémentation de notre protocole ……... 53
   - 3.1 Préparation de l'environnement d'implémentation ……..... 53
   - 3.2 Implémentation de notre protocole hybride ……..…. 55
     - 3.2.a Procédure de formation de chaîne…….55
     - 3.2.b Procédure de recherche du nœud le plus loin de la chaîne……..... 58
     - 3.2.c Procédure de recherche du nœud le plus proche ……. 59
     - 3.2.d Procédure de transmission de données ……..…...……60
     - 3.2.e Procédure de réception de données …….……62
     - 3.2.f Procédure de réception de données par le cluster head……62
4. Conclusion …….……63

Chapitre 5 : «***Evaluation des performances à travers la simulation***»

1. Introduction……64
2. Environnement de simulation…….. ..……...64
3. Résultats de la simulation…… ……....66
4. Conclusion ……71

- Conclusion générale…….72
- Références ……...…73

Introduction générale

Les progrès réalisés lors de ces dernières décennies dans les domaines de la microélectronique, de la micromécanique, et des technologies de communication sans fil, ont permis de produire avec un coût raisonnable des composants de quelques millimètres cubes de volume. Ces derniers, appelés micro-capteurs, intègrent : une unité de captage chargée de capter des grandeurs physiques (chaleur, humidité, vibrations) et de les transformer en grandeurs numériques, une unité de traitement informatique et de stockage de données et un module de transmission sans fil. Un grand nombre de ces dispositifs (micro-capteurs) sont déployés dans la nature afin de créer un réseau de capteurs à des fins aussi bien de contrôle que de monitorisation. Le fort potentiel d'applications des réseaux de capteurs en font un domaine de recherche très actif.

Un micro-capteur est muni d'une ressource énergétique (généralement une batterie) pour alimenter tous ses composants. Cependant, en raison de sa taille réduite, la ressource énergétique dont il dispose est limitée et généralement irremplaçable. Dès lors, l'énergie est la ressource la plus précieuse dans un réseau de capteurs, puisque elle influe directement sur la durée de vie des micros capteurs, voire du réseau en entier, étant donnée que le routage de donnée est un facteur déterminant dans la gestion économique d'énergie plusieurs recherche on était effectuée afin de proposer des stratégies de routages dont certaines sont des adaptations de stratégies qui existaient pour d'autres types de réseaux (réseau ad hoc) tandis que d'autres ont été conçues spécialement pour les réseaux de capteurs sans fil. L'objectif de notre thèse est d'étudier et d'analyser la plus part des ces stratégie, repérer leurs inconvénients et déduire leurs avantages afin de proposer notre propre protocole de routage.

Le reste du mémoire sera organisé comme suit : En premier lieu, on introduira en présentant quelques généralités sur les réseaux de capteurs sans fil. Ensuite, on présentera un état de l'art sur les protocoles de routage dans les RCSFs ainsi que nos motivations. Après, on proposera notre protocole et ses grandes étapes. On va détailler, par la suite le code d'implémentation de notre protocole. Après, on va simuler le fonctionnement de notre algorithme afin d'évaluer ses performances. Enfin, On terminera avec une conclusion générale.

1. Introduction

Les réseaux de capteurs sans fil [1] (RCSFs ou WSNs : Wireless sensor networks en anglais) sont devenus de plus en plus omniprésents. Les milieux scientifiques et industriels leurs prêtent de en plus d'attention du fait de leurs riches applications dans les domaines : médical, commercial et militaire. Selon MIT's Technology Review, il s'agit de l'une des dix nouvelles technologies qui vont influer sur notre manière de vivre et de travailler. Les RCSFs sont des réseaux de nœuds sans fil dédiés à des applications spécifiques. Ils sont considérés comme un type particulier des réseaux Ad-hoc, dans lesquels les nœuds sont des capteurs intelligents (smart sensors). Les RCSFs sont composés d'un nombre potentiellement très grand (plusieurs milliers) de capteurs qui se communiquent selon un modèle de communication « sources multiples - destination unique », déployés dans la zone à couvrir. Chaque capteur est capable d'effectuer d'une manière autonome trois tâches complémentaires: mesure d'une valeur physique**,** traitement de ses mesures, et communication par voie hertzienne.

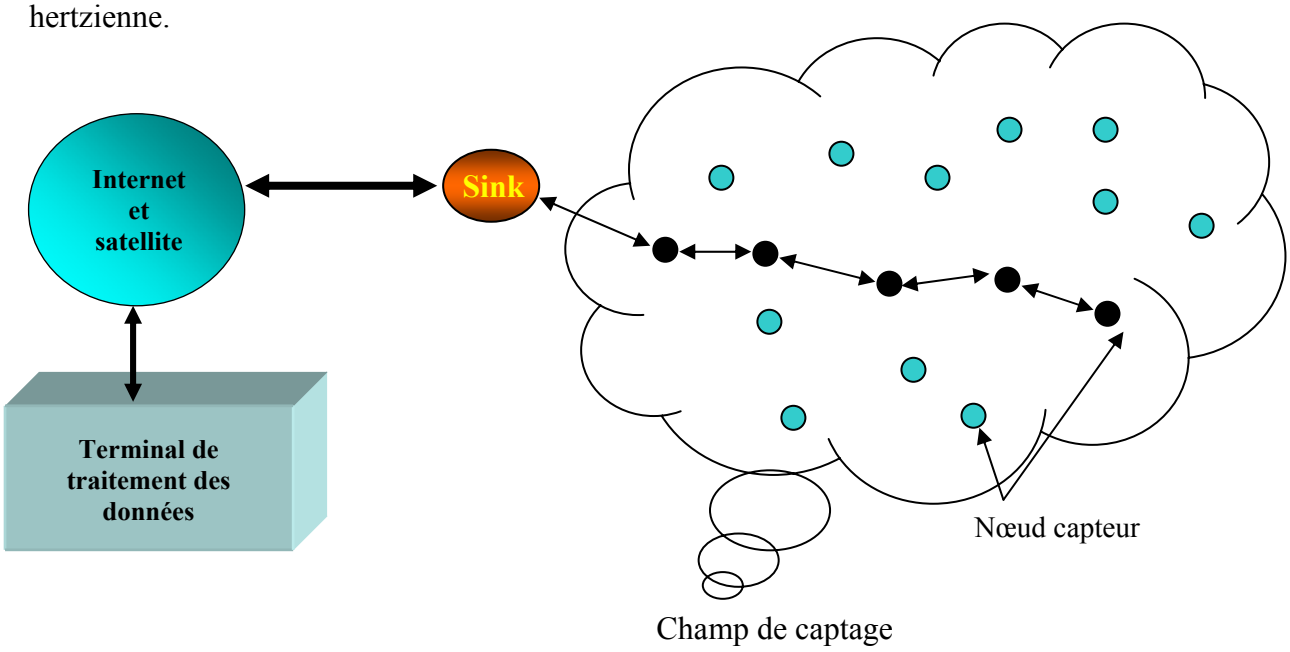


***Figure 1** : Réseau de capteur sans fil (WSN)*

Les capteurs sont des objets limités en terme de bande passante, de puissance de calcul, de mémoire disponible et d'énergie embarquée. La position de ces nœuds n'est pas

obligatoirement prédéterminée. Ils sont dispersés aléatoirement à travers une zone géographique, appelée champ de captage, qui définit le terrain d'intérêt pour le phénomène capté. Les données captées sont acheminées grâce à un routage multi-sauts à un nœud considéré comme un "point de collecte", appelé nœud puits (sink, ou station de base). Ce dernier peut être connecté à l'utilisateur du réseau via Internet ou un satellite. Ainsi, l'usager peut adresser des requêtes aux autres nœuds du réseau, précisant le type de données requises et récoltant les données environnementales captées par le biais du nœud puits.

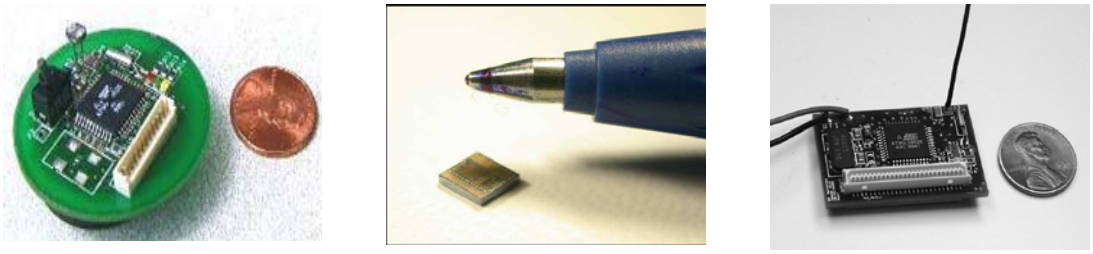


***Figure 2** : Exemple de capteur sans fil.*

Les réseaux de capteurs sans fil sont typiquement employés dans les environnements fortement dynamiques et hostiles sans existence humaine (à la différence des réseaux informatiques conventionnels), et donc, ils doivent être tolérants à l'échec (avec une participation humaine minime) et à la perte de connectivité.

2. Architecture d'un micro capteur

Un nœud capteur contient quatre unités de base : l'unité de captage, l'unité de traitement, l'unité de transmission, et l'unité de contrôle d'énergie. Il peut contenir également, suivant son domaine d'application, des modules supplémentaires tels qu'un système de localisation (GPS), ou bien un système générateur d'énergie (cellule solaire). On peut même trouver des micro-capteurs, un peu plus volumineux, dotés d'un système mobilisateur chargé de déplacer le micro-capteur en cas de nécessité

2. 1. L'unité de captage

L'unité de captage est généralement composée de deux sous-unités : le capteur lui-même et un convertisseur Analogique/Numérique. 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.

2. 2. L'unité de traitement

L'unité de traitement comprend un processeur associé généralement à une petite unité de stockage et fonctionne à l'aide d'un système d'exploitation spécialement conçu pour les micro-capteurs (TinyOS par exemple). Cette unité est chargée d'exécuter les protocoles de communication 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.

2. 3. L'unité de transmission

Cette unité est responsable d'effectuer toutes les émissions et réceptions des données sur un medium sans fil. Elle peut être de type optique (comme dans les nœuds Smart Dust), ou de type radio-fréquence. Les communications de type optique sont robustes vis-à-vis des interférences électriques. Néanmoins, elles présentent l'inconvénient d'exiger une ligne de vue permanente entre les entités communicantes. Par conséquent, elles ne peuvent pas établir de liaisons à travers des obstacles. Les unités de transmission de type radio-fréquence comprennent des circuits de modulation, démodulation, filtrage et multiplexage; ce qui implique une augmentation de la complexité et du coût de production du micro-capteur. Concevoir des unités de transmission de type radio-fréquence avec une faible consommation d'énergie est un véritable défi. En effet, pour qu'un nœud ait une portée de communication suffisamment grande, il est nécessaire d'utiliser un signal assez puissant. Cependant, l'énergie consommée serait importante. L'autre alternative serait d'utiliser de longues antennes, mais ceci n'est pas possible à cause de la taille réduite des micro-capteurs

2. 4. L'unité de contrôle d'énergie

Un micro-capteur est muni d'une ressource énergétique (généralement une batterie) pour alimenter tous ses composants. Cependant, en conséquence de sa taille réduite, la ressource énergétique dont il dispose est limitée et est généralement irremplaçable. Dès lors, l'énergie est la ressource la plus précieuse dans un réseau de capteurs, puisque elle influe directement sur la durée de vie des micro-capteurs, voire du réseau en entier. L'unité de contrôle d'énergie constitue donc l'un des systèmes les plus importants. Elle est responsable de répartir l'énergie disponible aux autres modules et de réduire les dépenses en mettant en veille les composants inactifs par exemple. Cette unité peut aussi gérer des systèmes de rechargement d'énergie à partir de l'environnement observé telles que les cellules solaires, afin d'étendre la durée de vie totale du réseau.

3. Applications des réseaux de capteurs

La taille de plus en plus réduite des micro-capteurs, le coût de plus en plus faible, la large gamme des types de capteurs disponibles (thermique, optique, vibrations,...) ainsi que le support de communication sans fil utilisé, permettent aux réseaux de capteurs d'envahir plusieurs domaines d'applications. Ils permettent aussi d'étendre les applications existantes et de faciliter la conception d'autres systèmes tels que le contrôle et l'automatisation des chaînes de montage. Les réseaux de capteurs ont le potentiel de révolutionner la manière même de comprendre et de construire les systèmes physiques complexes. Les réseaux de capteurs peuvent se révéler très utiles dans de nombreuses applications lorsqu'il s'agit de collecter et de traiter des informations provenant de l'environnement. Parmi les domaines où ces réseaux peuvent offrir les meilleures contributions, nous citons les domaines : militaire, environnemental, domestique, santé, sécurité, etc. Des exemples d'applications potentielles dans ces différents domaines sont exposés ci-dessous.

3.1. Applications militaires

Comme dans le cas de plusieurs technologies, le domaine militaire a été un moteur initial pour le développement des réseaux de capteurs. Le déploiement rapide, le coût réduit, l'auto-

organisation et la tolérance aux pannes des réseaux de capteurs sont des caractéristiques qui rendent ce type de réseaux un outil appréciable dans un tel domaine. Comme exemple

Figure 3 : *réseau de capteurs militaire.*

d'application dans ce domaine, on peut penser à un réseau de capteurs déployé sur un endroit stratégique ou difficile d'accès, afin de surveiller toutes les activités des forces ennemies, ou d'analyser le terrain avant d'y envoyer des troupes (détection d'agents chimiques, biologiques ou de radiations). Des tests concluants ont déjà été réalisés dans ce domaine par l'armée américaine dans le désert de Californie.

3.2. Applications à la sécurité

Figure 4 *: Applications à la sécurité.*

Les altérations dans la structure d'un bâtiment, suite à un séisme ou au vieillissement, pourraient être détectées par des capteurs intégrés dans les murs ou dans le béton, sans alimentation électrique ou autres connexions filaires. Les capteurs doivent s'activer périodiquement et peuvent ainsi fonctionner durant des années, voire des décennies. Un réseau de capteurs de mouvements peut constituer un système d'alarme distribué qui servira à détecter les intrusions sur un large secteur. Déconnecter le système ne serait plus aussi simple, puisque il n'existe pas de point critique. La surveillance de voies ferrées pour prévenir des accidents avec des animaux et des êtres humains peut être une application intéressante des réseaux de capteurs. La protection des barrages pourrait être accomplie en y introduisant des capteurs. La détection de fuites d'eau permettrait d'éviter des dégâts. Les êtres humains sont conscients des risques et attaques qui les menacent. Du coup, ils mettent à disposition toutes les ressources humaines et financières nécessaires pour leur sécurité. Cependant, des failles sont toujours présentes dans les mécanismes de sécurisation appliqués aujourd'hui, sans oublier leur coût très élevé. L'application des réseaux de capteurs dans le domaine de la sécurité pourrait diminuer considérablement les dépenses financières consacrées à la sécurisation des lieux et à la protection des êtres humains tout en garantissant de meilleurs résultats.

3.3. Applications environnementales

Figure 5 : *applications environnementales.*

Des thermo-capteurs dispersés à partir d'un avion sur une forêt peuvent signaler un éventuel début d'incendie dans le champ de captage; ce qui permettra une meilleure efficacité pour la lutte contre les feux de forêt. Dans les champs agricoles, les capteurs peuvent être semés avec les graines. Ainsi, les zones sèches seront facilement identifiées et l'irrigation sera donc plus efficace. Sur les sites industriels, les centrales nucléaires ou dans les pétroliers, des capteurs peuvent être déployés pour détecter des fuites de produits toxiques (gaz, produits chimiques, éléments radioactifs, pétrole, etc.) et alerter les utilisateurs dans un délai suffisamment court pour permettre une intervention efficace. Une grande quantité de capteurs peut être déployée en forêt ou dans un environnement de conservation de la faune afin de recueillir des informations diverses sur l'état du milieu naturel et sur les comportements de déplacement. Par exemple, l'université de Pise en Italie a réalisé des réseaux de capteurs pour le contrôle des parcs naturels (feux, animaux,..). Il est ainsi possible "d'observer", sans déranger, des espèces animales difficiles à étudier dans leur environnement naturel et de proposer des solutions plus efficaces pour la conservation de la faune. Les éventuelles conséquences de la dispersion en masse des micro-capteurs dans l'environnement ont soulevé plusieurs inquiétudes. En effet, chaque micro-capteur est doté d'une batterie qui contient des métaux nocifs. Néanmoins, le déploiement d'un million de capteurs de 1 millimètre cube chacun ne représente qu'un volume total d'un litre. Même si tout ce volume était constitué de batteries, cela n'aurait pas des répercussions désastreuses sur l'environnement.

3.4. Applications médicales

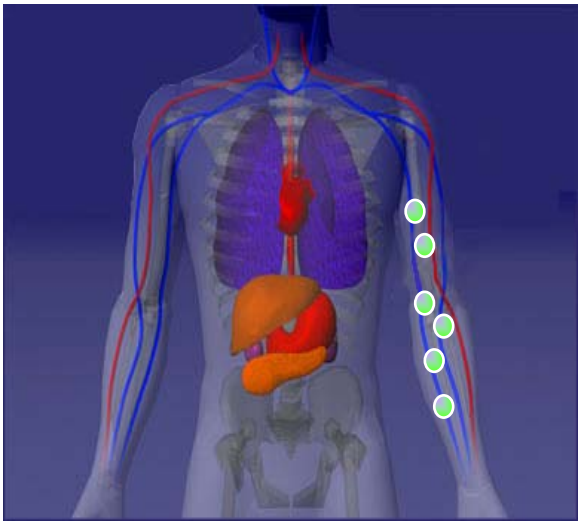


***Figure 6** : ensemble de capteurs dans un corps humain.*

On pourrait imaginer que dans le futur, la surveillance des fonctions vitales de l'être humain serait possible grâce à des micro-capteurs qui pourront être avalés ou implantés sous la peau. Actuellement, des micro-caméras qui peuvent être avalées existent. Elles sont capables, sans avoir recours à la chirurgie, de transmettre des images de l'intérieur d'un corps humain avec une autonomie de 24 heures. Les auteurs d'une récente étude, présentent des capteurs qui fonctionnent à l'intérieur du corps humain pour traiter certains types de maladies. Leur projet actuel est de créer une rétine artificielle composée de 100 micro-capteurs pour corriger la vue. D'autres ambitieuses applications biomédicales sont aussi présentées, tel que : la surveillance du niveau de glucose, le monitoring des organes vitaux ou la détection de cancers. L'utilisation des réseaux de capteurs dans le domaine de la médecine pourrait apporter une surveillance permanente des patients et une possibilité de collecter des informations physiologiques de meilleure qualité, facilitant ainsi le diagnostic de quelques maladies.

3.5. Applications commerciales

Il est possible d'intégrer des nœuds capteurs au processus de stockage et de livraison. Le réseau ainsi formé, pourra être utilisé pour connaître la position, l'état et la direction d'un paquet ou d'une cargaison. Il devient alors possible pour un client qui attend la réception d'un paquet, d'avoir un avis de livraison en temps réel et de connaître la position actuelle du paquet. Pour les entreprises manufacturières, les réseaux de capteurs permettront de suivre le procédé de production à partir des matières premières jusqu'au produit final livré. Grâce aux réseaux de capteurs, les entreprises pourraient offrir une meilleure qualité de service tout en réduisant leurs coûts. Dans les immeubles, le système de climatisation peut être conçu en intégrant plusieurs micro-capteurs dans les tuiles du plancher et les meubles. Ainsi, La climatisation pourra être déclenchée seulement aux endroits où il y a des personnes présentes et seulement si c'est nécessaire. Le système distribué pourra aussi maintenir une température homogène dans les pièces. Utilisée à grande échelle, une telle application permettrait de réduire la demande mondiale en énergie réduisant du même coup les gaz à effet de serre. Rien que pour les Etats-Unis, on estime cette économie à 55 milliards de dollars par an avec une diminution de 35 millions de tonnes des émissions de carbone dans l'air. Ainsi, dans un contexte mondial où le réchauffement de la planète devient une préoccupation grandissante, une telle conséquence environnementale serait un pas dans la bonne direction.

4. Besoins et facteurs de conception dans un réseau de capteurs sans fil

Dans ce qui suit, on présentera certains des besoins de base et des facteurs de conception du réseau de capteurs sans fil, qui servent de directives au développement des protocoles et des algorithmes pour l'architecture de communication WSN.

4.1. Tolérance aux fautes, adaptabilité et fiabilité : Les réseaux de capteurs sont requis pour fonctionner en s'adaptant aux changements environnementaux que les capteurs contrôlent. Les réseaux devraient être *autodidactes.* La fiabilité est la capacité de maintenir les fonctionnalités de réseau de capteurs sans la moindre interruption qui sera due à l'échec du noeud capteur. Ce dernier peut échouer en raison du manque d'énergie, de dommages physiques, de problèmes de communication, d'inactivité, ou d'interférence environnementale. Le réseau devrait pouvoir détecter l'échec d'un noeud et *s'organiser*, *se reconfigurer* et *récupérer* des échecs de noeud sans desserrer aucune information.

***4.2. Gestion et consommation d'énergie*:** Un des composants des noeuds capteurs est la source d'énergie qui peut être une batterie. Le noeud capteur sans fil, étant un dispositif microélectronique, peut seulement être équipé d'une source d'énergie limitée. Au-delà de l'endroit inaccessible à distance avec moins de contrôle et d'existence humaine, les sources d'énergie jouent un rôle critique dans la survie des noeuds capteurs. La source d'énergie devrait être intelligemment divisée au-delà du captage, du calcul, et des phases de communication selon le besoin. Les capteurs peuvent être hibernés lorsqu'ils sont inactifs. Un bon nombre de recherches courantes se concentrent sur la conception de protocoles et d'algorithmes power-aware (consommation d'énergie minime) pour les réseaux de capteurs sans fil. Récemment, l'énergie solaire est également considérée comme une option pour autoriser les noeuds capteurs à distance qui sont considérés comme un environnement exposé.

Le modèle de consommation d'énergie dans les réseaux de capteurs

Un capteur utilise son énergie pour réaliser trois actions principales : l'acquisition, la communication et le traitement des données.

- ***Acquisition*** : L'énergie consommée pour effectuer l'acquisition n'est pas très importante. Néanmoins, elle varie en fonction du phénomène et du type de surveillance effectué.

- *Communication* : Les communications consomment beaucoup plus d'énergie que les autres tâches. Elles couvrent les communications en émission et en réception. La figure 7 présente un modèle d'antenne et les règles de consommation d'énergie associées [3].

- Pour transmettre un message de ***k*** bits sur une distance de ***d*** mètres, l'émetteur consomme:

$$E_{Tx}(k,d) = E_{Tx}(k) + E_{Tx\_amp}(k,d) \quad (1)$$

$$E_{Tx}(k,d) = \begin{cases} k*E_{elec}(k,d) + k*\varepsilon_{amp}*d^2 & \text{si } d < d_{crossover} \\ k*E_{elec}(k,d) + k*\varepsilon_{amp}*d^4 & \text{sinon} \end{cases} \quad (2)$$

- Pour recevoir un message de ***k*** bits, le récepteur consomme :

$$E_{rx}(k) = k*E_{elec} \quad (3)$$

Avec :

E_{elec} : énergie de transmission/réception électronique ;

k : taille d'un message ;

d : distance entre l'émetteur et le récepteur ;

E_{Tx_amp} : énergie d'amplification;

ε_{amp} : facteur d'amplification;

$d_{crossover}$: distance limite pour laquelle les facteurs de transmission changent de valeur.

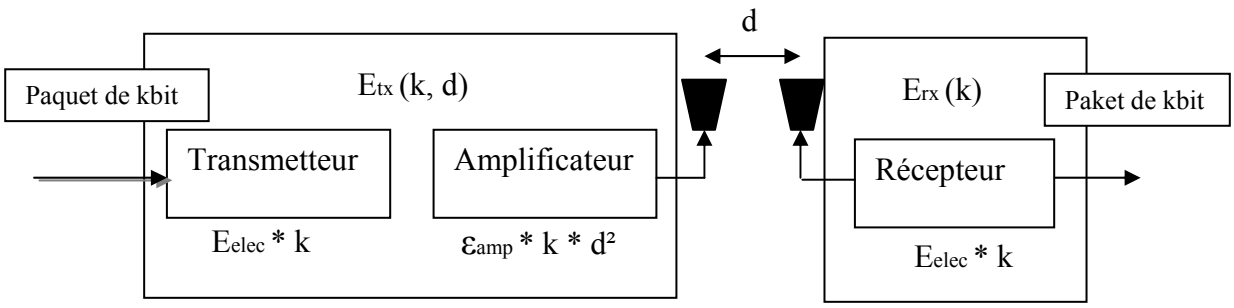


Figure 7 : *Modèle de consommation d'énergie pour la communication.*

- *Traitement des données :* L'énergie consommée pour les opérations de calcul est beaucoup plus faible que l'énergie de communication. L'énergie nécessaire pour transmettre *1 KB* sur une distance de *100m* est approximativement équivalente à l'énergie nécessaire pour exécuter

3 millions d'instructions avec une vitesse de *100 millions* d'instructions par seconde (MIPS). Ce niveau peut être dépassé en fonction des circuits installés dans les noeuds et des fonctionnalités requises. L'énergie consommée par le traitement des données est calculée en appliquant la formule suivante :

$$E_{DA} = 5nj/bit/signal \quad (4)$$

4.3. Efficacité du réseau et agrégation de données : Diffuser les données captées sur le réseau peut facilement encombrer ce dernier. Certaines applications critiques comme les détecteurs d'intrus exigent une transmission pressante et un traitement plus rapide des données qui peuvent dégrader l'exécution et perdre la fiabilité due à la congestion ou à la latence dans le réseau. L'agrégation intelligente des données captées et l'élimination de l'information non désirée et redondante ainsi que la compression des données peuvent être une solution pour l'utilisation efficace de ressource et d'énergie et l'évitement de congestion. Plusieurs algorithmes comme la diffusion dirigée [15, 29] sont proposés pour faciliter l'agrégation et la diffusion de données dans le contexte des WSNs.

4.4. Routage Intelligent : Dans plusieurs applications, les noeuds capteurs sont des noeuds mobiles et peuvent changer dynamiquement l'emplacement. Les protocoles de routage doivent être adaptatifs à ces changements et devraient être auto- curatifs et auto-configurant. L'information devrait être persistante malgré les changements des noeuds du réseau. La basse capacité de traitement d'un noeud crée intelligemment beaucoup de défis pour les paquets de routage dans tous les noeuds voisins. Comme discuté précédemment, certaines applications peuvent exiger une communication plus rapide et une réponse instantanée. Les algorithmes de routage devraient être intelligents pour choisir les sauts et les pats de distance miniums pour le transfert des données.

4.5. Le défi de gestion : contrôler la communication sur les réseaux hétérogènes représente un défi de base dans le système auto- contrôlant du fait que les politiques et les protocoles de communication projettent un rôle important dans la communication du réseau. En outre, il est nécessaire d'équilibrer le niveau du détail que le réseau fournit au client par rapport au taux auquel l'énergie est consommée tout en recueillant les données. Clairement, il est préférable d'avoir le réseau qui fait automatiquement cet accord, plutôt que d'exiger une intervention manuelle.

Ces basics besoins et objectifs de conception servent comme défi à la technologie courante. Bien que des protocoles courants de routage IP existent et aient des applications significatives dans les réseaux et Internet courants, ils ne répondent pas à des exigences complètes de conception dans les réseaux de capteurs sans fil parce que les noeuds WSN ont typiquement des capacités de calcul limitées et moins de puissance. Ainsi, WSN requiert une infrastructure et un ensemble de protocoles différents, qui peuvent être implémentés en utilisant le concept de calcul autonome.

5. Conclusion

Nous avons essayé à travers ce chapitre de mettre le point sur l'architecture des RCSFs, ainsi que leurs principaux domaines d'application et leur facteur et défi de conception. Cette mise au point nous a permis de déduire que les protocoles de routage jouent un rôle déterminant et crucial dans la conception des RCSFs. Cela nous a mené à faire une étude des principaux protocoles de routage proposés dans le chapitre qui suit.

1. Introduction

Bien qu'un grand nombre d'applications mettent en jeu des WSN, ceux-ci ont plusieurs restrictions que ces applications doivent contourner. Par exemple, ils ont une faible puissance de calcul, une réserve d'énergie limitée et une bande passante réduite aux connections sans fil entre capteurs. Un des principaux objectifs dans la conception de WSN est la transmission fiable de données via une heuristique de préservation d'énergie et de prévention de perte de connectivité (e.g. aucun noeud isolé). Ceci est fait par l'utilisation d'une politique stricte de gestion d'énergie. En effet, la principale source de consommation d'énergie d'un capteur est l'utilisation du réseau sans fil via son module de radiocommunication.

Les protocoles de routage au sein des WSN sont influencés par un facteur déterminant à savoir **Consommation d'énergie sans perte d'efficacité.** Les capteurs utilisent leur réserve d'énergie à des fins de calcul et de transmission de données. La durée de vie d'un capteur dépend essentiellement de celle de sa batterie. Dans un WSN, chaque noeud joue le rôle d'émetteur et de routeur. La défaillance énergétique d'un capteur peut changer significativement la topologie du réseau et imposer une réorganisation coûteuse de ce dernier.

La plupart des protocoles de communication dans les réseaux Ad-Hoc ne s'adaptent pas aux particularités des réseaux de capteurs, d'où la nécessité de les améliorer ou de développer de nouveaux protocoles. De nombreuses stratégies de routages ont été créées pour les réseaux de capteurs sans fil. Certaines sont des adaptations de stratégies qui existaient pour d'autres types de réseaux (principalement pour les réseaux sans fil au sens le plus large) tandis que d'autres ont été conçues spécialement pour les réseaux de capteurs sans fil.

2. Les algorithmes de routage dans RCSFs

Les algorithmes de routage sont en fait découpés en trois familles : les algorithmes de routage données centrales, hiérarchiques ou géographiques**.**

2.1. Les protocoles données centrales (Data- centric protocols):

Le routage *données centrales* [1, 2, 17, 19, 20, 25, 32], est le modèle le plus simple où chaque noeud dans le réseau transmet à la station de base. Chaque noeud joue typiquement le même rôle et les noeuds capteurs collaborent pour accomplir la tâche de capter. La station de base envoie des requêtes à certaines régions et se met en attente des données des capteurs situés dans les régions choisies.

On peut citer comme exemple :

2.1.1. Propagation et discussion (flooding and gossiping):

Dans la propagation, chaque capteur recevant un paquet de données le renvoi à tous ses voisins. Ce processus continue jusqu'à ce que le paquet arrive à destination ou jusqu'à ce que le nombre maximum de sauts pour le paquet soit atteint. D'autre part, le gossiping est une version légèrement améliorée du flooding où le nœud récepteur envoi le paquet à un voisin choisi aléatoirement.

2.1.2. Protocoles de capteur pour l'information par négociation (SPIN):

L'idée derrière le SPIN est de nommer les données en utilisant des descripteurs de haut niveau ou des méta données. Avant la transmission, les méta- données sont échangées entre les capteurs par un mécanisme de publicité de données. Chaque nœud recevant de nouvelles données, l'annonce à ses voisins et ceux intéressés récupèrent les données en envoyant une requête (figure 8).

2.1.3. Routage par rumeur:

L'idée est de transmettre les requêtes aux nœuds qui ont observé un évènement particulier. Quand un nœud détecte un événement, il l'ajoute à sa table locale et génère un agent. L'agent parcoure le réseau afin de propager des informations sur les événements locaux aux nœuds distants. Quand un nœud génère une requête pour un événement, les nœuds qui connaissent l'itinéraire peuvent répondre en référant la table d'événements.

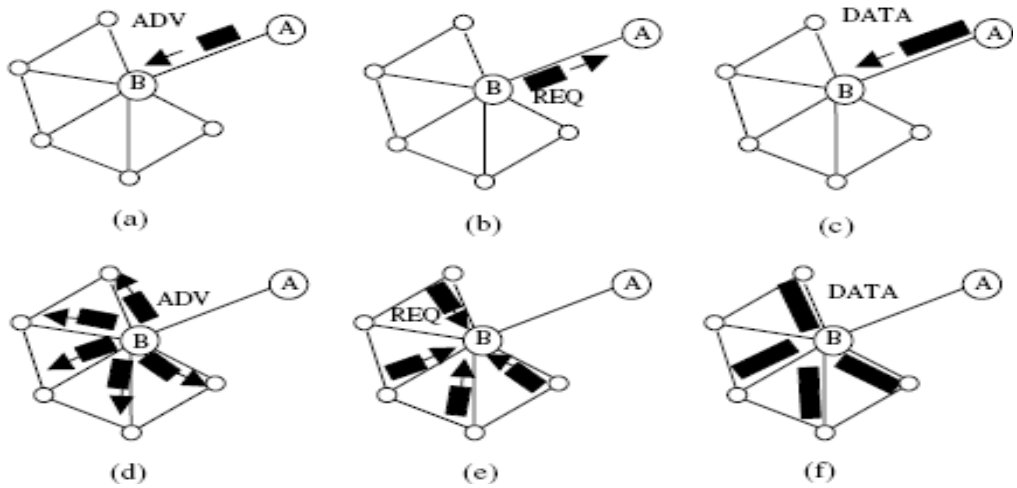


Figure 8 : *le protocole SPIN. Le nœud A annonce ses données au nœud B (a). B répond par une requête (b). B reçoit les données requises (c). B fait de la publicité à ses voisins (d) qui répondent par des requêtes (e-f)*

2.1.4. Diffusion dirigée:

Une requête intérêt est définie par une liste de couples attribut-valeur et est diffusée par le sink vers ses voisins. L'intérêt contient aussi plusieurs champs gradient. Un gradient est un lien de réponse de la part du voisin recevant l'intérêt. En utilisant les intérêts et les gradients [15, 29], plusieurs chemins peuvent être établis entre le sink et la source. L'un de ces chemins est sélectionné par renforcement (figure 9). Si ce chemin échoue, un nouveau ou un alternatif doit être identifié, puisque les données sont demandées par des requêtes. En outre, cette méthode est coûteuse en terme de consommation d'énergie et ne représente pas un bon modèle pour WSN.

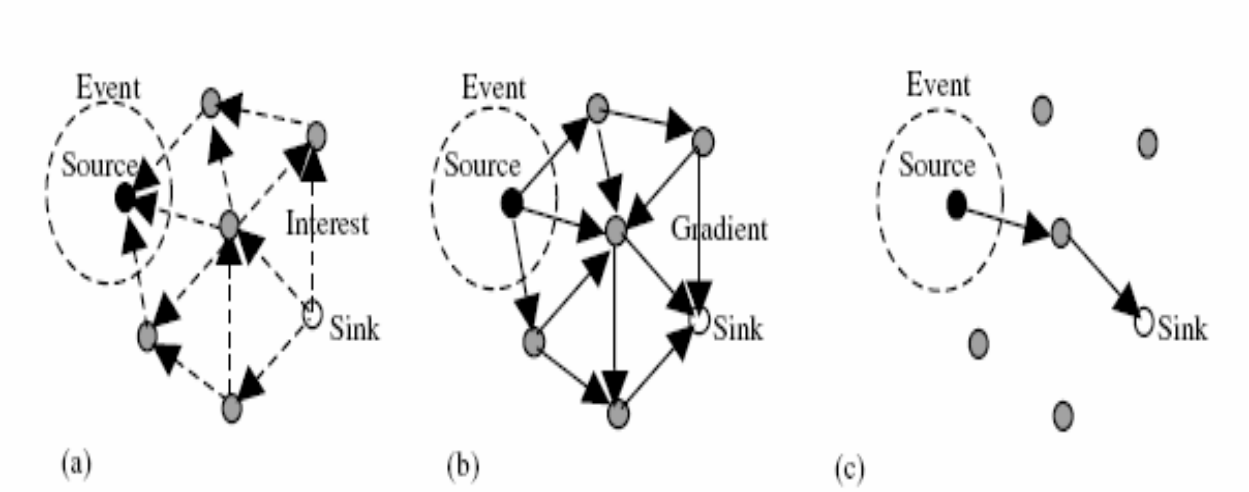


Figure 9 : *phases du protocole de diffusion dirigée. (a) propagation de l'intérêt. (b) construction des gradients initiaux (c) livraison des données par renforcement.*

2.2. Les protocoles basés sur la localisation (géographique) :

Dans ce type de routage [1, 2, 17, 19, 20, 25, 32], les noeuds capteurs sont adressés en fonction de leurs localisations. La distance entre les noeuds voisins peut être estimée sur la base des forces entrantes de signal. Des coordonnées relatives des noeuds voisins peuvent être obtenues en échangeant une telle information entre les voisins. Alternativement, la location des noeuds peut être disponible directement en communiquant avec un satellite en utilisant GPS (système de positionnement global).

Dans la plupart des protocoles de routage, l'information sur la localisation des nœuds est nécessaire afin de calculer la distance entre deux nœuds particuliers de sorte que la consommation d'énergie puisse être estimée. On peut mentionner parmi ces protocoles :

2.2.1. MECN et SMECN :

MECN (Minimum energy communication network) utilise le GPS à basse puissance. L'idée principale est de trouver un sous réseau, qui aura moins de nœuds et qui exige moins de puissance pour la transmission entre deux nœuds particuliers quelconques. SMECN (Small MECN) est une extension de MECN. Le sous- réseau construit par SMECN est probablement plus petit (en terme de nombre d'arcs) que celui construit par MECN.

2.2.2. GAF :

GAF (Geographic adaptive fidelity) conserve l'énergie par la mise en veille des nœuds inutiles dans le réseau sans affecter le niveau de fidélité du routage. Il forme une grille virtuelle pour le domaine couvert. Chaque nœud emploie sa position indiquée par le GPS pour s'associer à un point dans la grille virtuelle. Des nœuds liés au même point sur la grille sont considérés équivalents en terme de coût de routage. Une telle équivalence est exploitée en maintenant quelques nœuds, situés dans un secteur particulier de la grille, dans l'état de sommeil afin d'économiser de l'énergie.

2.2.3. GEAR (geographic and energy-aware routing):

L'idée est de restreindre le nombre d'intérêts dans la diffusion dirigée en considérant seulement certaines régions plutôt que d'envoyer les intérêts au réseau tout entier.

Le routage géographique suppose que tous les nœuds connaissent leur position. Une solution basée sur le GPS peut être trop coûteuse, d'autant plus que le nombre de nœuds à équiper est très grand.

2.3. Les protocoles hiérarchiques :

Le routage hiérarchique [[1, 2, 9, 16] est considéré comme étant l'approche la plus favorable en terme d'efficacité énergétique. Il se base sur le concept (nœud standard – nœud maître) où les nœuds standard acheminent leurs messages à leur maître, lequel les achemine ensuite dans le réseau tout entier via d'autres nœuds maîtres jusqu'à la station de base (sink).

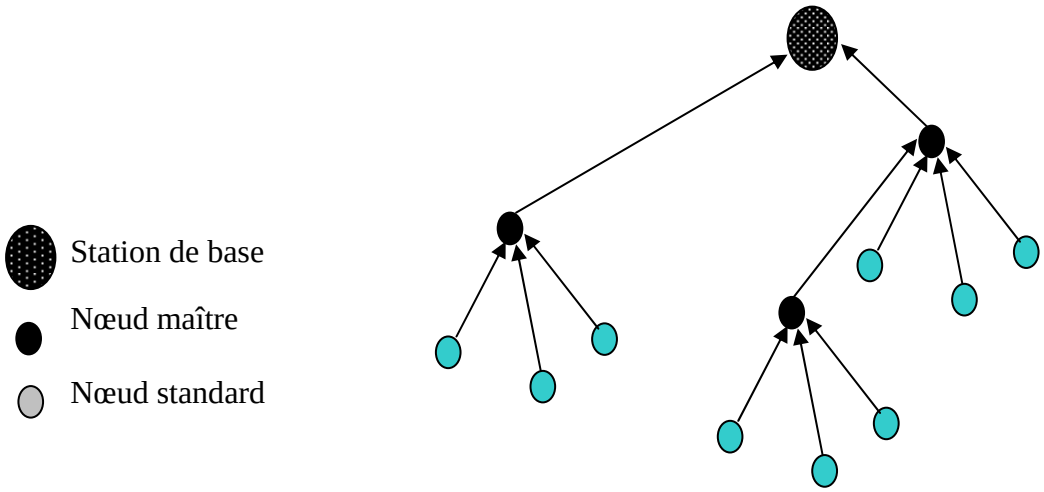


***Figure10** : Le routage hiérarchique*

Le point fort de ce type de protocoles est **l'agrégation et la fusion des données** afin de diminuer le nombre de messages transmis au sink, ce qui implique une meilleure économie d'énergie. En fait, deux grandes approches sont dérivées de ce type de protocoles à savoir ***:*** ***chaîne-based*** approach [7, 8, 30] (approche chaînée) et ***cluster-based*** approach [5, 13, 14, 31] (approche à grappe).

2.3.1. Cluster-based approach (approche à grappe)

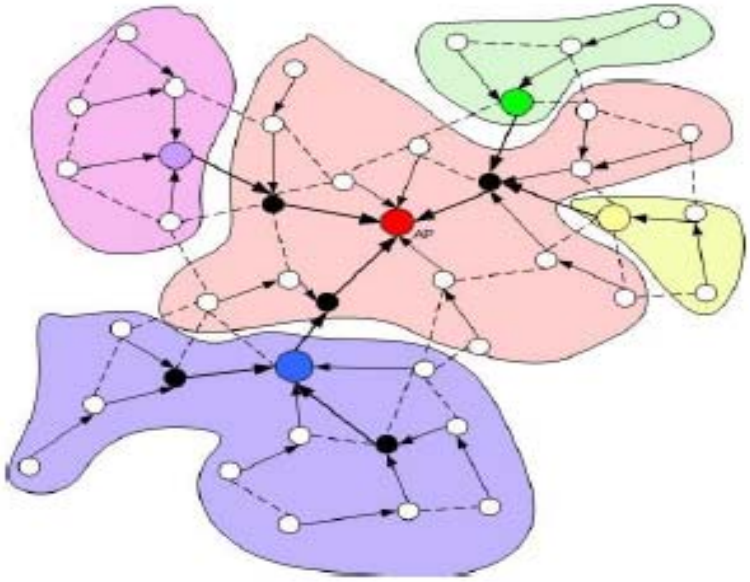


***Figure 11** : Cluster-based approach (approche à grappe)*

Elle consiste, de façon similaire aux réseaux téléphoniques cellulaires, à partitionner le réseau en groupes (clusters) où dans chacun d'entre eux, un seul capteur est sélectionné comme leader (Cluster Head) pour jouer le rôle spécial de point de transfert. D'ailleurs, chaque CH créé un plan de transmission pour les capteurs dans la grappe, ce qui permet aux antennes radio de chaque nœud non-CH d'être éteintes toutes les fois, excepté pendant son temps de transmission. Les noeuds dont l'énergie est la plus élevée peuvent être employés dans le traitement et l'envoie d'informations tandis que ceux de basse énergie peuvent être employés dans la collection de données.

L'agrégation des capteurs en grappes permet de réduire la complexité des algorithmes de routage, d'optimiser la ressource médium en la faisant gérer localement par un chef de grappe, de faciliter l'agrégation des données, de simplifier la gestion du réseau, en particulier l'affectation d'adresses, d'optimiser les dépenses d'énergie, et enfin de rendre le réseau plus évolutif (scalable). Le clustering permet aux noeuds d'effectuer des communications sur de petites distances avec leurs CHs.

La rotation des CHs s'avère également un facteur important pour l'organisation des réseaux de capteurs. Puisque la BS (station de base ou sink) est généralement loin du champ des capteurs, les CHs diffusent une quantité plus importante d'énergie pour la transmission de données à la BS. Par conséquent, les CHs mourront rapidement si le même noeud fonctionne continuellement comme un CH. Ainsi, pour ne pas puiser la puissance de batterie d'un capteur simple, les algorithmes de groupement (clustering) étudiés jusqu'ici présentent la rotation de CHs parmi les capteurs.

2.3.1.1 LEACH (Low Energy Adaptive Clustering Hierarchy)

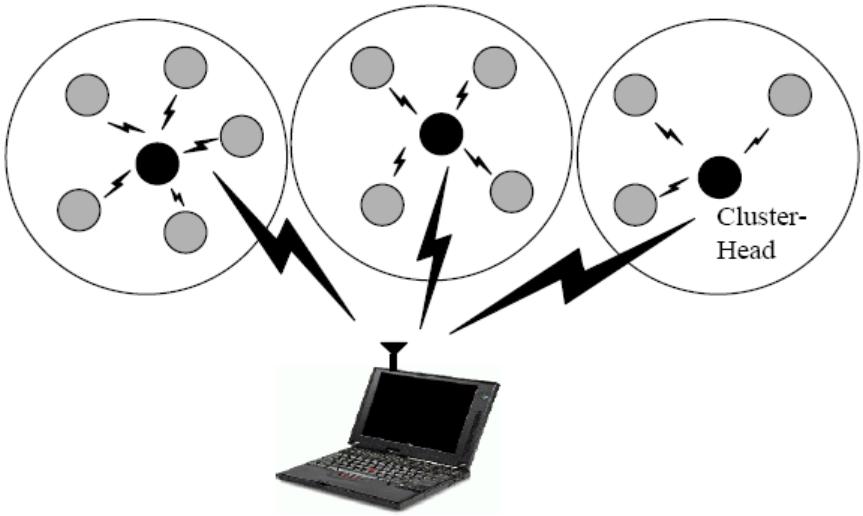


***Figure 12**: Algorithme de routage LEACH*

LEACH (*Low Energy Adaptive Clustering Hierarch*) [3, 4] est un protocole auto-organisateur basé sur le clustering adaptatif, qui utilise la rotation randomisée des têtes de grappe pour distribuer équitablement la charge d'énergie entre les noeuds capteurs dans le réseau. Il est considéré comme étant l'une des premières approches de routage hiérarchique basées sur le clustering. LEACH est fondé sur deux hypothèses de base :

- la station de base est fixe et est placée loin des capteurs,
- Tous les noeuds du réseau sont homogènes et limités en énergie.

L'idée derrière LEACH est de former des clusters des noeuds capteurs selon la force reçue du signal et d'utiliser les têtes locales de grappe (cluster head) comme des routeurs pour conduire des données à la station de base. Les dispositifs principaux de LEACH sont :

- *Coordination et contrôle localisés* pour l'initialisation et le traitement de grappe.
- *Rotation randomisée de la grappe* (effectuée par "les stations de base" ou "les têtes de cluster").
- Compression locale *(agrégation)* Les nœuds CH compressent les données arrivant des nœuds appartenants à leurs grappes respectives, et envoient un paquet d'agrégation à la station de base afin de réduire la quantité d'information qui doit être transmise à la station de base.

Dans LEACH, le traitement est séparé dans des cycles de longueur constante, où chaque cycle commence par une phase d'initialisation suivie d'une phase de transmission. La durée d'un cycle est déterminée. Dans la première phase, les grappes sont organisées et les CHs sont sélectionnés. Cette élection est basée sur le pourcentage désiré de CHs et le nombre d'itérations au cours duquel un noeud a pris le rôle de CH. Ainsi, un noeud **n** prend une valeur aléatoire entre **0** et **1**. Si cette valeur est inférieure au seuil **T (n)**, le noeud se déclare CH.

$$T(n)=\begin{cases}\dfrac{P}{1-P\times\left(r \bmod \dfrac{1}{P}\right)} & \text{si } (n\in G)\\ 0 & \text{sinon}\end{cases} \qquad (5)$$

Avec :

- ***P*** : pourcentage désiré de CHs.
- ***r*** : itération actuelle.
- ***G*** : ensemble des noeuds qui ont été sélectionnés comme CH durant les dernières (1/ *P*) itérations.

Chaque CH élu émet un message de signalisation au reste des nœuds dans le réseau, et qui présentent les nouveaux leaders. Tous les nœuds non leaders, et après avoir reçu ce message, décident de la grappe à laquelle ils veulent appartenir. Cette décision est basée sur la force du

signal du message. Les nœuds non leader informent les leaders appropriés qu'ils seront un membre de la grappe.

Après la réception de tous les messages des nœuds qui voudraient être inclus dans la grappe et être basés sur le nombre de nœuds dans la grappe, le nœud leader créé un programme TDMA et assigne à chaque nœud un slot de temps quand il peut transmettre. Ce programme est émit à tous les nœuds dans la grappe.

L'algorithme LEACH utilise la technique de multiplexage temporel TDMA (Time division multiplexed access) comme méthode d'accès au médium. Chaque nœud utilise la totalité de la bande passante allouée par le système de transmission durant son slot de temps. En fait, chaque CH agit comme un centre de commande local pour coordonner les transmissions des données dans sa grappe. Il crée un modèle de communication en se basant sur TDMA, ensuite il le transmet aux nœuds de sa grappe (cluster). Etant donné que chaque nœud connaît d'avance le slot de temps qu'il va occuper, cela permet alors au nœud de passer à l'état "endormi" durant les slots inactifs. Ainsi, la perte d'énergie due aux états de sur écoute (overhearing) et d'écoute passive (idle) est évitée.

Chaque CH choisit aléatoirement un code dans une liste de codes de propagation CDMA, il le transmet aux noeuds appartenant à son cluster afin de l'utiliser pour leurs transmissions, ceci afin de minimiser les interférences entre les messages des CHs les plus proches.

Dans la deuxième phase, le transfert des données actuelles à la station de base a lieu. La durée de la deuxième phase est plus longue que celle de la première phase afin de réduire au minimum les problèmes d'oveheading. Cependant, la collection de données est centralisée et est exécutée périodiquement. Par conséquent, ce protocole s'avère le plus approprié quand on constate un besoin de surveillance de constante par le réseau de capteurs.

Après un intervalle de temps donné, une rotation randomisée du rôle du CH est conduite de sorte que la dissipation uniforme d'énergie dans le réseau de capteurs soit obtenue. Les auteurs ont trouvé, en se basant sur leur modèle de simulation, que seulement 5% des nœuds ont besoin d'agir comme leaders.

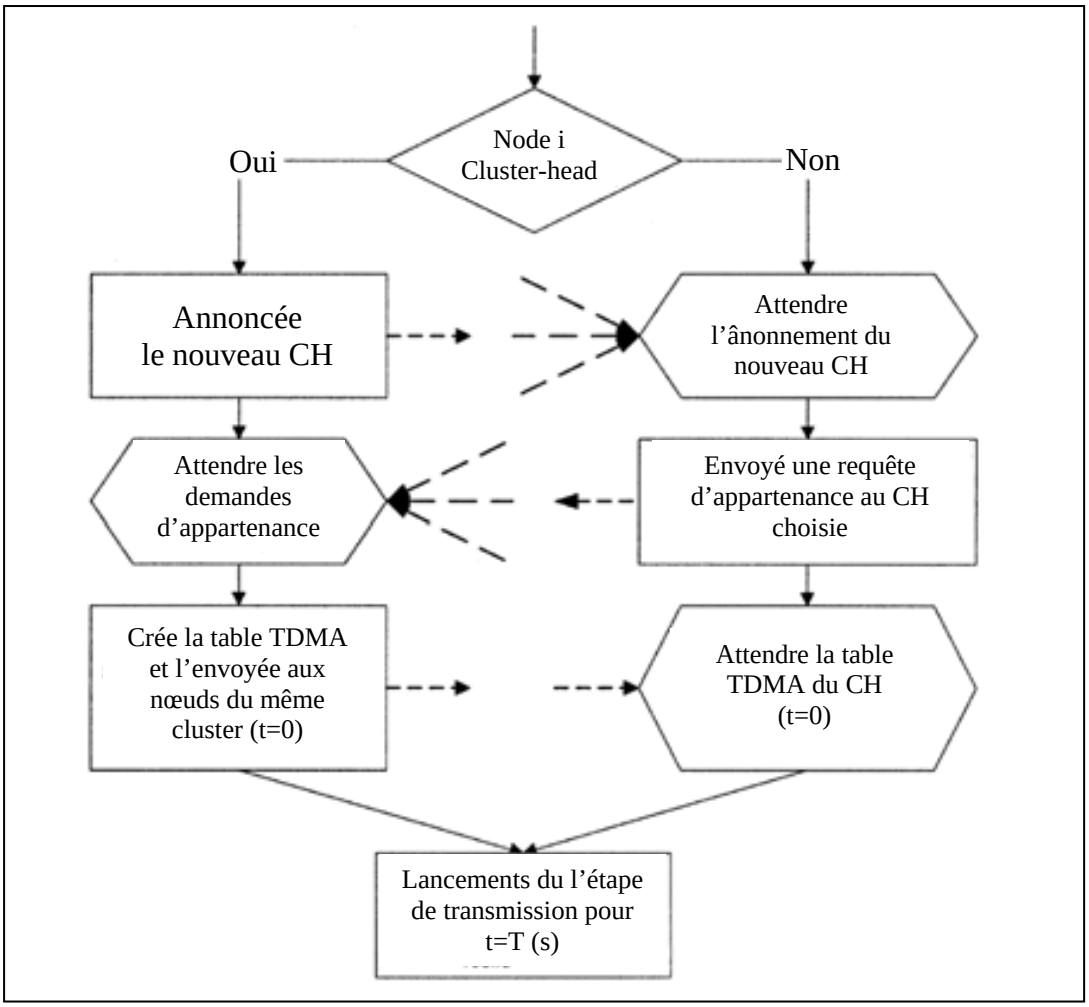


Figure 13 *: opérations de l'étape d'initialisation*

Etant donné que cet algorithme ne garantit pas le nombre de CHs prévu à l'initialisation de l'algorithme ni la distribution équitable des CHs, une version centralisée de l'algorithme (LEACH-C) [3] est donc proposée. Cette dernière permet de déterminer, à partir de la position exacte des noeuds, la configuration optimale pour minimiser l'énergie dépensée. LEACH-C (LEACH-CENTRALISE) est une variante de LEACH où les grappes sont formées d'une manière centralisée par la station de base. LEACH-C utilise la même étape de transmission que LEACH. Durant la phase d'initialisation la BS reçoit de chaque nœud des informations concernant leur localisation, et leur réserve d'énergie. Ensuite, elle exécute un algorithme de formation de grappes centralisé afin de former les grappes et sélectionner leurs CHs. LEACH-C utilise l'algorithme de la réussite –simulée [11] pour obtenir des grappes optimales. Dès que les grappes sont formées, la station de base envoie ces informations à tous les nœuds du

réseau. Cependant, la version centralisée de LEACH n'est pas adaptée aux réseaux de grande dimension.

Un développement ultérieur est LEACH-F [3] (LEACH avec grappes fixes). LEACH-F est basé sur les grappes qui sont formés une fois - et puis sont fixées. Ensuite, la position de tête de grappes tourne parmi les noeuds dans la grappe. L'avantage est que, une fois que les grappes sont formées, aucune autre phase d'initialisation aura lieu, LEACH-F utilise le même algorithme centralisé de formation de grappe que LEACH-C. Les grappes fixes dans LEACH-F ne permettent pas à de nouveaux noeuds d'être ajoutés au système et n'ajustent pas leur comportement basé sur la mort de noeuds.

Malgré le fait que LEACH réalise au-dessus d'un facteur de réduction de 7 dans la dissipation d'énergie, comparé à la communication directe, et un facteur de 4-8, comparé au protocole de routage de transmission minime d'énergie, ainsi que la réduction des messages de contrôle, la réutilisation de largeur de bande, l'amélioration du contrôle de puissance, le non gaspillage d'énergie, un nombre d'inconvénients restent plus au moins apparents. Nous citons parmi eux :

- L'idée de formation de grappes augmente le nombre de messages échangés entre les nœuds, qui peuvent diminuer le gain dans la consommation d'énergie ;
- Les nœuds les plus éloignés du CH meurent rapidement par rapport à ceux plus proches ;
- Le nombre de messages reçus par les CHs équivaut, approximativement, celui des nœuds gérés. Ceci mène à l'épuisement rapide de leur réserve d'énergie ;
- L'utilisation de singel-hop (un saut) au lieu de multi-hop (multi sauts) épuise rapidement l'énergie des nœuds et consomme d'avantage de largeur de bande.
- LEACH suppose que tous les nœuds peuvent transmettre avec une énergie assez suffisante pour atteindre la BS si nécessaire. Ce qui n'est pas toujours le cas.
- Station de base stationnaire peut s'avérer peu judicieuse.
- Il ne peut pas être appliqué à des applications avec une contrainte de temps du fait qu'il résulte en une longue latence.

Une deuxième approche de routage hiérarchique (Chain-based approach) a été proposée, afin de réduire les inconvénients de la première (cluster-based approach).

2.3.2. Chain-based approach (approche à chaîne)

Dans cette approche [30], le principe de clustering est abandonné, les nœuds du réseau sont organisés de façon à former une grande chaîne de proches voisins, où un seul nœud est sélectionné pour transmettre au sink. En fait, l'idée de formation de chaîne a été proposée pour la première fois dans l'algorithme PEGASIS [7, 8] (Power-Efficient Gathering in Sensor Information Systems).

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

Un perfectionnement meilleur que celui du protocole LEACH a été proposé. Le protocole PEGASIS (Power-Efficient Gathering in Sensor Information Systems) est un protocole basé chaîne, proche de l'optimal. L'idée de base du protocole est que, dans le but de prolonger la durée de vie du réseau, les nœuds vont être organisés de telle sorte à ce qu'ils forment une chaîne, n'auront ainsi besoin de communiquer qu'avec seulement leurs voisins les plus proches et se relaient dans la communication avec la station de base. Quand le cycle de tous les noeuds communiquant avec la station de base aura finit, un nouveau cycle commence et ainsi de suite. Ceci réduit l'énergie exigée pour transmettre des données par cycle du moment que la dissipation d'énergie est diffusée uniformément au-dessus de tous les noeuds. Par conséquent, PEGASIS a deux principaux objectifs. D'abord, augmenter la durée de vie de chaque noeud en employant des techniques de collaboration et augmenter par conséquent la durée de vie du réseau. En second lieu, permettre seulement la coordination locale entre les noeuds voisins de sorte que la largeur de bande consommée dans la communication soit réduite. A la différence de LEACH, PEGASIS évite la formation de grappes et n'utilise qu'un seul nœud dans une chaîne afin de transmettre à la BS, au lieu d'en utiliser plusieurs.

Dans PEGASIS, pour localiser le voisin le plus proche, chaque nœud utilise la force du signal pour mesurer la distance vers tous les nœuds voisins, et ajuster par la suite la force du signal de telle sorte que seul un nœud peut être entendu. La forme agrée des données sera envoyée à la station de base par n'importe quel nœud dans la chaîne et les nœuds dans cette dernière vont se prendre en relais dans la transmission à la station de base.

Approche de construction de chaîne

Les nœuds vont être organisés de sorte qu'ils forment une chaîne, qui peut être soit calculée d'une façon centralisée par la BS et émise à tous les nœuds, ou accomplie par les noeuds capteurs eux-mêmes en employant un algorithme avide (greedy algorithme). Si la chaîne est calculée par les noeuds capteurs, ils peuvent d'abord obtenir toutes les données sur l'emplacement des noeuds capteurs et calculent localement la chaîne en utilisant le même algorithme avide. Puisque tous les noeuds ont les mêmes données d'emplacement et exécutent le même algorithme, ils vont tous produire le même résultat.

Pour construire la chaîne, PEGASIS commence avec le nœud le plus éloigné de la BS (choisir un noeud aléatoirement s'il y a une cravate). Le voisin le plus proche de ce nœud sera le nœud suivant dans la chaîne. Les voisins successifs sont sélectionnés de cette manière parmi les noeuds non visités (avec une cravate brisée arbitrairement) afin de former la chaîne de noeuds.

L'algorithme commence par le noeud le plus lointain pour s'assurer que les nœuds les plus loin de la BS ont des voisins proches à mesure que, dans l'algorithme avide, les distances voisines augmenteront graduellement puisque des noeuds déjà présents sur la chaîne ne peuvent pas être revisités. La figure 14 montre le noeud c0 se reliant au noeud c1, le noeud c1 se reliant au noeud c2, et le noeud c2 se reliant au noeud c3, dans cet ordre. Quand un nœud meure, la chaîne est reconstruite de la même manière pour dévier le noeud mort.

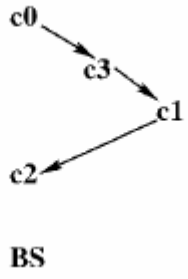


Figure 14 : *Construction de chaînes en utilisant l'algorithme avide.*

Pour collecter des données des nœuds capteurs dans chaque cycle, chaque nœud reçoit les données d'un voisin, les fusionne avec les siennes, et transmet à un autre voisin dans la chaîne. Il faut noter que ce noeud i serait à une certaine position aléatoire j sur la chaîne. Les

nœuds se relient dans la transmission à la BS et PEGASIS va utiliser le mod N du numéro de nœud i (N représente le nombre de noeuds) afin de transmettre à la BS dans le cycle i. Ainsi, le leader dans chaque cycle de communication sera à une position aléatoire sur la chaîne.

Chaque cycle de collecte de données peut être lancé par la BS avec un signal de balise qui synchronisera tous les noeuds capteurs. Puisque tous les noeuds connaissent leurs positions sur la chaîne, PEGASIS peut employer une approche de slot de temps (TDMA) pour la transmission des données. Dans le **i**^{ème} cycle de collecte de données, le nœud (i-1) sera leader.

Le noeud c_0 de fin transmettra ses données au noeud c_1 dans le premier slot, c_1 fusionne et transmet les données dans le deuxième slot, et ainsi de suite jusqu'à ce que le nœud leader soit atteint. Dans les slots suivants, les transmissions de données ont lieu depuis le nœud $c_{(n-1)}$ et se déplacent vers le nœud leader de la bonne extrémité de la chaîne. Finalement, dans le n^{ième} slot, le leader transmet les données à la BS.

Alternativement, dans un cycle donné, PEGASIS peut utiliser une approche de déplacement de jeton à contrôle simple lancée par le leader pour commencer la transmission des données des extrémités de la chaîne. Le coût est très petit du fait que la taille du jeton est très petite. Dans la figure 15 suivante, le nœud c_2 est le leader et va passer le jeton le long de la chaîne commençant au noeud c_0. Le nœud c_0 passera ses données au noeud c_2. Après que le nœud c_2 ait reçu les données du nœud c_1, il passera le jeton au nœud c_4, et le nœud c_4 passera ses données au nœud c_2 avec fusion des données le long de la chaîne.

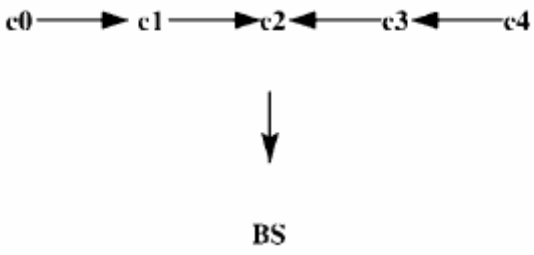


Figure 15 : Approche de dépassement de jeton.

PEGASIS exécute la fusion de données à chaque noeud excepté les noeuds de fin de chaîne. Chaque nœud va fusionner les données de ses voisins avec les siennes afin de générer un paquet simple et les transmet par la suite à son autre voisin (Si il en a deux). Dans l'exemple précédent, le nœud c_0 va transmettre ses données au nœud c_1. c_1 fusionne les données de c_0

avec les siennes et transmet ensuite au leader. Après que c_2 ait passé le jeton à c_4, le nœud c_4 transmet ses données à c_3. Le nœud c_3 fusionne ses données avec celles de c_4 et les transmet ensuite au leader. Le nœud c_2 attend de recevoir les données de ses deux voisins et fusionne alors ses données avec les leurs. Finalement, le nœud c_2 transmet un message à la BS. Ainsi, dans PEGASIS, chaque nœud va recevoir et transmettre un paquet de données dans chaque cycle et sera le leader une fois chaque N cycles. En addition, les nœuds reçoivent et transmettent des paquets de contrôle de jeton très petits.

Dans la formation de la chaîne, il est possible que certains nœuds puissent relativement avoir des voisins distants le long de cette dernière. De tels nœuds vont dissiper plus d'énergie dans chaque cycle, comparé à d'autre capteurs. L'auteur a amélioré la performance de PEGASIS en ne permettant pas à de tels nœuds de devenir leaders. L'auteur a accompli ceci en plaçant un seuil sur la distance voisine pour être leaders. Il peut être capable d'améliorer légèrement la performance de PEGASIS en appliquant un seuil adaptatif au niveau d'énergie restante dans les noeuds. Chaque fois qu'un noeud meurt, la chaîne sera reconstruite et le seuil peut être changé afin de déterminer quel nœud peut être leader.

Le protocole PEGASIS s'améliore par rapport à LEACH par l'économie énergie dans plusieurs étapes. En premier lieu, dans la collecte locale, les distances que la plupart des nœuds transmettent sont beaucoup moins par rapport à la transmission au leader dans LEACH. En second lieu, la quantité de données à recevoir par le leader est deux messages au plus au lieu de 20 (20 nœuds par grappe dans LEACH pour un réseau de 100 noeuds). Finalement, seul un nœud transmet à la BS dans chaque cycle de communication.

2.3.3 Comparaison des protocoles (LEACH et PEGASIS) à l'aide de la simulation

Les résultats de simulation détaillée dans [8] montrent que PEGASIS est capable de prolonger la durée de vie du réseau deux fois autant que celle du réseau sous le protocole LEACH. Un tel gain dans la performance est réalisé à travers l'élimination d'overhead causé par la formation de grappes dynamiques dans LEACH, et à travers aussi la diminution du nombre de transmissions et de réceptions en employant l'agrégation de données.

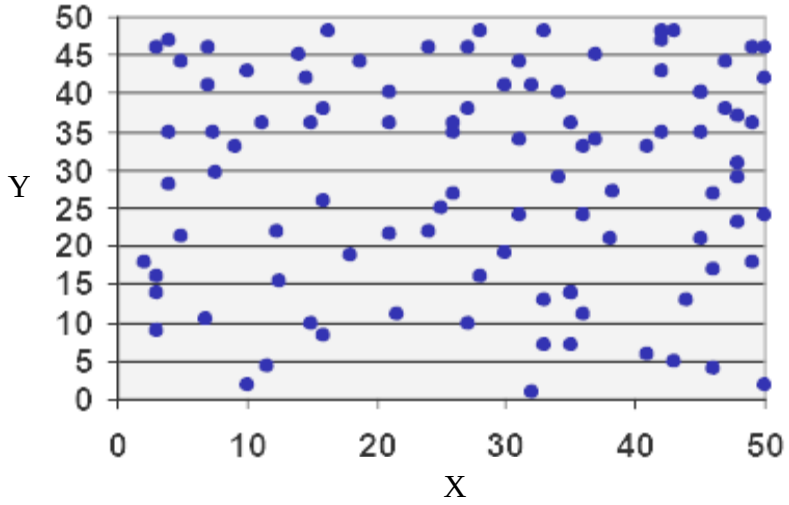


Figure 16 : *Topologie de 100 noeuds aléatoires pour un réseau 50m x 50m*
BS est localisée à (25, 150), qui est au moins de 100m

La simulation utilise plusieurs réseaux aléatoires de 100 nœuds. La figure précédente montre un réseau aléatoire de 100 nœuds. La BS est localisée à (25, 150) dans un champ de 50m x 50m, et localisée à (50, 300) dans un champ de 100m x 100m. Les simulations ont été exécutées afin de déterminer le nombre de cycles de communication quand 1%, 20%, 50% et 100% des nœuds meurent en utilisant la transmission directe. Une fois que le nœud meure, il restera considéré tel pour le reste de la simulation.

La simulation montre que PEGASIS accompli :

- Approximativement 2x le nombre de cycles, comparé à LEACH quand 1%, 20%, 50%, et 100% des noeuds meurent pour un réseau de 50m x 50m.
- Approximativement 3x le nombre de cycles, comparé à LEACH quand 1%, 20%, 50%, et 100% des noeuds meurent pour un réseau de 100m x 100m.
- Dissipation équilibrée d'énergie parmi les noeuds capteurs pour avoir l'utilisation complète de tout le réseau de capteurs.
- Une performance proche de l'optimal.

Le tableau suivant récapitule les résultats avec une énergie initiale par nœud de 0.25J, .5J, et 1.0J pour les réseaux de 50m x 50m et 100m x 100m. La position ombrée du tableau est pour le réseau de 50m x50m.

Les nœuds commencent à mourir à un taux plus uniforme après environ la mort de 20% des nœuds. Ceci est dû au fait que les distances entre les nœuds deviennent plus grandes, et les nœuds ont à devenir leaders plus souvent causant ainsi la diminution plus rapide d'énergie. Comme prévu, le nombre de cycles double autant que celui de l'énergie/nœud pour une taille, de réseau donnée.

Energie J/nœud	Protocoles	1%	20%	50%	100%
	Direct	54	62	76	117
.25	LEACH	402	480	523	635
	PEGASIS	788	1004	1041	1096
	Direct	108	124	152	235
.5	LEACH	803	962	1036	1208
	PEGASIS	1578	2011	2082	2192
	Direct	215	248	304	471
1.0	LEACH	1610	1921	2055	2351
	PEGASIS	3159	4023	4165	4379
	Direct	14	16	20	30
.25	LEACH	166	204	232	308
	PEGASIS	335	624	684	779
	Direct	28	32	40	61
.5	LEACH	339	408	461	576
	PEGASIS	675	1250	1362	1544
	Direct	56	64	80	122
1.0	LEACH	690	812	911	1077
	PEGASIS	1346	2497	2720	3076

Tableau 1. *Nombre de tours quand 1%, 20%, 50%, et 100% nœuds meurent. La portion ombragée représente un réseau de 50m x 50m, et la portion non ombragée représente un réseau de 100m x 100m.*

La figure suivante (figure 17) montre le nombre de cycles jusqu'à ce que 1%, 20%, 50%, 100% de nœuds meurent pour un réseau de 50m x 50m et la figure qui la suit (figure 18) montre les mêmes paramètres mais pour un réseau de 100m x 100m. PEGASIS est approximativement 2x meilleur que LEACH dans tous les cas pour un réseau de 50m x 50m. La valeur énergétique initiale pour les nœuds est 0.25J dans la première figure et 0.50J dans la deuxième. Comme le niveau d'énergie double, le nombre de cycles doubles aussi pour tous les cas. Pour un réseau de 100m x 100m, PEGASIS exécute 3x mieux que LEACH.

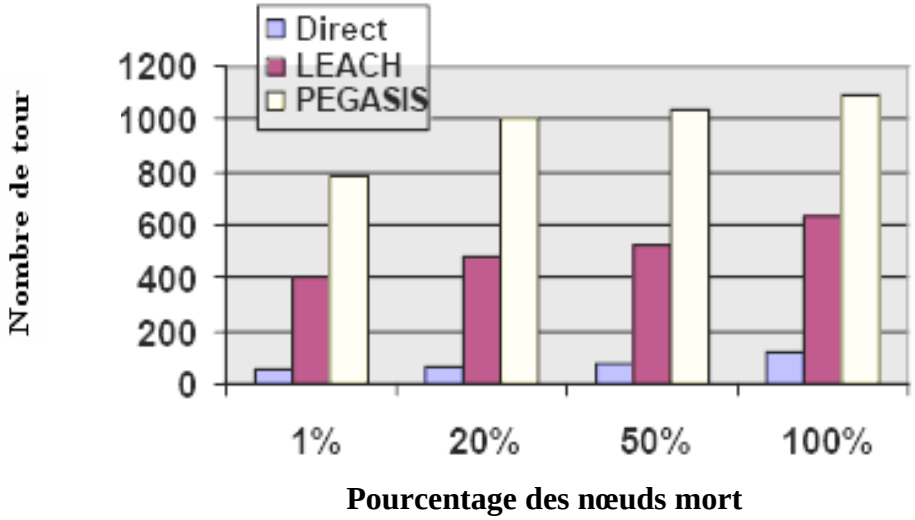


Figure 17 : Les résultats de performance pour un réseau de 50m x 50m avec une énergie initiale de .25J/nœud

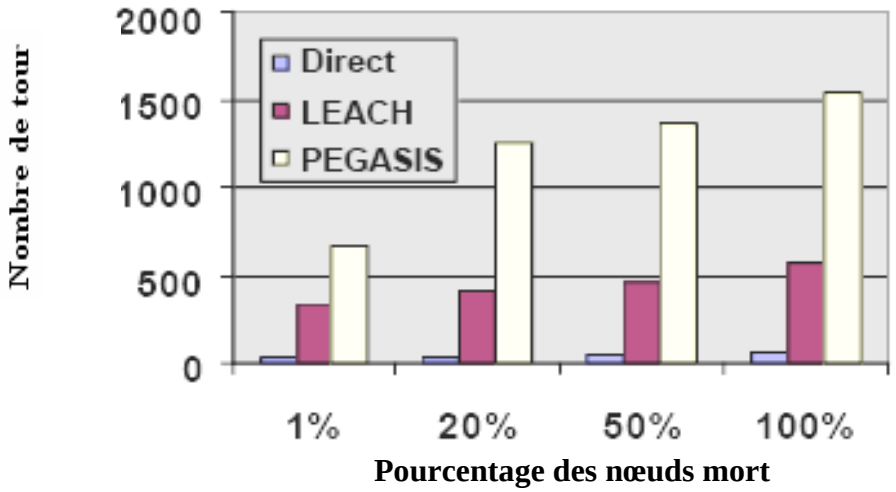


Figure 18 : Les résultats de performance pour un réseau de 100m x 100m avec une énergie initiale de .5J/nœud

Bien que le 'clustering' soit évité PEGASIS (l'approche de chaîne) présente un **retard excessif** pour le noeud le plus éloigné sur la chaîne. PEGASIS suppose que chaque noeud capteur doit pouvoir communiquer directement avec la BS. Dans des cas pratiques, les nœuds capteurs utilisent la communication multi sauts pour atteindre la station de base. En outre,

PEGASIS suppose que tous les noeuds maintiennent une base de données complète à propos de la location de tous les autres nœuds dans le réseau. La méthode par laquelle les nœuds sont localisés n'est pas décrite. En outre, PEGASIS suppose que les nœuds capteurs ont tous le même niveau d'énergie et q'ils sont susceptibles de mourir en même temps. Ceci ne peut pas être appliqué à un réseau de capteurs dans lequel il n'est pas facile d'obtenir la connaissance globale du réseau (position de tous les noeuds).

Une amélioration de PEGASIS, appelée PEGASIS hiérarchique [8] a été présentée, qui vise à diminuer le retard encouru pour les paquets pendant la transmission à la station de base à travers les transmissions simultanées des messages de données.

Pour éviter les collisions et les interférences possibles du signal le long des capteurs, deux approches ont été investiguées. La première approche incorpore le codage du signal, exemple CDMA. Dans la deuxième approche, seuls les nœuds spatialement séparés ont la permission de transmettre en même temps.

Le protocole basé chaîne et utilisant une approche de transmission CDMA, construit une chaîne de nœuds, qui forme un arbre hiérarchique, et chaque nœud sélectionné dans un niveau particulier transmet les données au nœud du niveau supérieur de la hiérarchie. Cette méthode permet d'assurer la transmission des données en parallèle et réduit significativement le retard.

Puisque l'arbre est équilibré, le retard sera dans O(lg N) où N est le nombre de nœuds. Par exemple, dans la figure qui suit, le nœud c_3 est le leader désigné pour le $3^{ème}$ cycle. Puisque le noeud c_3 est dans la position 3 (en comptant depuis 0) sur la chaîne (position impaire), tous les nœuds en position égale enverront à leur voisin de droite.

Les noeuds qui reçoivent à chaque niveau s'élèvent au niveau suivant dans la hiérarchie. Maintenant au niveau suivant, le nœud c_3 est toujours dans une position impaire (1). Encore, tous les nœuds dans la même position vont agréger leurs données avec celles reçues et vont ensuite les envoyer à leur droite. Au troisième niveau, le nœud c_3 n'est pas en position impaire. Ainsi le nœud c_7 va agréger ses données et transmettra à c_3. Finalement, c_3 va combiner ses données courantes avec celles reçues de c_7 et transmettra le message à la BS.

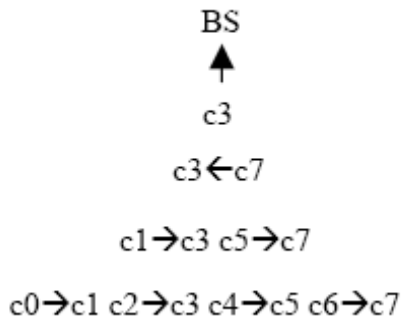


***Figure 19**: collecte de données dans une chaîne à arrangement binaire.*

3. Motivation et objectives

Après avoir étudié les trois types de protocoles de routage (*données centrales, géographique et hiérarchique*), on peut déduire que le routage hiérarchique est le plus approprié au RCSFs en terme d'efficacité énergique. En effet, deux grands algorithmes se partagent le domaine d'application du routage hiérarchique, proposant chacun une approche différente d'organisation du réseau à savoir l'algorithme LEACH avec l'organisation à grappe (Clustering), ainsi que l'algorithme PEGASIS avec l'organisation à chaîne (Chaine-based approach).

L'approche à grappe (clustering) avec LEACH a montré son efficacité comparé aux autres approches (*centrés données, géographique*), en termes de consommation et de dissipation uniforme d'énergie prolongeant ainsi le temps de vie du réseau, En effet, LEACH (cluster-based approach) réalise au-dessus d'un facteur de réduction de 7 dans la dissipation d'énergie, comparé à la communication directe (direct diffusion), et un facteur de 4-8, comparé au protocole de routage de transmission minime d'énergie. A l'inverse, un nombre d'inconvénients restent plus au moins apparents. Nous citons parmi eux :

- Les nœuds les plus éloignés du CH meurent rapidement par rapport à ceux plus proches ;

- Le nombre de messages reçus par les CHs équivaut, approximativement, celui des nœuds gérés. Ceci mène à l'épuisement rapide de leur réserve d'énergie ;
- L'utilisation de singel-hop (un saut) au lieu de multi-hop (multi sauts) épuise rapidement l'énergie des nœuds et consomme d'avantage de largeur de bande.

A la différence de LEACH, PEGASIS (chaine-based approach) évite la formation de grappes et organise les nœuds du réseau sous forme d'une chaîne où un seul nœud est sélectionné afin de transmettre à la BS, au lieu d'en utiliser plusieurs. Ceci réduit l'énergie nécessaire à la transmission des données par cycle, du moment que la dissipation d'énergie est distribuée uniformément au-dessus de tous les nœuds. Les distances que la plupart des nœuds transmettent sont beaucoup moins réduites par rapport à la transmission au leader (CHs) dans LEACH. En second lieu, la quantité de données à recevoir par le leader est deux messages au plus au lieu de 20 (20 nœuds par grappe dans LEACH pour un réseau de 100 noeuds).

Bien que le 'clustering' soit évité, PEGASIS présente un retard excessif (extrême latence) pour le noeud le plus éloigné sur la chaîne. PEGASIS suppose que chaque noeud capteur doit pouvoir communiquer directement avec la BS. Dans des cas pratiques, les nœuds capteurs utilisent la communication multi sauts pour atteindre la station de base. En outre, PEGASIS suppose que les nœuds capteurs ont tous le même niveau d'énergie et q'ils sont susceptibles de mourir en même temps.

Après avoir analysé les deux algorithmes (LEACH et PEGASIS), on a constaté qu'on peut améliorer le premier protocole (LEACH) on appliquant le concept du deuxième protocole (PEGASIS) au niveau de la grappe (cluster), ce qui nous amène à proposer un nouveau protocole hybride combinant les avantages des deux grandes approches à savoir (Cluster-based approach) et (Chain-based approach).

En fait, l'idée de combinaison a déjà été proposée par T. Thua Huynh et C. Seon Hong dans [10] au niveau des cluster heads où une chaîne de proche voisin est construite a partir des CHs de chaque cluster affin d'éviter que les CHs les plus éloignés de la BS ne meurent plus rapidement. D'un autre point de vue, notre approche applique le principe de PEGASIS à l'intérieur du cluster où la formation d'une chaîne entre les nœuds appartenants à la même grappe améliore la dissipation de l'énergie. Les nœuds communiquent uniquement avec leurs proches voisins et non pas directement avec leur CH ce qui économise d'avantage d'énergie.

Les CHs reçoivent moins de messages et vivent ainsi plus longtemps. Chaque CH envoie les données collectées au sein du cluster indirectement à la BS en utilisant l'approche multi sauts à travers les CHs voisins, afin de réduire la consommation énergétique. Ainsi, l'idée de combinaison va permettre de prolonger le temps de vie du réseau, comparé à LEACH, et de réduire significativement le retard excessif (latence) introduit par PEGASIS.

4. Conclusion

L'étude et l'analyse des principaux protocoles et approches de routage pour les RCSFs nous a permis de proposer notre propre protocole de routage dont l'objectif principal est le prolongement du temps du vie du réseau ainsi que la gestion efficace de la consommation énergétique. Le concept de base ainsi que l'architecture de fonctionnement de ce dernier vont être présentés dans le chapitre qui suit.

1. Introduction

Dans ce chapitre, nous allons présenter notre protocole hybride son architecture de fonctionnement ainsi que son concept de base afin de mieux expliquer la manière par laquelle on combine l'approche de communication à chaîne inspirée du protocole PEGASIS *(Power-Efficient Gathering in Sensor Information Systems)* avec l'approche de communication à grappe proposée par le protocole LEACH *(Low Energy Adaptive Clustering Hierarchy)*.

2. Concept de base de notre protocole

L'organisation des nœuds appartenant à la même grappe (Cluster) sous forme d'une chaîne permet d'améliorer et de réguler la dissipation d'énergie, ce qui permet de réduire la charge sur le CH (cluster-head). En effet, les nœuds communiquent uniquement avec leurs proches voisins et non pas directement avec leur CH, ce que économise d'avantage d'énergie et offre une meilleure utilisation de la largeur de bande. L'agrégation des données au niveau de chaque nœud dans la chaîne réduit la quantité de données échangées entre les nœuds et leur CH, ce qui a pour effet de préserver les réserves d'énergie de ces derniers.

La figure 20 montre comment les nœuds seront organisés à l'intérieur des grappes (Clusters), le nœud N_0 transmet ses données à son proche voisin N_1, N_1 quant à lui agrège les données reçues avec les siennes et les transmet à son autre voisin jusqu'à atteindre le CH qui les transmet directement à la BS ou en utilisant l'approche multi sauts pour préserver d'avantage d'énergie. Donc, dans cette nouvelle organisation (grappe à chaînes), tous les nœuds du cluster vont transmettre leurs données colletées à leurs CHs respectifs en se reliant à travers la chaîne, tandis que chaque CH doit recevoir les données collectées du nœuds entête de la chaîne.

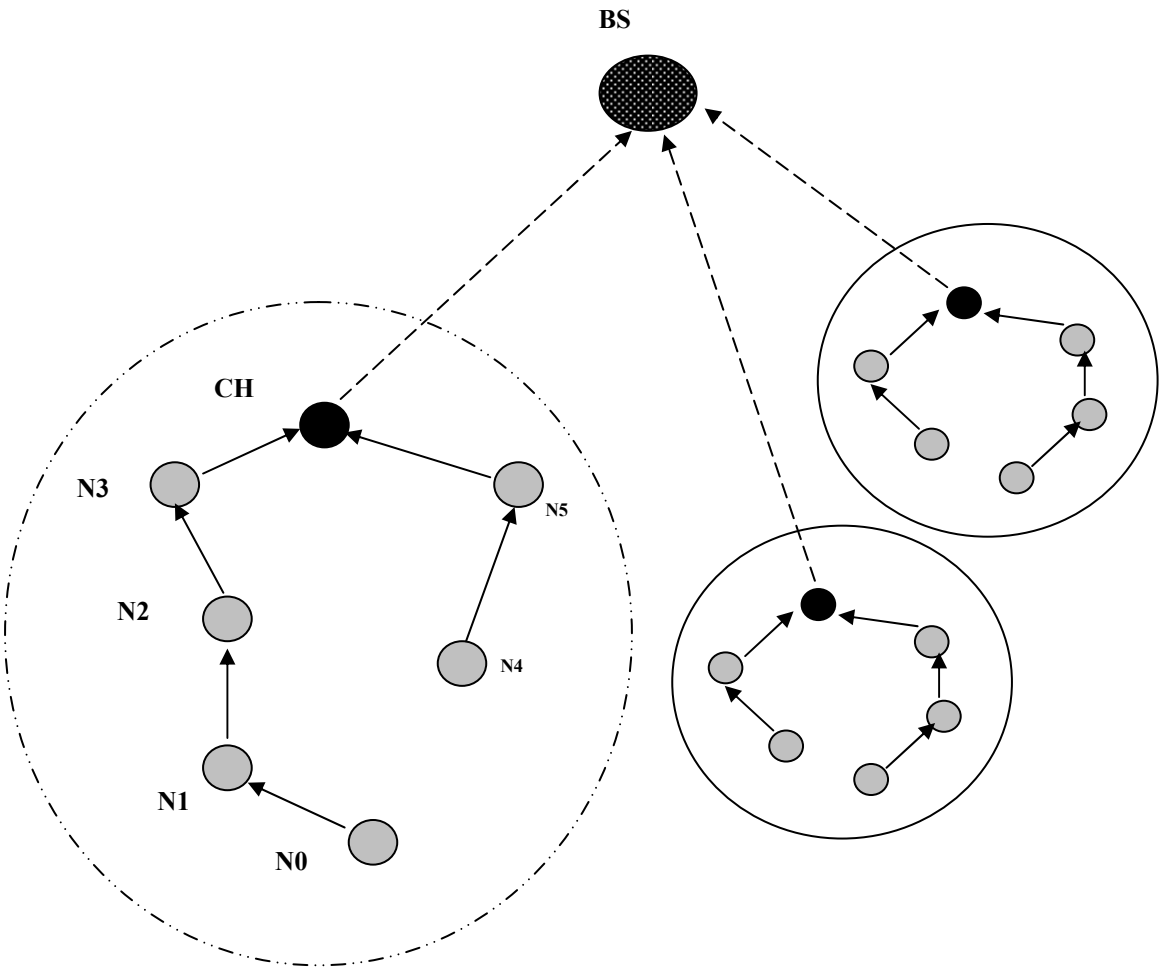


Figure 20 : Organisation des nœuds dans le réseau.

A l'inverse de LEACH [3], le nombre de nœuds qui communiquent avec le CH est considérablement réduit. Ceci implique une meilleure économie d'énergie et prolonge d'avantage le temps de vie des CHs, car si ces derniers meurent (épuisent leur réserve d'énergie), tous les nœuds du cluster vont perdre leur pouvoir de communication avec la BS et par conséquent le cluster tout entier est considéré comme invalide (ne communique pas avec la BS)

Dans notre protocole, nous avons adopté le concept de la rotation aléatoire du rôle de CH proposé par LEACH, qui régule la dissipation d'énergie et évite que les nœuds choisis comme CHs meurent plus rapidement. Cependant, par opposition à LEACH, nous avons réutilisé le concept de PEGASIS [7, 8] en organisant les nœuds du cluster sous forme de chaîne, ce qui a

pour effet de prévenir que les noeuds les plus éloignés des CHs épuisent leur réserve d'énergie

En ce qui concerne l'accès au médians (*Media access*), nous avons utilisé la même méthode proposée dans LEACH [3], où les CHs établissent un plan de transmission (*TDMA schedule*) qui attribut à chaque nœud le temps exact pendent lequel il doit transmettre ses données collectées. Cela permet aux nœuds d'éteindre leurs antennes radio et passer à l'état endormi, ce qui va permettre d'économiser plus d'énergie. De plus, l'utilisation du *TDMA schedule* va nous permettre d'éviter les problèmes de collusion et d'interférence entre les nœuds du cluster.

3. Approche de formation de grappes à chaînes

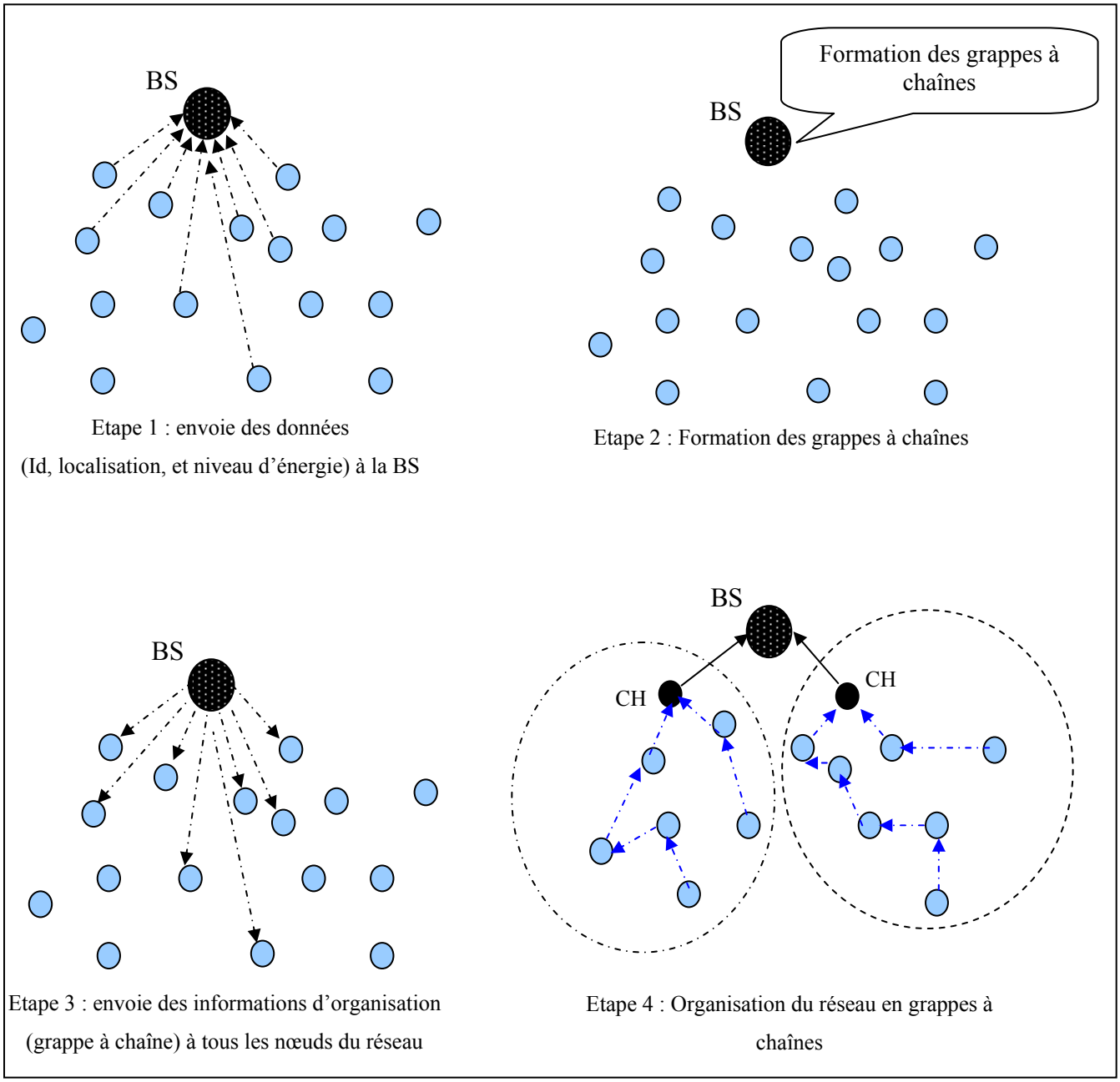
La formation de grappes et de chaînes peut être effectuée d'une manière centralisée par la station de base ou d'une manière distribuée par les CHs (cluster-head). Pour obtenir des résultats meilleurs en terme de répartition égale des noeuds entre les clusters, on a choisi l'approche centralisée proposée dans le protocole LEACH-C [3] où chaque nœud envoi un paquet de données à la BS contenant l'identificateur du nœud, la réserve d'énergie et la localisation sur le réseau (en utilisant par exemple le système de localisation GPS). La station de base va exécuter un algorithme d'optimisation afin de former les grappes (clusters). L'algorithme de détermination du cluster optimal est algorithme d'approximation NP-complexe (NP-HARD), comme par exemple l'algorithme de recherche TABOO, ou l'algorithme de réussite simulée (*simulated annealing*), qui ne nous donnent pas des résultats exactes mais proches de l'optimal. Dans notre algorithme hybride, nous avons opté pour le même algorithme d'optimisation utilisé dans LEACH-C à savoir la réussite simulée [11]. En effet, cet algorithme est découvert en 1982 par S. Kirkpatrick dans lequel le choix des solutions est basé sur une loi de probabilité (*distribution de Bolztmann*) qui mesure la probabilité $P(X)$ de visiter l'état X en fonction de son énergie $E(X)$ et de la température T

$$P(X) = e^{-\frac{E(X)}{kT}}/N(T) \qquad (6)$$

K : est bien sur la constante de Bolztmann.

Donc, cette méthode de formation de grappe centralisée permet de déterminer, à partir de la position exacte des noeuds, la configuration optimale pour minimiser l'énergie dépensée. Dès

que les grappes sont formées, la station de base va passer à l'élision de CHs. Ces derniers sont choisis d'une manière très simple où seul le nœud qui a la plus grande réserve d'énergie est éligible de devenir le prochain cluster head. On peut améliorer le choix du CH en ajoutant des critères de choix supplémentaires comme par exemple la position du CH par rapport aux têtes de chaînes, c'est-à-dire choisir comme CH le nœud qui a la position la plus proche pratiquement de tous les nœuds à partir desquels il va recevoir les données



Figue 21: Etapes de formation des grappes à chaînes.

.

En ce qui concerne la formation de chaînes, on a adopté la même idée utilisée dans PEGASIS où les nœuds de la même grappe forment une chaîne de proches voisins. Chaque nœud reçoit des donnée de l'un de ses voisins, fusionne (agrège) les données de ces derniers avec ses propres données et les envoi à son tour à son autre voisin dans la chaîne. L'opération d'agrégation est exécutée au niveau de chaque nœud afin d'éliminer les informations redondantes et de réduire la quantité de données échangées pour préserver l'énergie. Afin de réduire le degré de latence introduit par la chaîne, on peut permettre à quelques nœuds de transmettre simultanément. Pour éviter les problèmes d'interférence, on peut codifier les transmissions ou bien permettre aux nœuds séparés géographiquement d'émettre au même slot de temps.

Pour construire la chaîne, on a proposé un algorithme simplifié, détaillé dans la figure 22 et la figure 23. Ce dernier suit le même concept utilisé dans PEGASIS où l'on commence avec le nœud le plus éloigné de la BS (choisir un noeud aléatoirement s'il y a une cravate c'est-à-dire un ensemble de nœuds possédant la même distance par rapport à la BS). Ce nœud représente la tête de la chaîne.

Ensuite, le nœud le plus proche de la tête de la chaîne est choisi pour être ajouté et devenir ainsi la nouvelle tête de la chaîne. Les voisins successifs sont sélectionnés de cette manière parmi les noeuds non visités (avec une cravate brisée arbitrairement). L'opération se répète jusqu'à ce que tous les nœuds soient dans la chaîne.

En effet, l'algorithme proposé commence par le noeud le plus lointain pour s'assurer que les nœuds les plus loin du CH ont des voisins proches. Les distances voisines augmenteront graduellement puisque des noeuds déjà présents sur la chaîne ne peuvent pas être revisités. La figure 20 montre le noeud N_0 se reliant au noeud N_1, le noeud N_1 se reliant au noeud N_2, et le noeud N_2 se reliant au noeud N_3, dans cet ordre jusqu'à atteindre le CH. Quand un nœud meure, la chaîne est reconstruite de la même manière pour évier le noeud mort.

Afin de collecter les données des nœuds capteurs dans chaque cycle, chaque nœud reçoit les données d'un voisin, les fusionne avec les siennes et transmet à un autre voisin dans la chaîne jusqu'à atteindre le CH. Chaque cycle de collecte de données est lancé par la BS avec un

signal de balise qui synchronisera tous les noeuds capteurs. Puisque tous les nœuds connaissent leurs positions sur la chaîne, on a employé une approche de slot de temps (TDMA) pour la transmission des données. Le noeud N_0 de fin transmettra ses données au noeud N_1 dans le premier slot, N_1 fusionne et transmet les données dans le deuxième slot, et ainsi de suite jusqu'à ce que le nœud leader (CH) soit atteint. Finalement, dans le $n^{ième}$ slot, le CH transmet les données à la BS

```
DEBUT
  Chaîne : la chaîne à construire,
  Tête : la tête de la chaîne,
  NLCH () : procédure qui retourne le nœud le plus loin,
  NP () : procédure qui retourne le nœud le plus proche,
  N : ensemble des nœuds appartenants à la même grappe,

  Chaîne = {}
  Tête = NLCH (CH)            {trouver le nœud le plus éloigné du CH}
  N = N – Tête

    Tant Que (N ≠ Φ) Faire

      Tête = NP (Tête)        {trouver le nœud le plus proche de la tête de chaîne}
      Ajouter (Chaîne, Tête)
      Supprimer (N, Tête)

    Fin Tant Que
Fin
```

Figure 22 : Algorithme de construction des chaînes.

.

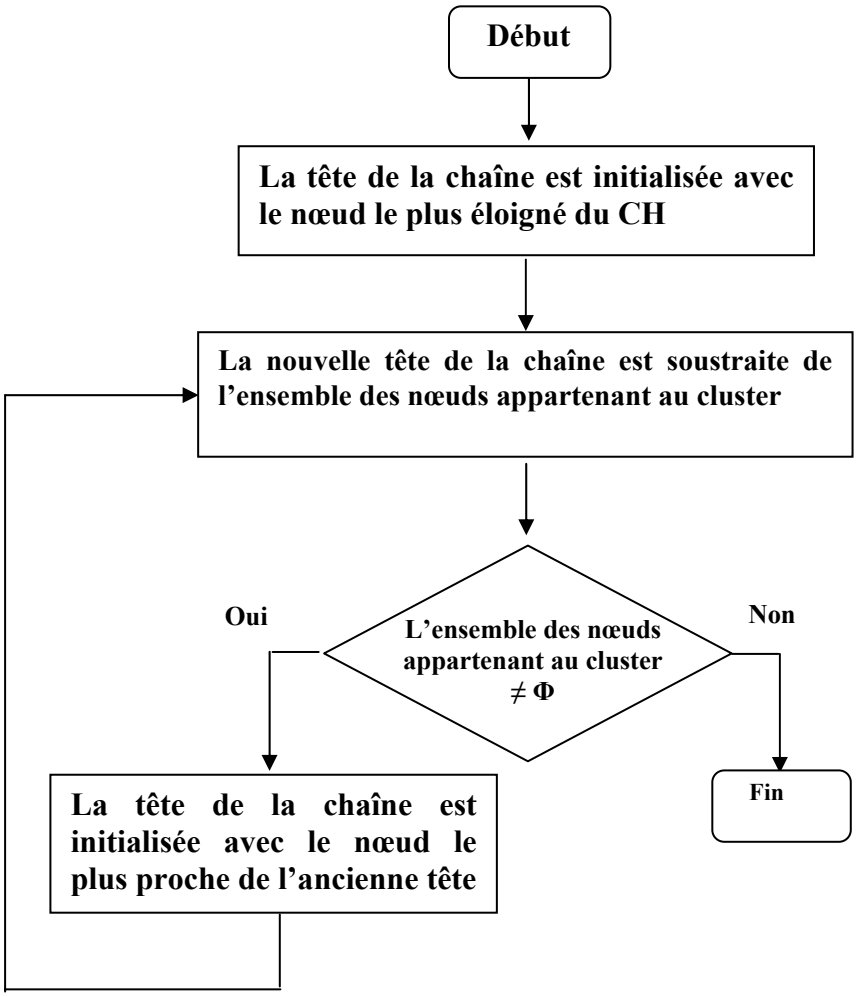


Figure 23 : Organigramme de construction des chaînes.

Remarque :

Afin d'améliorer notre approche, il serait plus avantageux de diviser la chaîne de nœuds en de plus petites chaînes dans lesquelles les nœuds envoient les données à leurs voisins, ou directement aux CHs, si ces dernier sont les plus proches. Par conséquent, les distances de transmission de données entre les nœuds seront réduites, ce qui a pour effet de consommer moins d'énergie. De plus, la latence introduite par la chaîne va baisser considérablement. La figure 24 montre comment la chaîne de nœuds sera divisée.

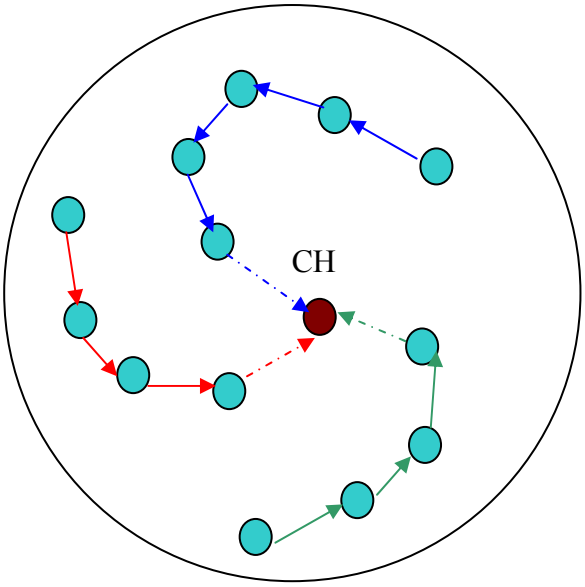


Figure 24 : Division de la chaîne de nœuds.

4. Les grandes étapes de notre algorithme

Le déroulement de notre protocole hybride est devisé en plusieurs cycles d'exécution. Chaque cycle commence par une phase d'initialisation dans laquelle les cluster à chaînes sont formés et les CHs sont élus, suivie d'une phase de transmission où les données collectées sont transmises à travers les chaînes aux CHs qui vont à leur tour transmettre à la station de base. Les nœuds doivent être tous synchronisés de façon à participer à la phase d'initialisation en même temps.

Afin de minimiser les problèmes d'interférence et d'overhead, la durée de la phase d'initialisation est fixée de façon à être beaucoup plus petite par rapport à la phase de transmission.

Figure 25 : Etapes d'exécution de notre protocole.

4.1 Etape d'initialisation :

L'étape d'initialisation commence par la création des grappes, dans laquelle on adopte la même approche centralisée utilisée dans LEACH-C, et où la station de base utilise la réussite simulée pour former les grappes. Cette approche offre un résultat meilleur, par rapport à l'approche distribuée, utilisée dans LEACH, en terme de formation de grappes et de préservation d'énergie. Après la formation des grappes, les CHs sont choisis d'une manière simplifiée où seul le nœud qui a la plus grande réserve d'énergie, parmi les nœuds de la même grappe, est élu. Ensuite, on aborde la construction des chaînes où l'on suit une méthode centralisée dans laquelle la station de base utilise les informations envoyées par les nœuds pour former les chaînes en appliquant l'algorithme de formation de chaînes proposé précédemment.

On utilise la technique de multiplexage temporel comme moyen d'accès au médium. Cette technique consiste à construire une table TDMA pour partager le temps de transmission sur le nœud. Etant donné que chaque nœud connaît d'avance le slot de temps qu'il va occuper, cela permet alors au nœud de passer à l'état « endormi » durant les slot inactifs. La table TDMA ainsi créée, pour chaque grappe, sera diffusée pour tous les nœuds lui appartenants.

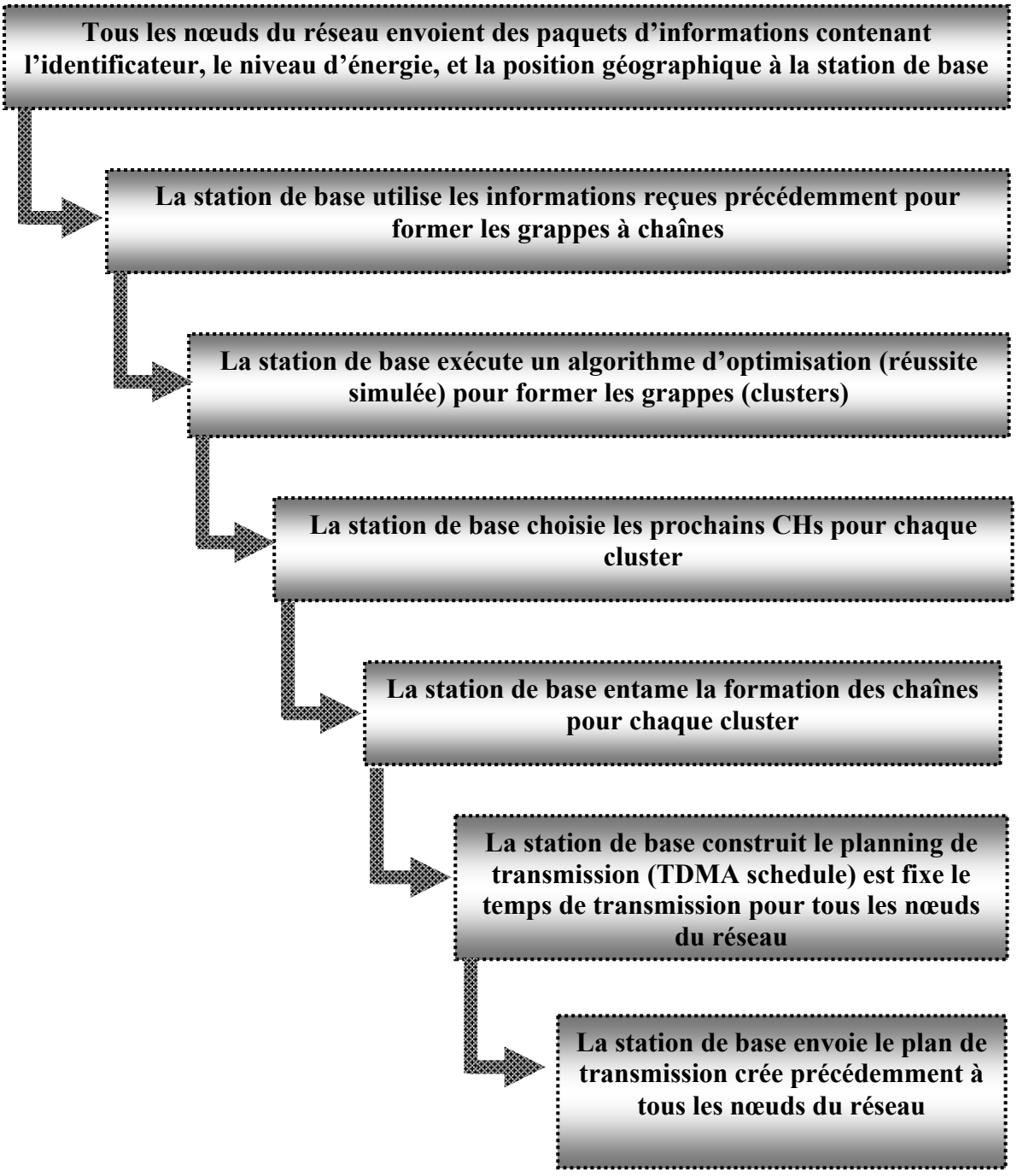


Figure26 : Opérations de l'étape d'initialisation.

4.2 Etape de transmission :

L'étape de transmission est devisée en plusieurs itérations dans lesquelles les nœuds vont transmettre leurs données colletées à travers la chaîne aux CHs. Dans chaque itération, un nœud transmet au moins un paquet de données durant son slot de temps alloué précédemment par la station de base. Sachant que le slot de temps alloué à chaque nœud est constant, le temps de chaque itération de transmission va dépendre évidemment du nombre de nœuds existants dans le cluster.

Afin de réduire l'énergie consommée durant la transmission des données, chaque nœud va régler la puissance de son antenne radio de façon à pouvoir transmettre uniquement à ses proches voisins, à l'inverse du protocole LEACH où les nœuds les plus loin des CHs perdent beaucoup d'énergie afin de pouvoir transmettre leurs données. L'utilisation des TDMA schedule crées par la station de base va permettre aux nœuds de pouvoir éteindre leurs antennes radio en attendant leur temps de transmission. Cela permet alors au nœud de passer à l'état « endormi » durant les slot inactifs. Ainsi, la perte d'énergie due aux états de sur écoute (overhearing) et d'écoute passive (idle) est évitée. Par opposition à LEACH, les CHs peuvent éteindre leurs antennes jusqu'à l'arrivée des données collectées par la chaîne des nœuds et cela nous permet de préserver plus d'énergie.

Chaque nœud transmet, durant son slot de temps, les données collectées à son proche voisin dans la chaîne. Comme il est montré dans la figure 20, le dernier nœud de la chaîne n_0 va transmettre ses données au nœud n_1 durant le premier slot de temps. n_1 agrège les données de n_0 avec les siennes et transmet à son autre voisin dans le deuxième slot de temps et ainsi de suite jusqu'à ce que l'on atteigne le CH

On peut utiliser une simple approche pour contrôler la transmission tout le long de la chaîne où le CH envoi un paquet de contrôle (Jeton) afin de coordonner la transmission et de la démarrer au niveau des extrémités de la chaîne débutant par la droite. Le coût de cette approche est très petit car la taille du paquet de contrôle (Jeton) n'est pas grande.

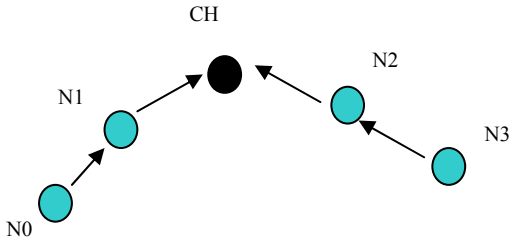


Figure. 27. Approche de contrôle de transmission

Dans la figure 27, le CH transmet le paquet de contrôle en premier lieu au nœud N_0 qui va transmettre ses données collectées à son voisin N_1. N_1 quant à lui agrège les données reçues avec les siennes et les transmet au CH. Après que le CH ait reçu les données de la partie droite de la chaîne, il exécute la même opération de l'autre partie. Enfin, le CH envoi les données reçues à la BS.

Le choix du protocole de transmission au niveau de la couche MAC est essentiel dentant plus que la transmission sans fil est affectée par les problèmes de parasites, terminal caché, et d'interférences. La communication dans un cluster peut encore être affectée par les clusters voisins. Par conséquent, nous avons opté pour le même protocole de transmission utilisé dans LEACH à savoir DS-SS (*direct-sequence spread spectrum*) [3] où l'on attribut à chaque cluster un code unique. Tous les nœuds appartenant au même cluster vont utiliser ce code pour transmettre au CH et ce dernier va utiliser le code attribué au cluster pour filtrer les données reçues. De cette manière, on évitera que le CH d'un cluster va réceptionner les données transmises à partir des nœuds des clusters voisins.

De cette façon, la combinaison de l'idée de TDMA schedule et le protocole DS-SS va réduire les problème d'interférences intra-cluster, car les nœuds ne transmettent que pendant leur slot de temps de manière à ce que les nœuds proches ne transmettent pas en même temps, ainsi que les problème d'interférences extra-cluster car les CHs filtrent les données reçues afin de détecter les données intruses.

En ce qui concerne la transmission à la BS, les CHs envoient leurs données en utilisant une approche multi sauts si ces derniers sont loin, ou directement pour le cas contraire.

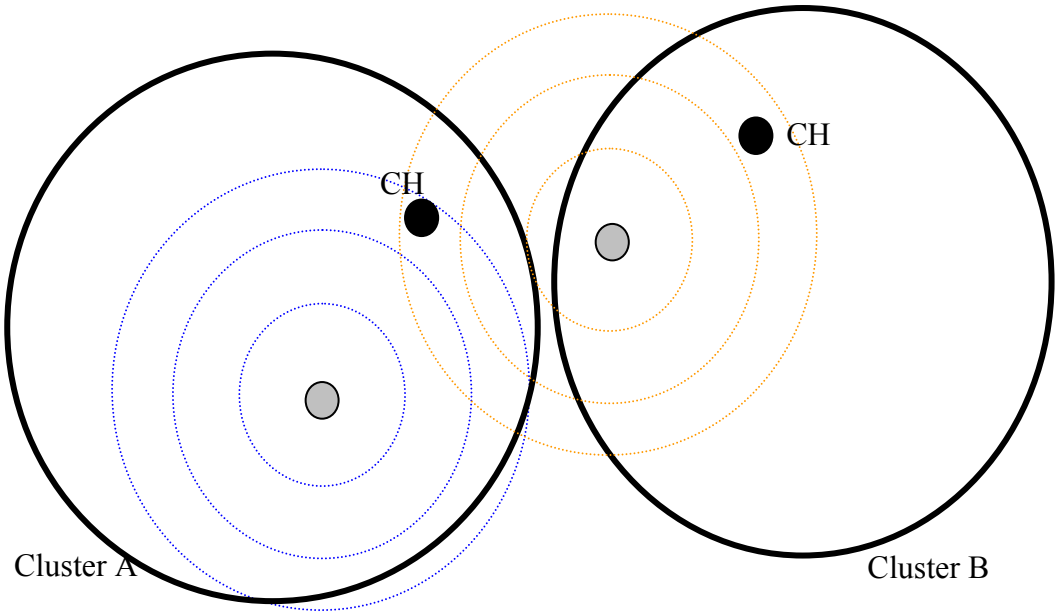


Figure 28 : problèmes d'interférence entre les clusters

L'agrégation des données [6] qui permet d'éliminer les informations dupliquées (compression des données) est effectuée au niveau de chaque nœud dans la chaîne contrairement au protocole LEACH où seul le CH est chargé d'agréger les données qui vont être transmises à la station de base. De cette manière, notre protocole hybride réduit la quantité de données envoyées au CH, ce qui implique évidemment le prolongement de la durée de vie de ces derniers.

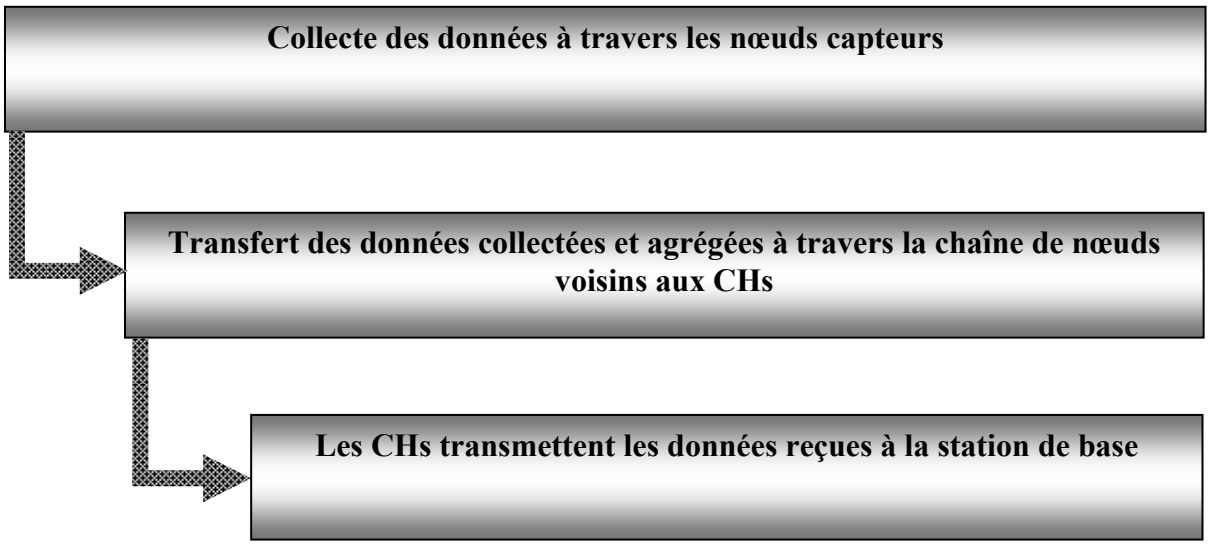


***Figure 29**: Opérations de l'étape de transmission.*

5. Conclusion

Après avoir présenté le concept de base ainsi que l'architecture de fonctionnement de notre protocole de routage hybride, on va entamer dans le chapitre qui suit l'étape d'implémentation qui constitue un point important dans le processus d'analyse des performances de notre algorithme.

1. Introduction

Afin d'implémenter notre protocole hybride, nous avons réutilisé le code d'implémentation du protocole LEACH et LEACH-C, et effectué certaines modifications à plusieurs niveaux. Ces modifications consistent en l'injection de nouveaux morceaux de code comme par exemple la procédure de formation de chaînes, procédure de recherche des nœuds les plus proches (voisins proches), procédure de recherche de nœuds les plus loin, procédure de création de TDMA schedule pour les chaînes de nœuds, ….etc.

2. Choix du langage et de l'environnement d'implémentation

Nous avons choisi comme langage d'implémentation pour notre protocole hybride le langage TCL (*Tool Command Language*) [33, 34]. Ce dernier est connu comme étant un langage de commandes *interprété et* extensif. En effet, les programmes écrits en TCL sont des fichiers texte constitués de commandes TCL qui sont traités via un interpréteur TCL au moment de l'exécution. L'avantage d'implémenter notre algorithme en TCL est de pouvoir facilement l'interpréter avec l'interpréteur TCL intégré dans le simulateur NS2 [12], ce qui nous permet de simuler son fonctionnement afin d'évaluer ses performances.

Nous avons choisi comme plate- forme d'implémentation, le simulateur NS2 [12] version NS2.30, sous le système d'exploitation *LUNIX MANDRIVA 2007*. Ce choix est basé sur le fait que NS2 est le leader en terme d'extensibilité puisque on peut facilement ajouter nos propres protocoles, que ce soit au niveau de la couche réseau, mac, application, etc.… De plus, NS2 est bâti selon les idées de conception par objets, de réutilisation du code et de modularité. Il est devenu aujourd'hui un standard de référence dans ce domaine, son utilisation est gratuite et il est exécutable tant sous UNIX que sous Windows. NS2 est construit autour du langage de programmation TCL dont il est une extension. Le noyau de ce dernier est implémenté en commandes C++.

Du point de vue utilisateur, la mise en œuvre de ce simulateur se fait via une étape de programmation qui décrit la topologie du réseau et le comportement de ses composant.

Ensuite, vient l'étape de simulation proprement dite et enfin l'interprétation des résultats. La description de la topologie du réseau peut être facilement effectuée en utilisant des primitives de base comme par exemple *Nodes*, *Links*, *Agents*, et *Applications*, où la primitive *Nodes* représente un nœud dans le réseau, la primitive *Links* représente le support de communication, la primitive *Agents* est utilisée pour implémenter différents protocoles réseaux, et la primitive *Applications* est chargée de la génération des données ainsi que la réalisation de plusieurs tâches spécifiques.

3. Etapes d'implémentation de notre protocole

L'implémentation de notre protocole passe par plusieurs étapes à savoir :

3.1. Préparation de l'environnement d'implémentation

La préparation de l'environnement d'implémentation consiste à installer le simulateur de réseau NS2 sous le système d'exploitation *LUNIX MANDRIVA 2007*. On a choisi la version NS2.30 puisqu'elle prend en considération la topologie des réseaux sans fil. L'installation de NS2.30 sous *LUNIX MANDRIVA 2007* s'effectue suivant 3 étapes qui se résument en :

- ✓ Copier le package d'installation NS2.30 dans le répertoire USER du système ;
- ✓ Taper la commande d'installation ***'./Install'*** dans le terminal de commandes, dans le répertoire NS2.30 précédemment copié ;
- ✓ Modifier les variables d'environnement et cela consiste à ajouter les lignes de code à suivre dans le fichier '***.bashrc' ;***

```
export
PATH=${PATH}:/home/user/ns-allinone-2.30/bin:/home/user/
ns-allinone-2.30/tcl8.4.13/unix:/home/user/ns-allinone-
2.30/tk8.4.13/unix:/home/user/ns-allinone-2.30/nam-1.12

export
LD_LIBRARY_PATH=/home/user/ns-allinone-2.30/otcl-1.12,
/user/ns-allinone-2.30/lib

export
TCL_LIBRARY=/user/ns-allinone-2.30/tcl8.4.13/library
```

- ✓ Et enfin taper la commande ***'./make'*** dans le terminal de commandes afin de compiler NS2 et de générer tous les fichiers NS2 nécessaires à son fonctionnement.

Après avoir installé le simulateur de réseau NS2, il faut installer le protocole de routage LEACH. Pour ce faire, il faut suivre les étapes suivantes :

- ✓ Obtenir le package d'installation du protocole LEACH qu'on peut trouver sur le site Internet : http://www.internetworkflow.com/downloads/ns2leach/mit.tar.gz.;
- ✓ Placer le package d'installation dans le chemin suivant : 'user/ns-allinone-2.30/ns-2.30' ;
- ✓ Décompresser le package d'installation en tapant la commande ***'gunzip mit.tar.gz'*** et puis la commande ***'tar -xvf mit.tar'*** dans le terminal de commandes ;
- ✓ Ajouter les lignes de code à suivre dans la fiche de compilation NS2 nommée '***Makefile'*** qui se trouve dans le répertoire ns-allinone-2.30.

- Ajouter la ligne de commande ***'DMIT_uAMPS'*** dans la partie de déclaration ***'DEFINE list'***
- Ajouter la ligne de commande ***'I./mit/rca I./mit/uAMPS'*** dans la partie ***'INCLUDE list'***
- Ajouter les lignes de commande à suivre juste avant la ligne de commande ***'gaf/gaf.o \'***

***'mit/rca/energy.o mit/rca/rcagent.o ***
***mit/rca/rca-ll.o mit/rca/resource.o ***
mac/mac-sensor-timers.o mac/mac-sensor.o mit/uAMPS/bsagent.o \'

- ✓ Taper la commande 'make clean' dans le terminal de commandes afin d'effacer l'ancien fichier de compilation.
- ✓ Et enfin recompiler de nouveau NS2 afin de prendre en considération l'ajout du nouveau protocole de routage LEACH en tapant la commande ***'nohup make 2>error.log >make.log &'*** dans le terminal de commandes.

3.2. Implémentation de notre protocole hybride

Après avoir préparé notre environnement d'implémentation, on entame l'étape d'implémentation de notre protocole qui consiste à modifier le code d'implémentation du protocole LEACH afin qu'il corresponde à l'architecture de fonctionnement de notre protocole. Pour cela, on a ajouté diverses fonctions et procédures telles que : la procédure de formation de chaînes, procédure de recherche des nœuds les plus proches (voisins proches), procédure de recherche de nœuds les plus loin, procédure de création de TDMA schedule pour les chaînes de nœuds, ….etc.

En effet le, code d'implémentation LEACH est composé de plusieurs fichiers TCL dont chacun joue un rôle bien précis. Ainsi, et pour réutiliser ce dernier, nous avons dû analyser et comprendre le rôle de chacun de ces fichiers. Etant donné qu'un fichier TCL est présenté comme une classe qui contient plusieurs sous classes, le protocole LEACH est vu comme un ensemble de classes qui se communiquent pour réaliser une tâche commune.

Le protocole LEACH est une application spécifique des protocoles de routage de données. Il est implémenté comme sous classe par rapport à la classe *Applications* du simulateur NS2. En ce qui concerne le protocole LEACH-C, il est implémenté comme sous classe du protocole LEACH du fait qu'il utilise les mêmes fonctions.

3.2.1. Procédure de formation de chaîne

La procédure de formation de chaîne de noeuds capteurs est la plus importante étape dans l'implémentation de notre protocole, puisque en doit changer carrément le mode de fonctionnement du protocole LEACH. Etant donné que l'implémentation du protocole LEACH suit une méthodologie orientée objets, La procédure de formation de chaîne est implémentée comme un sou classe de la classe LEACH-C. Elle reçoit comme paramètres d'appelle l'ensemble de nœuds qui forment le cluster ainsi que l'identificateur du cluster head associé à ce dernier (cluster). La figure 30 présente le code d'implémentation de la procédure de formation des chaînes sous le langage TCL :

```
LEACH/LEACH-C instproc chaine {tdma chid} {

#declaration des variables globales
global ns_ opt tabdist tabchaine nodech nodechaine
#declaration des variables locales
$self instvar nbr_node farnode nearnode r cord dn dch

set tdma [sup $tdma $chid]  #définir les nœuds appartenant au cluster
set nbr_node [llength $tdma]#définir le nombre de noeuds appartenant au cluster
set nodechaine ""

if {$nbr_node > 0} {
set farnode [$self far_node $chid $tdma  ]#recherche du noeud le plus loin du
CH
set tdma [sup $tdma $farnode]
set nodechaine "$farnode" #initialiser la chaîne avec le noeud le plus loin
#recherche du noeud le plus proche de la tête de chaîne de noeuds
set nearnode [$self near_node $farnode $tdma $chid  ]
#obtenir les cordonnées x,y du CH
set cord $tabdist($chid)
set XCH [lindex $cord 0]
set YCH [lindex $cord 1]
#obtenir les cordonnées x,y du noeud le plus loin du CH
set cord $tabdist($farnode)
set X1 [lindex $cord 0]
set Y1 [lindex $cord 1]
#obtenir les cordonnées x,y du noeud le plus proche de la tête de la chaîne
set cord $tabdist($nearnode)
set X2 [lindex $cord 0]
set Y2 [lindex $cord 1]
#calculer la distance entre le CH et la tête de la chaîne
set dch [dist $X1 $Y1 $XCH $YCH]
#calculer la distance entre le noeud le plus proche et le CH
set dn [dist $X1 $Y1 $X2 $Y2]

set info "$nearnode 0 $dch $dn"          # Ajouter au tableau de la chaîne la distance du
set tabchaine($farnode) $info            # nœud le plus loin du CH et la distance du nœud
set r [$self remptab $info $farnode]     # le plus proche de la tête de chaîne, ainsi que leurs
                                         # identificateurs respectifs
set nodech $nearnode
set tdma [sup $tdma $nearnode]
set nodechaine [linsert $nodechaine end $nearnode]

#entamer la boucle de construction de chaîne
set i 1
while {[llength $tdma] != 0} {

set nearnode [$self near_node $nodech $tdma $chid  ]
#obtenir les cordonnées x,y du CH
set cord $tabdist($chid)
set XCH [lindex $cord 0]
set YCH [lindex $cord 1]

#obtenir les cordonnées x,y de la tête de la chaîne
set cord $tabdist($nodech)
set X1 [lindex $cord 0]
set Y1 [lindex $cord 1]
#obtenir les cordonnées x,y du noeud le plus proche de la tête de la chaîne
set cord $tabdist($nearnode)
set X2 [lindex $cord 0]
set Y2 [lindex $cord 1]
```

```
#calculer la distance entre le CH et la tête de la chaîne
set dch [dist $X1 $Y1 $XCH $YCH]
#calculer la distance entre le noeud le plus proche et le CH
set dn [dist $X1 $Y1 $X2 $Y2]

set info "$nearnode $i $dch $dn"
set tabchaine($nodech) $info
set r [$self remptab $info $nodech]
set tdma [sup $tdma $nearnode]
set nodechaine [linsert $nodechaine end $nearnode]
set nodech $nearnode
    incr i 1
}

set cord $tabdist($chid)
set XCH [lindex $cord 0]
set YCH [lindex $cord 1]

set cord $tabdist($nodech)
set X1 [lindex $cord 0]
set Y1 [lindex $cord 1]


set dch [dist $X1 $Y1 $XCH $YCH]

set info "$chid $i $dch $dch"
set tabchaine($nodech) $info
set r [$self remptab $info $nodech]
set indice [lsearch $nodechaine $chid]

if {$indice == -1} {
set nodechaine [linsert $nodechaine end $chid]
} else {
set nodechaine [sup $nodechaine $chid]
}

} else {

    puts "tdma vide."
}
set nodechaine [sup $nodechaine $chid]

}
```

Figure 30 : Code d'implémentation de la procédure de formation des chaînes

L'invocation de la procédure de formation de chaîne est effectué au niveau de la classe '*recvBS_CH_INFO*' après l'élection des CH en ajoutant le morceau de code suivant :

```
'$self chaine $tdma(1) $id'
```

La procédure de formation de chaîne invoque deux autre procédure importants à savoir la procédure de recherche du nœud le plus loin de la chaîne et la procédure de recherche de nœud le plus proche de la tête de la chaîne de nœuds.

3.2.2. Procédure de recherche du nœud le plus loin de la chaîne

La procédure de recherche du nœud le plus loin est invoquée tout au début de l'algorithme de formation de chaîne afin de déterminer le premier nœud qui qu'on va choisir pour construire la chaîne ce dernier doit être le plus loin du CH,les parametre d'appelle de cette procédure sont l'identificateur du cluster head ainsi que l'ensemble de nœuds qui appartiennent au cluster, le code d'implémentation est résumé dan la figure 31.

```
LEACH/LEACH-C instproc far_node {chid tdma} {

global ns_ opt tabdist
$self instvar farnode d1 d2
set tdmalength [llength $tdma]
#obtenir les cordonnées x,y du CH
set cord $tabdist($chid)
set XCH [lindex $cord 0]
set YCH [lindex $cord 1]

for {set i 0} {$i < $tdmalength} {incr i 1} {#boucle de recherche

set cord $tabdist([lindex $tdma $i])
set X1 [lindex $cord 0]
set Y1 [lindex $cord 1]
set d1 [dist $X1 $Y1 $XCH $YCH]

}
set farnode [lindex $tdma 0]
set cord $tabdist($farnode)
set X1 [lindex $cord 0]
set Y1 [lindex $cord 1]
for {set i 1} {$i < $tdmalength} {incr i 1} {

set cord $tabdist([lindex $tdma $i])
set X2 [lindex $cord 0]
set Y2 [lindex $cord 1]
```

```
set d1 [dist $X1 $Y1 $XCH $YCH]
set d2 [dist $X2 $Y2 $XCH $YCH]
if {$d2 > $d1} {
    set farnode [lindex $tdma $i]
    set cord $tabdist([lindex $tdma $i])
    set X1 [lindex $cord 0]
    set Y1 [lindex $cord 1]
} else {
    puts ".."
}

}

  return $farnode
}
```

Figure 31 : Code d'implémentation de la procédure du nœud le plus loin de la chaîne

3.2.3. Procédure de recherche du nœud le plus proche

La procédure de recherche du nœud le plus proche est invoquée par l'algorithme de formation de chaîne chaque fois qu'on veut déterminer le prochain nœud qui va appartenir à la chaîne. Evidemment, ce dernier doit être le plus proche de la tête de la chaîne. Les paramètres d'appelle de cette procédure sont la tête de la chaîne, l'identificateur du cluster head ainsi que l'ensemble de nœuds qui appartiennent au cluster. Le code d'implémentation est résumé dans la figure 32.

```
LEACH/LEACH-C instproc near_node {node tdma chid} {
global ns_ opt tabdist
$self instvar nearnode d1 d2
set tdmalength [llength $tdma]

if {$tdmalength > 0} {
#obtenir les cordonnées x,y de la tête de la chaîne
set cord $tabdist($node)
set XCH [lindex $cord 0]
set YCH [lindex $cord 1]

for {set i 0} {$i < $tdmalength} {incr i 1} {#boucle de recherche

set cord $tabdist([lindex $tdma $i])
set X1 [lindex $cord 0]
set Y1 [lindex $cord 1]
set d1 [dist $X1 $Y1 $XCH $YCH]

}
```

```
set nearnode [lindex $tdma 0]
set cord $tabdist($nearnode)
set X1 [lindex $cord 0]
set Y1 [lindex $cord 1]

for {set i 1} {$i < $tdmalength} {incr i 1} {

set cord $tabdist([lindex $tdma $i])
set X2 [lindex $cord 0]
set Y2 [lindex $cord 1]

set d1 [dist $X1 $Y1 $XCH $YCH]
set d2 [dist $X2 $Y2 $XCH $YCH]
if {$d2 < $d1} {
if {$d1>0} {
if {$d2>0} {

    set nearnode [lindex $tdma $i]
    set cord $tabdist([lindex $tdma $i])
    set X1 [lindex $cord 0]
    set Y1 [lindex $cord 1] }}
} else {
    puts ".."
}


}
  } else {
set nearnode $chid #si le cluster est vide retourner le noeud CH
}


  return $nearnode
}
```

Figure 32 : Code d'implémentation de la procédure du nœud le plus proche

3.2.4. Procédure de transmission de données

La procédure de transmission de données consiste à modifier la manière de transmission de données centralisées dans le protocole LEACH, dans lequel tous les nœuds appartenants au cluster transmettent à leur cluster head à une manière de transmission distribuée, où les données sont transmises à travers la chaîne de nœuds voisins jusqu'au cluster head

```
Application/LEACH instproc sendData {} {

  global ns_ opt DATA MAC_BROADCAST BYTES_ID tabchaine ss snd
  $self instvar next_change_time_ frame_time_ end_frm_time_
  $self instvar currentCH_ dist_ code_ alive_ tab distCH distN

  set nodeID [$self nodeID]
  set msg [list [list $nodeID , [$ns_ now]]]
  set spreading_factor $opt(spreading)
  set datasize [expr $spreading_factor * \
               [expr [expr $BYTES_ID * [llength $msg]] +
$opt(sig_size)]]#calculer la taille du segment de données à envoyer

  $self WakeUp #le noeud passe à l'état éveillé

  pp "$nodeID sending data $msg to $currentCH_ at [$ns_ now] (dist =
$dist_)"
#spécifier le type de protocole de transmission de la couche MAC
  set mac_dst $MAC_BROADCAST
  set next $currentCH_

catch {set infor $tabchaine($nodeID)}
#spécifier le noeud de destination pour lequel les données vont être
#transmises (le noeud voisin dans la chaîne)
catch {set next [lindex $infor 0]}
catch {set distCH [lindex $infor 2]}
catch {set distN [lindex $infor 3]}

set link_dst $next
#transmission des données
$self send $mac_dst $link_dst $DATA $msg $datasize $dist_ $code_

#transmission des données durant le slot de temps spécifié
  set xmitat [expr [$ns_ now] + $frame_time_]
  if {$alive_ && [expr $xmitat + $end_frm_time_] < \
                 [expr $next_change_time_ - 10 * $opt(ss_slot_time)]}
{
    $ns_ at $xmitat "$self sendData"
  }

if {$distN<$distCH} {
catch {set sense_energy [expr  $opt(Esense)* $opt(sig_size) *
8*$distN]}
} else {
catch {set sense_energy [expr  $opt(Esense)* $opt(sig_size) *
8*$distCH]}
}
  catch {pp "Node $nodeID removing sensing energy = $sense_energy J."}
# calcule de l'énergie dépensée durant la transmission des données
 catch {[$self getER] remove $sense_energy}

  if {$currentCH_ != $nodeID} {
    $self GoToSleep #passer à l'état endormi
  }

}
```

Figure 33 : Code d'implémentation de la procédure de transmission de données

3.2.5. Procédure de réception de données

La procédure de réception de données est ajoutée pour permettre aux nœuds de réceptionner les données transmises par leurs nœuds voisins dans la chaîne et de les agréger avec leurs propres données.

```
Application/LEACH instproc recvDATAnode {msg} {

  global ns_ opt
  $self instvar TDMAschedule_ receivedFrom_ dataReceived_ nodercv

  set nodercv [lindex $msg end]
  set nodeID [lindex $msg 0]
  pp "Node $nodercv received data ($msg) from $nodeID at [$ns_ now]"
  set receivedFrom_ [lappend receivedFrom_ $nodeID]

if {$nodercv != 100} {

set num_sigs 1
set compute_energy [bf $opt(sig_size) $num_sigs]
pp "\tcompute_energy = $compute_energy"
#calculer de l'énergie dépensée Durant la réception des données
    [$self getER] remove $compute_energy
    set receivedFrom_ [lappend receivedFrom_ $nodercv]
    set dataReceived_ $receivedFrom_
    set receivedFrom_ ""
}

}
```

Figure 34 : Code d'implémentation de la procédure de réception de données

3.2.6. Procédure de réception de données par le cluster head

Cette procédure permet au CH de réceptionner les données transmises par les têtes de chaînes afin de les agréger et les envoyer à la station de base.

```
Application/LEACH instproc recvDATA {msg} {

  global ns_ opt
  $self instvar TDMAschedule_ receivedFrom_ dataReceived_

  set chID [$self nodeID]
  set nodeID [lindex $msg 0]
  pp "CH $chID received data ($msg) from $nodeID at [$ns_ now]"
  set receivedFrom_ [lappend receivedFrom_ $nodeID]

  set last_node [expr [llength $TDMAschedule_] - 1]
  if {$chID == [lindex $TDMAschedule_ $last_node]} {
```

```
set last_node [expr $last_node - 1]
  }
  if {$nodeID == [lindex $TDMAschedule_ $last_node]} {
    pp "CH $chID must now perform comp and xmit to BS."
    set num_sigs 1
#[llength $TDMAschedule_]
    set compute_energy [bf $opt(sig_size) $num_sigs]
    pp "\tcompute_energy = $compute_energy"
#calculer l'énergie dépensée durant la réception des données
    [$self getER] remove $compute_energy
    set receivedFrom_ [lappend receivedFrom_ $chID]
    set dataReceived_ $receivedFrom_
    set receivedFrom_ ""

#envoyer les données reçues à la station de base
$self SendDataToBS

  }
}
```

Figure 35 : Code d'implémentation de la procédure de réception de données par le cluster head

4. Conclusion

L'implémentation de notre protocole hybride sous le simulateur de réseau NS2 va nous permettre d'évaluer les performances de notre protocole en terme de prolongement de temps de vie du réseau et de gestion économique d'énergie, en le comparant avec le protocole LEACH, LEACH-C et PEGASIS.

1. Introduction

La simulation de notre algorithme hybride constitue la plus importante étape de notre travail puisque on peut prouver les améliorations effectuées en terme d'économie d'énergie et de prolongement de temps de vie global du réseau à l'aide des résultats fournis. L'analyse des performances de notre algorithme de routage hybride est évaluée à l'aide du simulateur de réseau NS2 [12]. Les résultats fournis par la simulation sont comparés aux algorithmes LEACH, LEACH-C, et PEGASIS en terme de temps de vie du réseau.

2. Environnement de simulation

Dans cette simulation, notre modèle d'expérimentation est établi sur 100 nœuds répartis aléatoirement sur une surface carrée de 100 x 100 m² présentée par la figure 36.

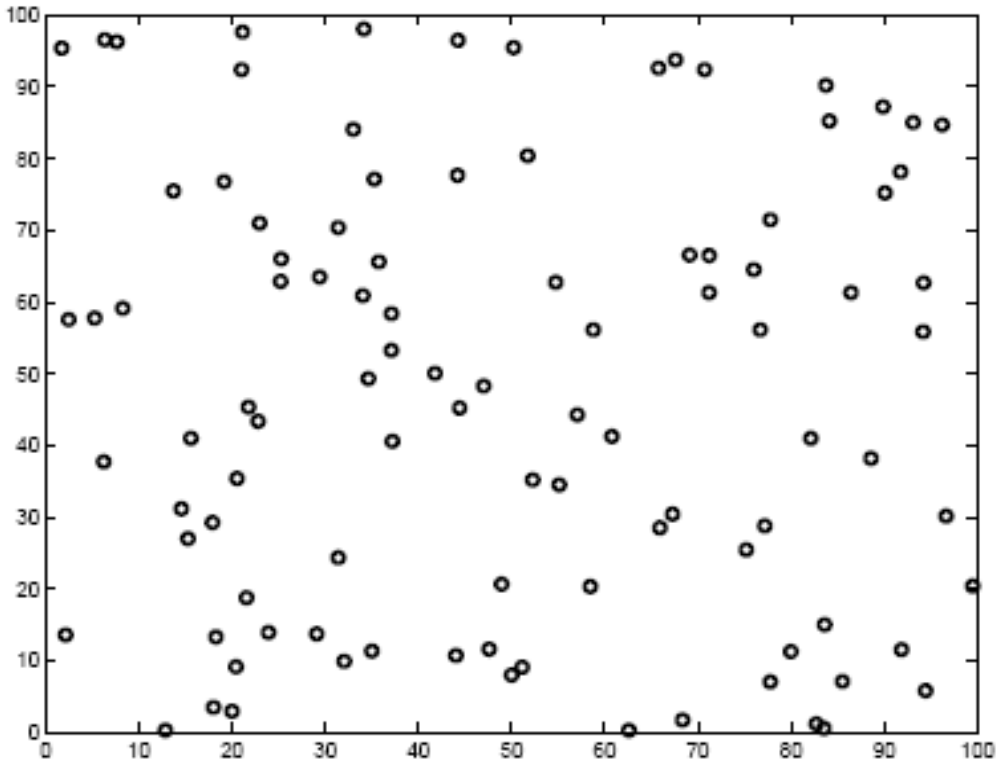


***Figure 36** : modèle d'expérimentation*

Nous assumons que tous les nœuds ont une position fixe durant toute la période de simulation. Notre modèle de simulation utilise les mêmes paramètres mentionnés dans [3], résumés dans le tableau ci-dessous :

Paramètre	***Valeur***
La surface du réseau	100 x 100 m²
La localisation de la BS	(50, 175)
Le nombre de noeuds	100
Le nombre de clusters	5
L'énergie initiale des noeuds	2 J
Taille du paquet de données	500 Byte
Eelecte	50nJ/bit
εfs	10nJ/bit/m²

Tableau 2 : Paramètres de simulation

La station de base est positionnée à 75 mètres par rapport au nœud le plus proche (X=50, Y=175), la largeur de bande de transmission est initialisée à 1Mbps, la latence de transmission et de réception d'un paquet de données est égale à 25µs, la taille d'un paquet de données est égale à 500 Bytes, avec une entête de paquet mesurant 25 Bytes.

Chaque nœud dans le réseau va consommer de l'énergie dans le cas où il va transmettre des paquets de données ainsi que dans le cas où il revoit des paquets de données sans oublier aussi le cas où il exécute des opérations de traitement de données (collecte et agrégation de données). Afin de calculer l'énergie dépensée pour chaque transaction d'envoi ou de réception de données, on a utilisé le modèle de dissipation d'énergie radio proposé dans [3] et présenté précédemment dans le chapitre 2 (états de l'art et motivation). L'énergie consommée pour transmettre q-bit de données sur une distance d à un nœud capteur est égale à :

$$\mathbf{Etx\ (q,d) = qEelect + q\varepsilon fsd^2.} \qquad (7)$$

L'énergie consommée pour recevoir q-bit de données est égale à :

$$\mathbf{Erx\ (q) = qEelect.} \qquad (8)$$

Où Eelect est l'énergie consommée par les circuits électroniques (énergie de calcul), εfs est l'énergie perdue dans l'espace de transmission.

Tous les nœuds du réseau commencent la simulation par énergie initiale égale à 2 J et une quantité de données illimitées à transmettre à la station de base. De plus, l'énergie de la station de base est considérée comme illimitée. Le temps de changement du cluster head ainsi que la reconstruction de grappe à chaînes est fixé à 20 s. Chaque nœud consomme sa réserve d'énergie limitée tout au long de la durée de simulation, ce qui implique l'épuisement de celle-ci. Ainsi, tout nœud qui a épuisé sa réserve d'énergie est considéré comme mort. Par conséquent, il ne peut ni transmettre ni recevoir des données.

3. Résultats de la simulation

Pour comparer la durée de vie du réseau entre les différents algorithmes existants (LEACH, LEACH-C, PEGASIS) et notre algorithme, nous avons mesuré l'énergie résiduelle des nœuds capteurs chaque 10 secondes tout au long de la durée de simulation, afin de calculer le nombre total des nœuds vivants. Nous avons réutilisé et reconfiguré, selon nos paramètres, les informations de simulation des protocoles LEACH et PEGASIS fournies dans [3] et [8] afin de les comparer avec le résultat de notre simulation, ce qui nous a donné comme résultat le graphe de la figure 37.

Le tableau présenté ci dessous résume les résultats de la simulation effectuée sur notre protocole hybride et celle des autres simulations effectuées sur les autres protocoles conquérant (LEACH, LEACH-C, et PEGASIS).

TEMPS DE SIMULATION (SECONDES)	***PROTOCOLE LEACH***	***PROTOCOLE LEACH-C***	***PROTOCOLE PEGASIS***	***NOTRE PROTOCOLE HYBRIDE***
	Nombre de nœuds vivants			
10s	100	100	100	100
30s	100	100	100	100
50s	100	100	100	100
70s	100	100	100	100
90s	100	100	100	100
110s	100	100	100	100
130s	100	100	100	100
150s	100	100	100	100
170s	100	97	100	100
190s	100	97	100	100
210s	99	95	100	99
230s	98	93	99	99
250s	95	88	97	97
270s	90	79	92	91
290s	87	72	87	89
310s	82	69	85	86
330s	76	60	82	83
350s	62	49	80	75
370s	52	47	77	69
390s	49	47	75	62
410s	42	36	66	61
430s	38	32	63	59
450s	32	30	60	56
470s	30	26	56	53
490s	24	22	54	50
510s	19	20	51	46
530s	15	19	47	39
550s	10	18	43	36

570s	10	14	30	28
590s	9	12	23	28
610s	6	11	20	25
630s	4	10	19	21
650s		9	19	18
670s		8	18	17
690s		7	17	15
710s		6	15	13
730s		4	14	12
750s			14	11
770s			12	10
790s			12	10
810s			12	9
830s			12	9
850s			11	9
870s			10	8
890s			10	8
910s			10	8
930 s			10	8
950 s			10	7
970s			9	7
990s			9	6
1010s			9	6
1030s			9	4
1050s			8	
1070s			8	
1090s			7	
1110s			6	
1130s			6	

Tableau 3 : Tableau comparatif des résultats de simulation.

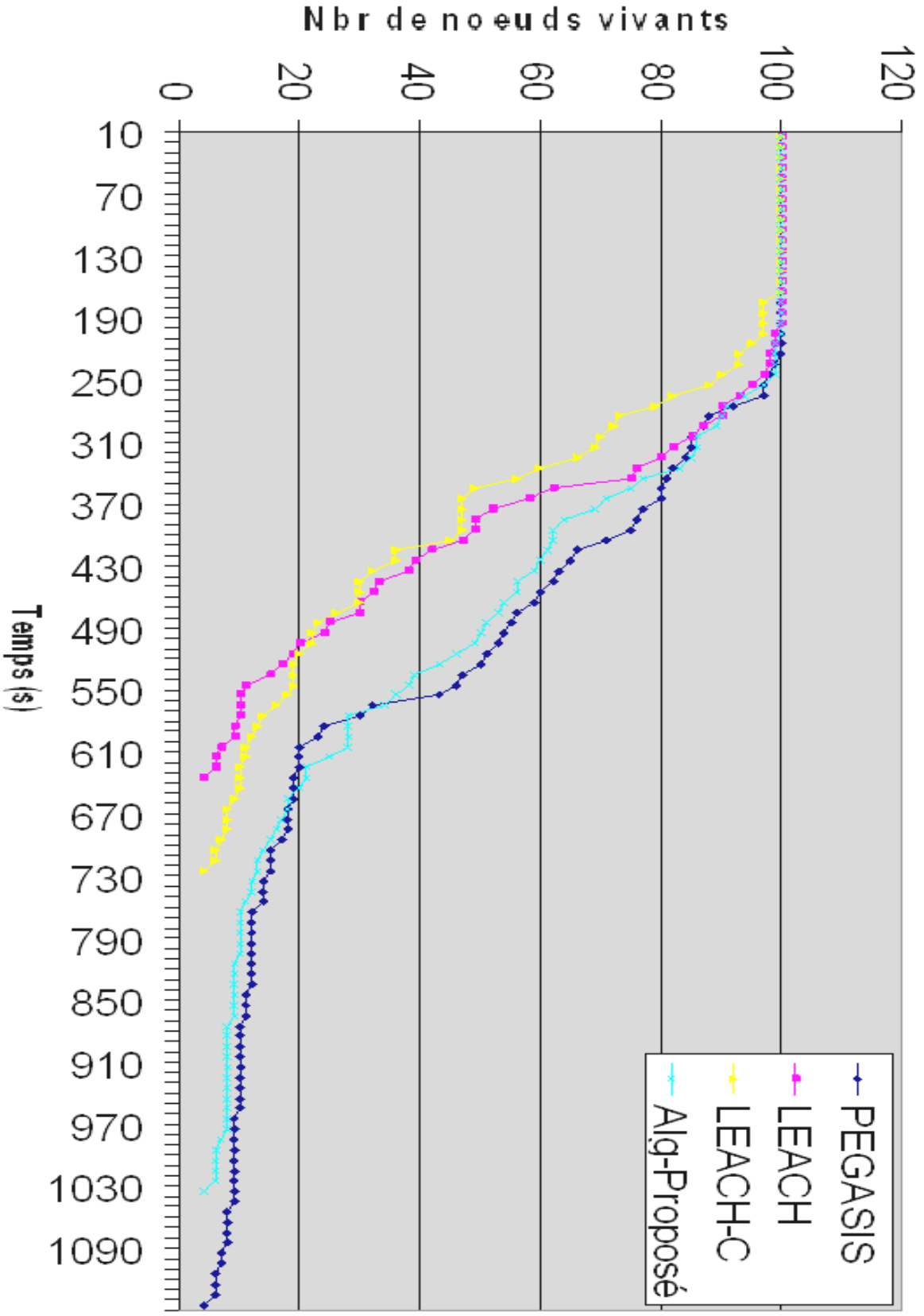


Figure 37. *Résultats de la simulation*

Pour bien approfondir l'évaluation des performances de notre algorithme hybride, nous avons mesuré le pourcentage de mortalité des nœuds tout au long de la durée de la simulation, ce qui nous a donné comme résultat le graphe de la figure.38 et la figure.39.

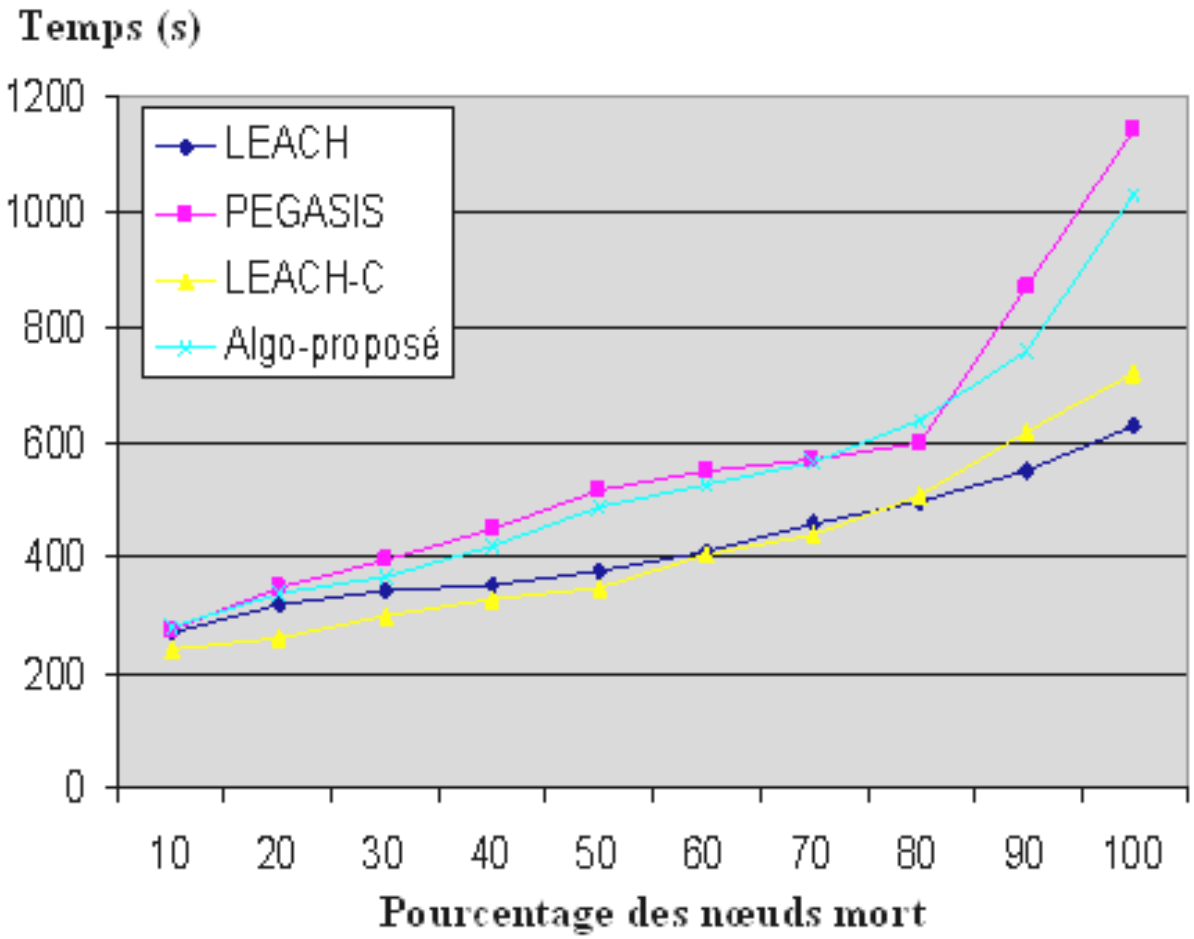


Figure. 38. Pourcentage de mortalité des noeuds dans le réseau

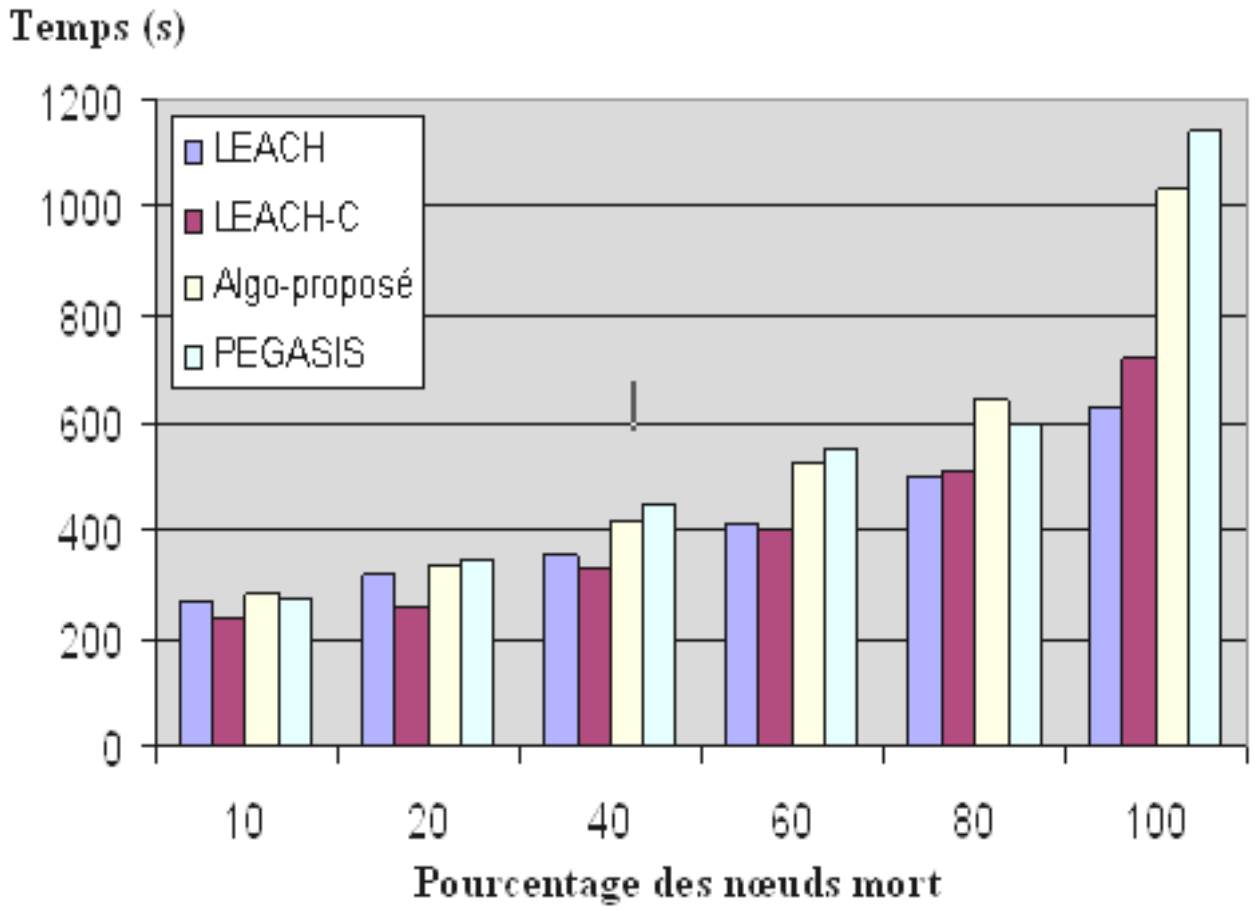


Figure. 39. Pourcentage de mortalité des noeuds dans le réseau

4. Conclusion

Basés sur les résultats de la simulation, nous avons démontré que notre protocole améliore la dissipation d'énergie à l'intérieur des clusters, augmente le gain d'énergie, et prolonge la durée de vie du réseau de 50% à 75% comparé au protocole LEACH, et de 45% à 55% comparé au protocole LEACH-C.

Reste à noter aussi que la durée de vie du réseau obtenue avec notre algorithme est très proche de celle fournie avec l'algorithme PEGASIS (de 7% à 15%). Mais, en combinant la métrique (Temps de vie X Latence), notre algorithme fourni le meilleur rapport, puisque il augmente le temps de vie du réseau comparé au protocole LEACH, et réduit considérablement l'extrême latence introduite par le protocole PEGASIS

Conclusion générale

Motivés par l'extrême latence introduite par la longue chaîne de nœuds dans le protocole PEGASIS ainsi que la mauvaise dissipation d'énergie dans le protocole LEACH, où les CHs et les nœuds éloignés de ces derniers meurent plus vite que les autres nœuds, nous avons essayé, de proposer un nouvel algorithme qui combine les avantages des deux protocoles dans le but de réduire leurs inconvénients et fournir un meilleur rapport durée de vie/latence.

Afin de valider les améliorations apportées par notre protocole en terme de prolongement de temps de vie du réseau ainsi que la gestion efficace de la consommation énergétique, nous avons simulé le fonctionnement de notre algorithme avec le simulateur NS2 version 2.30 et comparé les résultats fournis avec ceux du protocole LEACH, LEACH-C, et PEGASIS.

En effet, l'étape de simulation nous a posé un véritable défi puisque le code d'implémentation du protocole PEGASIS n'est pas disponible d'autant plus que celui du protocole LEACH est difficile à obtenir. De plus, le simulateur NS2 ne prend pas en considération les RCSFs. Donc, nous étions menés le reconfigurer manuellement afin de pouvoir l'utiliser.

Les résultats fournis par la simulation prouvent que notre protocole offre une meilleure gestion d'énergie par rapport aux protocoles LEACH, LEACH-C, et PEGASIS. De plus, le degré de latence engendré par la longue chaîne dans le protocole PEGASIS est significativement réduit.

Enfin, comme perspectives nous envisageons d'améliorer les performances de notre protocole de routage que ce soit au niveau de l'algorithme de construction de chaînes (prendre en considération les réserves d'énergie des nœuds dans la construction des chaînes), ou au niveau de l'algorithme d'élection et de formation des clusters (proposer d'autres algorithmes de formation de clusters et d'élection de CHs).

Liste des figures

1. Réseau de capteurs sans fil (WSN) …….. 4
2. Exemple de capteurs sans fil …… 5
3. Réseau de capteurs militaire ……..... 8
4. Applications à la sécurité ……. 8
5. Applications environnementales …….. 9
6. Ensemble de capteurs dans un corps humain ……..... 10
7. Modèle de consommation d'énergie pour la communication …… 13
8. Le protocole SPIN …….. 18
9. Phases du protocole de diffusion dirigée ……..... 18
10. Le routage hiérarchique …….. 20
11. Cluster-based approach (approche à grappe) …….. 21
12. Algorithme de routage LEACH …….. 22
13. Opérations de l'étape d'initialisation …….. 25
14. Construction de chaînes en utilisant l'algorithme avide …….. 28
15. Approche de dépassement de jeton …… 29
16. Topologie de 100 noeuds aléatoires pour un réseau 50m x 50m …… 31
17. Les résultats de performance pour un réseau de 50m x 50m …….. 33
18. Les résultats de performance pour un réseau de 100m x 100m …….. 33
19. Collecte de données dans une chaîne à arrangement binaire …….. 35
20. Des nœuds dans le réseau …… 39
21. Etapes de formation des grappes à chaînes …… 41
22. Algorithme de construction des chaînes …… 43
23. Organigramme de construction des chaînes …… 44
24. Division de la chaîne de nœuds …….. 45
25. Etapes d'exécution de notre protocole …… 46
26. Opérations de l'étape d'initialisation …….. 47
27. Approche de contrôle de transmission …… 49
28. Problèmes d'interférence entre les clusters …….. 50
29. Opérations de l'étape de transmission …… 50
30. Code d'implémentation de la procédure de formation des chaînes …… 57
31. Code d'implémentation de la procédure du nœud le plus loin de la chaîne ……... 59
32. Code d'implémentation de la procédure du nœud le plus proche …….. 60
33. Code d'implémentation de la procédure de transmission de données …….…… 61
34. Code d'implémentation de la procédure de réception de données …… 62
35. Code d'implémentation de la procédure de réception de données par le CH ……... 63
36. Modèle d'expérimentation …….. ..64
37. Résultats de la simulation ……..... 66
38. Pourcentage de mortalité des noeuds dans le réseau ……..... 70
39. Pourcentage de mortalité des noeuds dans le réseau …….. 71

Glossaire des acronymes et des notations

- **RCSF** : Réseau de capteurs sans fil
- **WSN** : Wireless sensors networks
- **LEACH** : Low Energy Adaptive Clustering Hierarchy
- **LEACH-C** : LEACH-Centralisé
- **LEACH-F** : LEACH avec grappes fixes
- **PEGASIS** : Power-Efficient Gathering in Sensor Information Systems
- **Power-aware** : Consommation d'énergie minime
- **SPIN**: Protocoles de capteurs pour l'information par négociation
- **GPS** : système de positionnement global
- **MECN** : Minimum energy communication network
- **SMECN** : Small MECN
- **GAF** : Geographic adaptive fidelity
- **GEAR** : Geographic and energy-aware routing
- **CH** : Cluster Head
- **BS** : Base station
- **TDMA** : Time division multiplexed access
- **CDMA** : Code division multiple access
- **DS-SS** : Direct-sequence spread spectrum
- **TCL** : Tool Command Language
- **NS** : Network simulator

- **Eelec** : Energie de transmission/réception électronique ;
- **k** : Taille d'un message ;
- **d** : Distance entre l'émetteur et le récepteur ;
- **ETx_amp** : Energie d'amplification;
- **ɛamp** : Facteur d'amplification;
- **dcrossover** : Distance limite pour laquelle les facteurs de transmission changent de valeur.

- ***P*** : Pourcentage désiré de CHs.
- ***r*** : Itération actuelle.
- ***G*** : Ensemble des noeuds qui ont été sélectionnés comme CH durant les dernières (1/ *P*) itérations.
- **K** : La constante de Bolztmann

Références

[1] F. Akyildiz, Weilian Su, Sankarasubramaniam, E. Cayirci, “A survey on sensor networks”, *IEEE Communications*, Aug 2002.

[2] C. Schurgers and M.B. Srivastava, “Energy efficient routing in wireless sensor networks”, *MILCOM Proceedings on Communications for Network-Centric Operations*, 2001

[3] W..R. Heinzelman, A. Chandrakasan, and H.Balakrishnan, “An Application-Specific Protocol Architecture for Wireless Micro sensor Networks”, *IEEE Transactions on the wireless communications*, Vol. 1, No. 4, pp. 660-670, Oct. 2002.

[4] W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan “Energy-efficient communication protocol for wireless micro sensor networks”, *IEEE Hawaii International Conference on System Sciences*, 2000.

[5] M. Younis, M. Youssef and K. Arisha, “Energy-aware Routing in Cluster-Based Sensor Networks”, *the Proceedings of the 10th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems(MASCOTS2002)*, October 2002.

[6] Jamal N. Al-Karaki, Raza Ul-Mustafa, Ahmed E. Kamal, “Data Aggregation in Wireless Sensor Networks - Exact and Approximate Algorithms”, *Proceedings of IEEE Workshop on High Performance Switching and Routing (HPSR)*, pp 18-21, Phoenix Arizona, USA. April 2004.

[7] S. Lindsey, S. Raghavendra, “Data Gathering Algorithms in Sensor Networks Using Energy Metrics”, *IEEE Transactions on parallel and distributed systems*, Vol.13, No.9, 2002.

[8] S. Lindsey, C. Raghavendra, “PEGASIS: Power-Efficient Gathering in Sensor Information Systems", *IEEE Aerospace Conference Proceedings*, 2002, Vol. 3, 9-16 pp. 1125-1130.

[9] S. Bandyopadhyay, E. Coyle, ”An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks", *Proceedings of IEEE INFOCOM*, Vol. 3, pp. 1713-1723. 2003

[10] T. Thua Huynh et C. Seon Hong “An Energy*Delay Efficient Routing Scheme for Wireless Sensor Networks”, *MMNS 2005, LNCS 3754*, pp. 11–22, 2005. IFIP International Federation for Information Processing 2005

[11] P. Agarwal and C. Procopiuc. Exact and Approximation Algorithms for Clustering. *In Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 658-667, January 1999

[12] Information Sciences Institute, "The Network Simulator ns-2" http://www.isi.edu/nanam/ns/, University of Southern California

[13] G. Lee, J. Kong, and O. Byeon; "Cluster based Energy Aware Routing Protocol for Sensor Networks", From Proceeding (527) Networks and Communication Systems – 2006.

[14] Handy, M., Haase, M., Timmermann, "Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection," *4th IEEE International Conference on Mobile and Wireless Communications Networks*, Stockholm, 2002

[15] Intanagonwiwat, C., Govindan, R., Estrin, D., "Directed Diffusion: a scalable and robust communication paradigm for sensor networks", ACM Press, 2000

[16] Y.Wang, C.Hsiao Tsai and H.Mao, 'HMRP: Hierarchy-Based Multipath Routing Protocol for Wireless Sensor Networks', *Tamkang Journal of Science and Engineering*, Vol. 9, No 3, pp. 255-264 (2006)

[17] Callaway, Edgar H., 'Wireless Sensor Networks: Architectures and Protocols', *Auerbach Publications* (2003).

[18] K.Kalpakis, K.Dasgupta, and P.Namjoshi, "Efficient Algorithms for Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," *Computer Networks: The International Journal of Computer and Telecommunications Networking,* Vol. 42, pp. 697_716 (2003).

[19] Al-Karaki, J. N. and Kamal, A. E., "Routing Techniques in Wireless Sensor Networks: A Survey," *IEEE Wireless Communications,* Vol. 11, pp. 6_28 (2004).

[20] E.Royer, and C.Toh, "A Review of Current Routing Protocols for ad hoc Mobile Wireless Networks," *IEEE Personal Communication,* Vol. 6, pp. 46_55 (1999).

[21] H.Özgür Tan and I.Körpeoglu, "Power Efficient Data Gathering and Aggregation in Wireless Sensor Networks," *Proc. ACM Conf. on SIGMOD*, San Diego, CA, Vol. 32, pp. 66_71 (2003).

[22] S.D.Muruganathan, D.C.F.Ma, Bhasin, R. I. and Fapojuwo, "A Centralized Energy-Efficient Routing Protocol for Wireless Sensor Networks," *IEEE Communication Magazine,*Vol. 43, pp. S8_13 (2005).

[23] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors", *Communications of the ACM*, Vol. 43, No. 5, pp 51-58, May 2000.

[24] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks,",*IEEE Transactions on Information Theory*, vol. IT-46, no. 2, pp. 388-404, March 2000.

[25] Ioan Raicu "Routing Algorithms for Wireless Sensor Networks", *ACM MOBICOM 2002*

[26] C. Chong and S. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proc. IEEE*, vol. 91, no. 8, Aug. 2003.

[27] A.A. Ahmed, H. Shi, and Y. Shang, “A Survey on Network Protocols for Wireless Sensor Networks,” *Proc. IEEE Int'l Conf. Information Technology: Research and Education (ITRE '03)*, Aug. 2003.

[28] B. Krishnamachari, D. Estrin, and S. Wicker, “Impact of Data Aggregation in Wireless Sensor Networks,” *Proc. 22nd Int'l Conf. Distributed Computing Systems*, July 2002.

[29] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, “Directed Diffusion for Wireless Sensor Networking,” *IEEE/ACM Trans. Networking*, vol. 11, no. 1, Feb. 2003.

[30] K. Du, J. Wu and D. Zhou, “ Chain-based protocols for data broadcasting and gathering in sensor networks,” *International Parallel and Distributed Processing Symposium*, April 2003.

[31] S.Lee, J.Yoo, T.Choong “Distance-Based Energy Efficient Clustering for Wireless Sensor Networks”, *29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, pp. 567-568, 2004.

[32] K. Akkaya and M. Younis, “A survey of routing protocols in wireless sensor networks”, *Elsevier Ad Hoc Network Journal*, vol. 3, no. 3, pp. 325–349, May 2005.

[33] L.Pierron “GUI programmation en Tcl”, mars 2002.

[34] B.Welch “Tcl Fundamentals”, http://www.beedub.com/book/, 1999.

[35] D.Boubiche, A.Bilami “ Une approche hybride dans le routage pour une économie d’énergie dans les R.C.S.F.”, *6ème Séminaire National en Informatique à Biskra (SNIB08)*, pp 125-130. 3 Mai 2008.

[36] A.Bilami, D.Boubiche “A hybrid Energy Aware Routing Algorithm for Wireless Sensor Networks ”, *The Thirteenth IEEE Symposium on Computers and Communications (ISCC’08)*, pp 975-980. 6 juillet 2008.

[37] D.Boubiche, A.Bilami “ Protocole de Routage à Base de Clusters à Chaînes Statiques dans les R.C.S.F. ”, *1stWorkshop on Next Generation Networks: Mobility (IEEE WNGN 2008)* ,18 juillet 2008.

Site Web

[1] http://www.internetworkflow.com/downloads/ns2leach/mit.tar.gz, (package d’installation du protocole LEACH)