الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي
Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

FACULTE DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE ET ELECTRONIQUE (GEE)

Mémoire

Pour l'obtention du diplôme de

MASTER en Télécommunication

Option : Réseaux Mobile et Services **(RMS)**

Présenté par
Melle Boumediene Siham
Melle Maharrar Nadia

THEME

Optimisation de la diffusion dans les Réseaux de capteurs

Soutenu en juin 2013 devant un jury composé de

Mr R. Bouabdallah	Président	MA. A à l'université de Tlemcen
Mr R. Merzougui	Examinateur	MC. B à l'université de Tlemcen
Mr B. Kadri	Examinateur	MC. B à l'université de Tlemcen
Mr D. Moussaoui	Encadreur	MA. A à l'université de Tlemcen

Remerciements

Nous remercions tout d'abord « ALLAH » qui nous à donnée la force et le courage a fin de parvenir à élaborer ce modeste travail.

Nous tenons à remercier en premier lieu Mr. Djillali Moussaoui notre encadreur de mémoire pour son sympathie, son disponibilité, son idées, son conseils et son encouragements qui nous ont permis de mener à bien ce mémoire.

Nous adressons aussi nos très sincères remerciements à Mr. Réda Bouabdallah de nous faire l'honneur de s'intéresser à ce travail et d'avoir présidé le jury.

Nous exprimons ensuite notre plus profonde gratitude à Mr. Benamar Kadri et Mr. Rachid Merzougui qui ont accepté d'examiner.

Nous exprimons également notre gratitude à tous les professeurs et enseignants qui ont collaboré à notre formation depuis notre premier cycle d'étude jusqu'à la fin de notre cycle universitaire.

Un merci pudique à toute notre famille et nos amis pour son soutien.

Dédicace

Je rends grâce à dieu de m'avoir donné le courage et la volonté ainsi que la conscience pour venir à terme de mes études.

Je dédie ce modeste travail :

A mes très chers parents qui étaient présents pour moi durant toute ma vie.

A mes frères et mes sœurs.

A toute ma grande famille.

A tous les professeurs et enseignants qui ont collaboré à ma formation depuis mon premier cycle d'étude jusqu'à la fin de mes études universitaires.

A tous ceux qui m'ont aidé de près ou de loin durant mes études.

MAHARRAR NADIA

Dédicace

Je rends grâce à dieu de m'avoir donner le courage et la volonté ainsi que la conscience d'avoir pu terminer mes études.

Je dédie ce modeste travail :

A ma très chère mère et Grand mère pour toute sa tendresse et pour ses nombreux sacrifices. Que Dieu les garde.

A mes très cher Oncles et tantes pour l'encouragement et affection qu'ils m'ont prodigués durant mes études. Que Dieu me lui garde.

A mes sœurs et mes frères.

A tous mes amis que J'aime énormément.

A tous ceux qui m'ont aidé de près ou de loin durant mes études.

BOUMEDIENE SIHAM

SOMMAIRE

Chapitre II Protocoles de routage dans les RCSFs

L'étude des réseaux de capteurs sans fil a connu une récente envolée, en raison des progrès technologiques réalisés dans les domaines de la microélectronique et des systèmes embarqués. Composé de nombreuses entités autonomes, appelées nœuds-capteurs, un tel réseau est utilisé pour collecter des informations sur l'environnement dans lequel il est déployé. Dans un scénario de déploiement typique, il est admis que ces informations sont ensuite acheminées vers une station de base afin d'y être traitées.

Un nœud-capteur est généralement un petit appareil de taille limitée contenant un ou plusieurs capteurs, un processeur, une mémoire, une interface de communication et une alimentation électrique. Une large variété de capteurs mécaniques, thermiques, biologiques, optiques et magnétiques peut être attachée au nœud-capteur pour mesurer les propriétés de son environnement. En raison de sa faible capacité de stockage et de ses conditions de déploiement, dans des zones souvent difficiles d'accès, chaque nœud dispose d'une interface radio lui permettant de transmettre ses données à une station de base, aux ressources moins limitées, appelée puits.

Ces avancées technologiques rendent possible le déploiement de réseaux de capteurs sans fil. En effet, les domaines d'application des réseaux de capteurs sont très variés : applications militaires, surveillance, environnementales, médicales domestiques, commerciales.

Le problème qui pose, c'est que chaque nœud diffuse le paquet a tout ces voisins ce qui va créer une surcharge de la bande passante « inondation aveugle » et va aussi gaspiller énormément d'énergie au niveau des nœuds vu que tous ces derniers participeront a cette inondation jusqu'à atteindre la destination désirée, ce qui risque de les faire disparaitre du réseau au cas où leurs batteries s'éteindront.

Dans le premier chapitre, nous verrons les concepts généraux relatifs au domaine des réseaux de capteurs sans fil, dans le deuxième chapitre, nous ferrons une étude bibliographique sur les protocoles de routage proposés pour ce genre de réseaux et dans le troisième chapitre on parlera de la diffusion, et on introduira la méthode CBEF comme méthode d'optimisé la diffusion et dans le dernière chapitre on présentera l'environnement de développement, aussi l'implémentation du programme (capteur et station de base).

I.1 Introduction

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

Les réseaux de capteurs se composent généralement d'un grand nombre de capteurs communicants entre eux via des liens radio pour le partage d'information et le traitement coopératif. Dans ce type de réseau, les capteurs échangent des informations par exemple sur l'environnement pour construire une vue globale de la région contrôlée, qui est rendue accessible à l'utilisateur externe par un ou plusieurs nœud(s). Les données collectées par ces capteurs sont acheminées directement ou via les autres capteurs de proche en proche à un « point de collecte », appelé station de base (ou SINK s'il s'agit d'un nœud). Cette dernière peut être connectée à une machine puissante via internet ou par satellite. En outre, l'utilisateur peut adresser ses requêtes aux capteurs en précisant l'information d'intérêt.

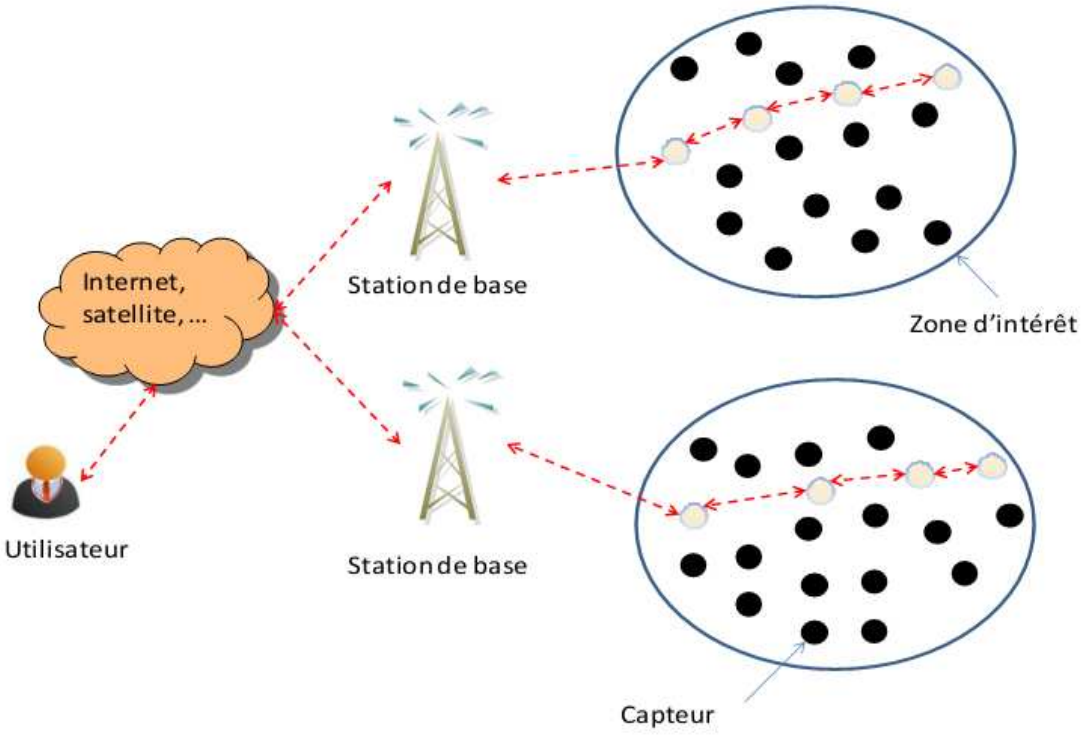

Figure I.1 Exemple de réseau de capteurs

Un exemple de réseaux de capteurs est fourni dans la **Figure I.1**: les capteurs sont déployés d'une manière aléatoire dans une zone d'intérêt, et une station de base, située à l'extrémité de cette zone, est chargée de récupérer les données collectées par les capteurs. Lorsqu'un capteur détecte un événement pertinent, un message d'alerte est envoyé à la station de base par le biais d'une communication entre les capteurs. Les données collectées sont traitées et analysées par des machines puissantes.

Les réseaux de capteurs viennent en soutien de l'environnement et de l'industrie grâce aux récents développements réalisés dans le domaine des techniques sans fils. Depuis quelques décennies, le besoin d'observer et de contrôler des phénomènes physiques tels que la température, la pression ou encore la luminosité est essentiel pour de nombreuses applications industrielles et scientifiques.

I.2 Réseau de capteurs sans fils

Depuis quelques décennies, le besoin d'observer et de contrôler des environnements hostiles est devenu essentiel pour de nombreuses applications militaires et scientifiques. Les nœuds utilisés doivent être autonomes, d'une taille miniature et peuvent être déployés d'une manière dense et aléatoire dans le champ surveillé. Une classe spéciale des réseaux Ad Hoc appelée réseaux de capteurs sans fil vient au secours. Ceux-ci sont apparus grâce aux développements technologiques tels que la miniaturisation des composants électronique, la diminution des coûts de fabrication et l'augmentation des performances et des capacités de stockage, d'énergie et de calcul. [1]

I.2.1 Définitions

a) Un capteur :

Un capteur est un mini-composant, qui permet d'acquérir des données sur son environnement, les traiter et les communiquer. Son intégration est une tâche difficile à réaliser en tenant compte de certaines contraintes : l'espace mémoire, la consommation énergétique, etc. [2]

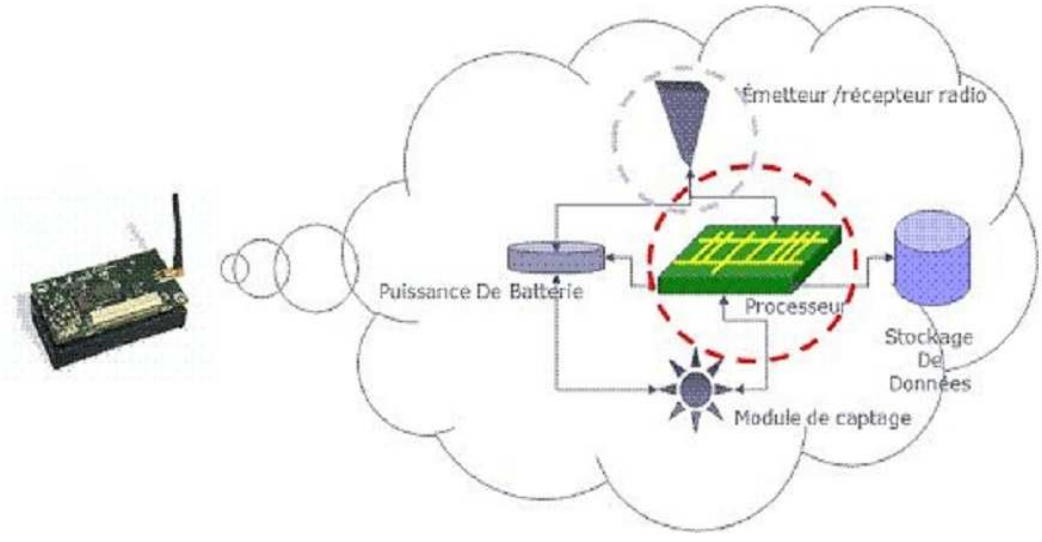

Figure I.2 Anatomie d'un nœud capteur. [3]

b) Un réseau de capteur :

Un réseau de capteur sans fil (RCSF ou WSN: Wireless Sensor Network) est un ensemble de nœuds capteurs dont le nombre varie de quelques dizaines d'éléments à plusieurs milliers voire des millions. Les capteurs ne sont pas intégrés à une quelconque architecture de communication préexistante, mais ils communiquent à l'aide d'un réseau Ad Hoc sans fil. L'alimentation électrique de chaque capteur est assurée par une batterie individuelle dont la consommation pour la communication et le calcul lié au traitement de l'information doivent être optimisés.

Le déploiement des nœuds est soit aléatoire (avion, missile) ou déterministe (manuelle, robots). Le déploiement aléatoire rend difficile la détermination des nœuds voisins. Par contre, le déploiement déterministe deviendra difficile si le nombre de nœuds est très élevé.

Le déploiement des nœuds est aussi dense, mais, il serait impossible pour plusieurs utilisateurs si la mise en place du réseau en entier se révélait très coûteuse. C'est pourquoi des recherches visent à concevoir des nœuds à prix bas (<0.5 dollars) tout en intégrant plusieurs composants sur la même interface (capteur, mémoire, processeur, etc.). [1]

I.3 Architecture physique d'un capteur

Pour bien comprendre le fonctionnement d'un réseau de capteur, il est important de voir l'architecture interne d'un capteur.

La structure d'un capteur est représentée par le schéma de la **Figure I.3**. Un nœud capteur peut être décomposé en quatre sous-systèmes principaux. Il s'agit de l'unité de capture, l'unité de traitement, l'unité de communication et l'unité de contrôle d'énergie.

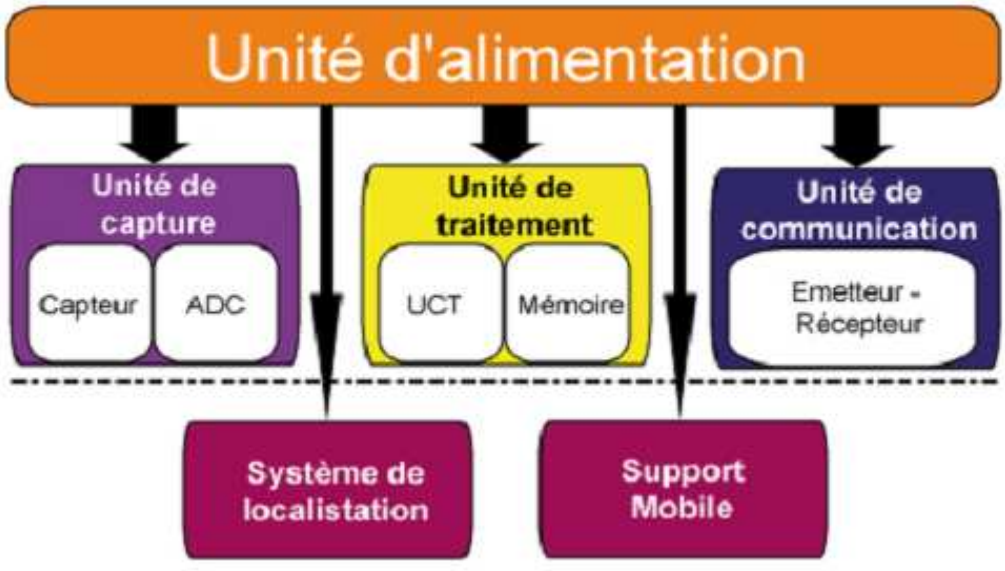

Figure I.3 Composant d'un capteur. [4]

I.3.1 Unité de capture

L'unité de capture est composée d'un capteur qui va obtenir des mesures numériques sur les paramètres environnementaux et d'un convertisseur Analogique/Numérique qui va convertir l'information relevée et la transmettre à l'unité de traitement.

I.3.2 Unité de traitement

L'unité de traitement est composée de deux interfaces, une interface avec l'unité de capture et une interface avec l'unité de transmission. Cette unité est également composée d'un processeur et d'un système d'exploitation spécifique. Elle acquiert les informations en provenance de l'unité de capture et les envoie à l'unité de transmission.

I.3.3 Unité de communication

L'unité de transmission est responsable de toutes les émissions de réceptions de données via un support de communication radio.

I.3.4 Unité 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 généralement irremplaçable. Dès lors, L'énergie est la ressource la plus précieuse dans un réseau de capteurs, puisqu'elle influe directement sur la durée de vie des micro-capteurs et du réseau en entier. L'unité de contrôle d'énergie constitue donc l'un des systèmes les plus importants.

Il se peut que le capteur contienne des composant additionnels, et ce, selon son domaine d'application. [4]

I.4 Caractéristiques principales d'un capteur

Deux entités sont fondamentales dans le fonctionnement d'un capteur : l'unité de capture qui est le cœur physique permettant la prise de mesure et l'unité de communication qui réalise la transmission de celle-ci vers d'autres dispositifs électroniques. Ainsi, fonctionnellement chaque capteur possède un rayon de communication (Rc) et un rayon de sensation (Rs).

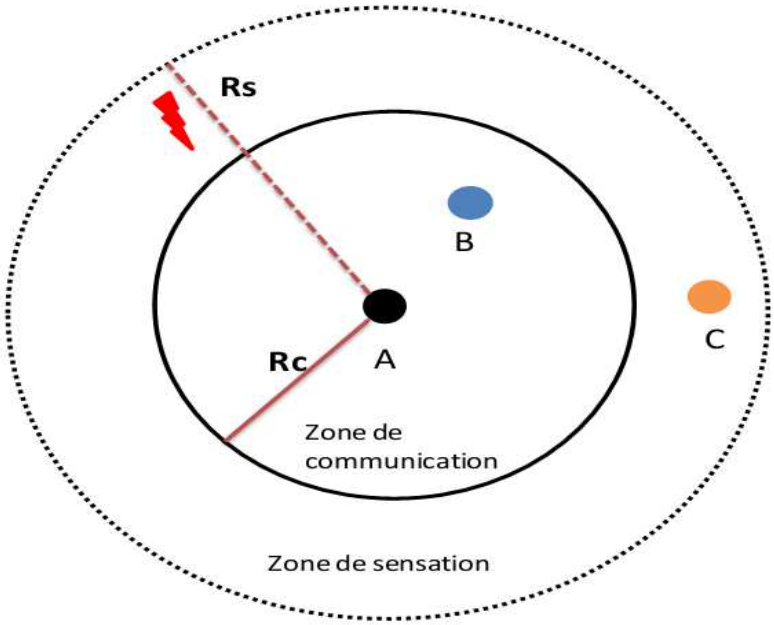

Figure1.4 Rayons de communication et de sensation d'un capteur

La **Figure I.4** montre les zones définies par ces deux rayons pour le capteur **A**. La zone de communication est la zone où le capteur **A** peut communiquer avec les autres capteurs (le capteur **B**). D'autre part, la zone de sensation est la zone où le capteur **A** peut capter l'événement [5], [6].

I.5 Caractéristiques des réseaux de capteurs sans fil

Un réseau de capteurs présente les caractéristiques suivantes:

- **Absence d'infrastructure** : les réseaux Ad-hoc en général, et les réseaux de capteurs en particulier se distinguent des autres réseaux par la propriété d'absence d'infrastructure préexistante et de tout genre d'administration centralisée.
- **Taille importante** : un réseau de capteurs peut contenir des milliers de nœuds.
- **Interférences** : les liens radio ne sont pas isolés, deux transmissions simultanées sur une même fréquence, ou utilisant des fréquences proches, peuvent interférer.
- **Topologie dynamique** : les capteurs peuvent être attachés à des objets mobiles qui se déplacent d'une façon libre et arbitraire rendant ainsi la topologie du réseau fréquemment changeante.
- **Sécurité physique limitée** : les réseaux de capteurs sans fil sont plus touchés par le paramètre de sécurité que les réseaux filaires classiques. Cela se justifie par les contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé.
- **Bande passante limitée** : une des caractéristiques primordiales des réseaux basés sur la communication sans fil est l'utilisation d'un médium de communication partagé. Ce partage fait que la bande passante réservée à un nœud est limitée.
- **Contrainte d'énergie, de stockage et de calcul** : la caractéristique la plus critique dans les réseaux de capteurs est la modestie de ses ressources énergétiques car chaque capteur du réseau possède de faibles ressources en termes d'énergie (batterie). Afin de prolonger la durée de vie du réseau, une minimisation des dépenses énergétiques est exigée chez chaque nœud. Ainsi, la capacité de stockage et la puissance de calcul sont limitées dans un capteur.

I.6 Architecture

Il existe deux types d'architectures pour les réseaux de capteurs sans-fil [7] :

I.6.1 Architecture plate ou horizontale

Dans ce type de configuration, tous les nœuds ont un même niveau et peuvent communiquer avec tous les autres nœuds (**Figure I.5**).

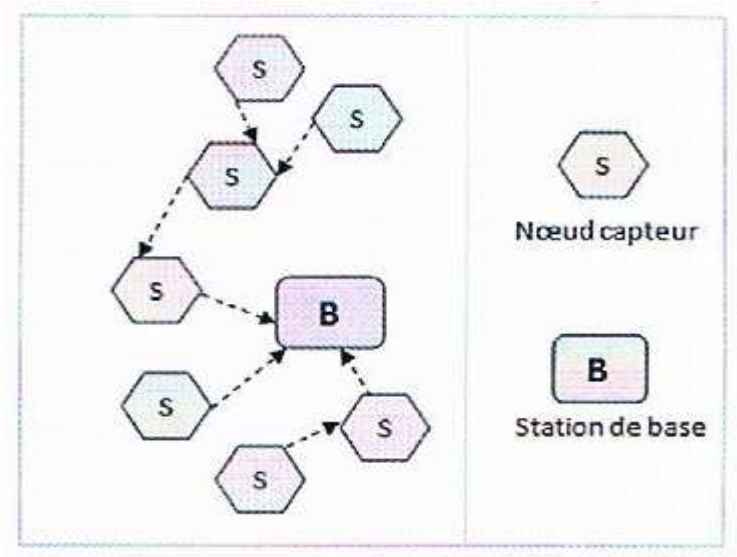

Figure I.5 Architecture plate [8]

On peut distinguer deux schémas :

Centralisé : dans lequel, toutes les données capturées par les nœuds capteurs sont envoyées vers un nœud central qui fait le traitement et la fusion des données pour les transmettre à la station de base. Ce schéma est très simple et il est utilisé seulement pour des réseaux de petite densité.

Distribué : il est plus compliqué. Plusieurs nœuds de traitement de données existent et peuvent communiquer entre eux. Un groupe de nœuds récoltent chacun de leur côté les données du réseau et les communiquent entre eux jusqu'à acheminement à la station de base.

I.6.2 Architecture hiérarchique

Une architecture hiérarchique a été proposée pour réduire la complexité de la plupart des nœuds capteurs et leur déploiement, en introduisant un ensemble de nœuds capteurs plus puissants. L'architecture hiérarchique est composée de plusieurs couches : une couche de capteurs, une couche de transmission et une couche de point d'accès.

Donc, le réseau est découpé en clusters, dans chaque cluster un Cluster-Head est élu pour gérer les communications inter et intra cluster. Toutes les données reçues d'un niveau inférieur sont traitées et agrégées par les cluster-heads de ce niveau avant d'être transmises vers le niveau supérieur.

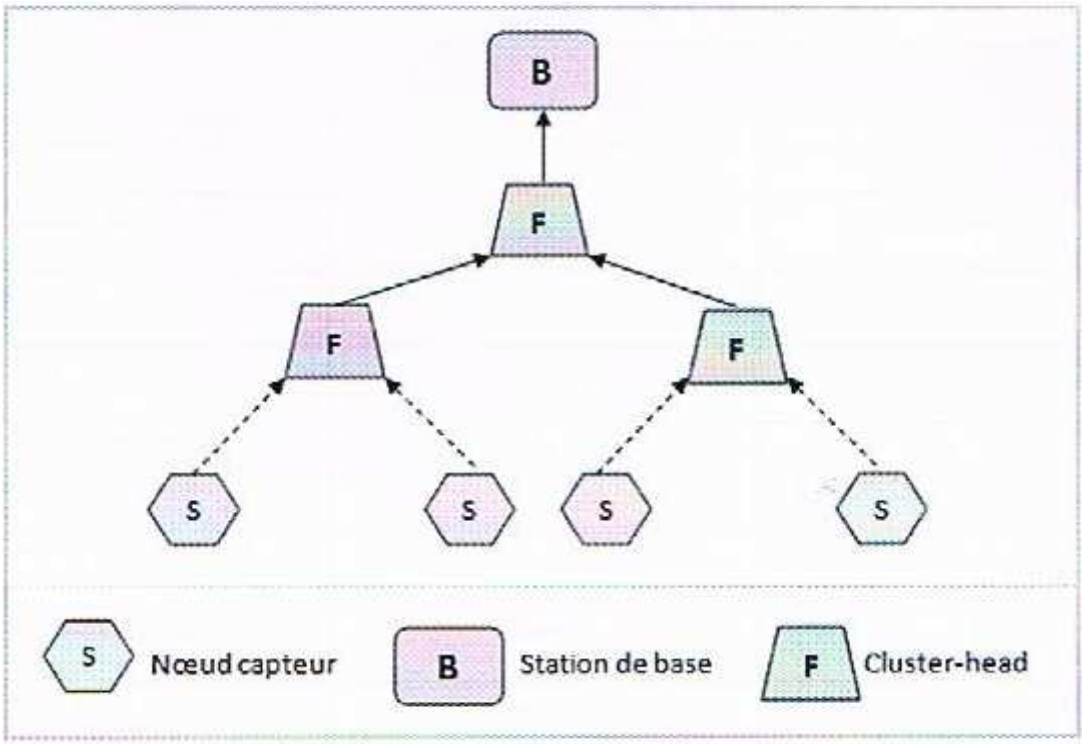

Figure I.6 Architecture Hiérarchique [8]

I.7 Technologies radio des réseaux de capteurs (protocoles) [9]

Parmi les technologies radio utilisées pour permettre les communications entre capteurs, nous pouvons citer : Bluetooth (IEEE802.15.1), UWB (ou IEEE 802.15.3) et Zigbee (IEEE 802.15.4)…

I.7.1 Bluetooth

La technologie Bluetooth, dont Ericsson a initié le projet en 1994, a pour but principal de remplacer les câbles sur de petites distances. Elle est utilisée dans la plupart des téléphones portables comme interface de connexion pour accéder à un PC. Malheureusement, le grand défaut de cette technologie est sa trop grande consommation d'énergie. Elle ne peut donc pas être utilisée par des capteurs qui sont alimentés par batterie et qui, idéalement, devraient fonctionner durant plusieurs années. [10]

I.7.2 Zigbee

Le standard Zigbee, combiné avec IEEE 802.15.4 offre des caractéristiques qui répondent encore mieux aux besoins des réseaux de capteurs. Zigbee offre des débits de données moindres, mais il consomme également nettement moins que Bluetooth. Un petit débit de données (250kbit/s max) n'est pas handicapant pour un réseau de capteurs où les fréquences de transmission ne sont pas soutenues et conséquents. [10]

I.7.3 UWB (Ultra Wide Band)

L'UWB peut être utilisé en tant que technique de communication sans fil, qui fournit des taux de transfert réseaux très élevés sur des distances relativement courtes et à faible puissance. De plus, l'atténuation du signal engendré par des obstacles est moindre qu'avec les systèmes radio à bande étroite conventionnels. [11]

I.7.4 Infrarouge

Les ondes infrarouges sont utilisées pour la communication à courte distance, ainsi elles s'adaptent au mode de communication entre les nœuds d'un réseau de capteurs [12]. Les ondes infrarouges sont préférées aux ondes radio, car elles n'interfèrent pas avec les autres signaux électromagnétiques.

Les émetteurs-récepteurs utilisant la technologie infrarouge ne sont pas chers, et ils sont faciles à fabriquer [12]. Les composants matériels utilisant ce type de communication sont généralement conformes aux standards publiés par l'Infrared Data Association(IrDA).

Enfin, le tableau ci-dessous récapitule la différence entre les technologies Zigbee, Bluetooth et Wifi.

Protocole	Zigbee	Bluetooth	Wifi
IEEE	802.15.4	802.15.1	802.11a/b/g
Besoin mémoire	4-32 Ko	250 Ko	1 Mo
Energie nécessaire	Faible	Moyenne	Elevée
Nombre de nœuds	Non limité	7	32
Vitesse de transfert	250 KB/s	720 Kb/s	11-54-108Mb/s
Portée	100m	1-100m	300m
Temps de démarrage	Court	Moyen	Long

Tableau I.1 Technologies sans fil et leurs caractéristiques

I.8 Couches de la pile protocolaire

Dans le but d'un établissement efficace d'un RCSF, une architecture en couche est adoptée afin d'améliorer la robustesse du réseau. Une pile protocolaire de cinq couches est donc utilisée par les nœuds du réseau. Citons la couche application, la couche transport, la couche réseau, la couche liaison de données et la couche physique.

De plus, cette pile possède trois plans (niveaux) de gestion : le plan de gestion des taches qui permet de bien affecter les taches aux nœuds capteurs, le plan de gestion de mobilité qui permet de garder une image sur la localisation des nœuds pendant la phase de routage, et le plan de gestion de l'énergie qui permet de conserver le maximum d'énergie.

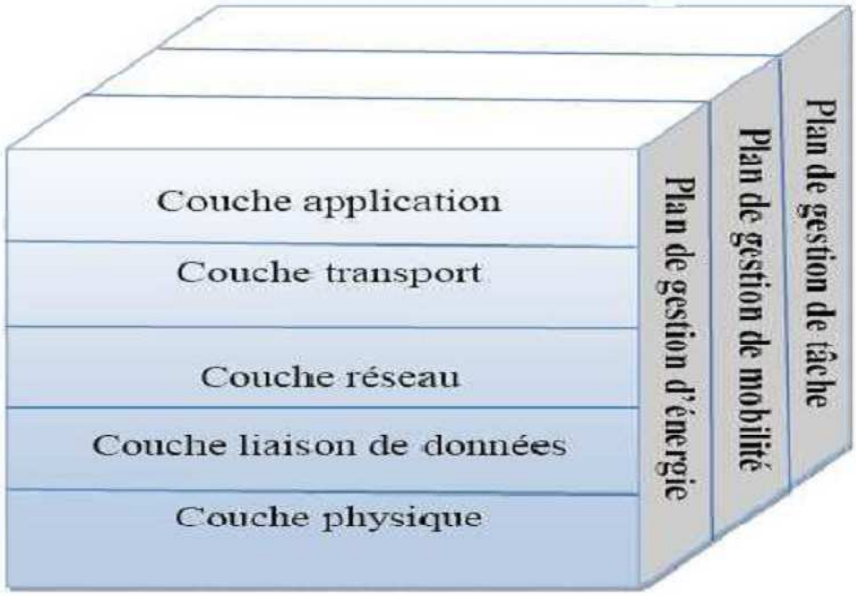

Figure I.7 Pile protocolaire des réseaux de capteurs. [13]

I.8.1 Couche application

Elle assure l'interface avec les applications. Il s'agit donc de la couche la plus proche des utilisateurs, gérée directement par les logiciels. Parmi les protocoles d'application, nous citons : SMP (Sensor Management Protocol) et TADAP (Task Assignement and Data Advertisement Protocol).

I.8.2 Couche transport

Elle vérifie le bon acheminement des données et la qualité de la transmission. Dans les RCSF, la fiabilité de transmission n'est pas majeure. Ainsi, les erreurs et les pertes sont tolérées. Par conséquent, un protocole de transport proche du protocole UDP et appelé UDP-Like(User Datagram Protocol Like) est utilisé.

Cependant, comme le protocole de transport universel est TCP (Transmission Control Protocol), les RCSF doivent donc posséder lors d'une communication avec un réseau externe, une interface TCP-spliting pour vérifier la compatibilité entre ces deux réseaux communicants.

I.8.3 Couche réseau

Elle s'occupe du routage de données fournies par la couche transport. Elle établit les routes entre les nœuds capteurs et le nœud puits et sélectionne le meilleur chemin en termes d'énergie, délai de transmission, débit, etc.

Les protocoles de routage conçus pour les RCSF sont différents de ceux conçus pour les réseaux Ad Hoc puisque les RCSF sont différents selon plusieurs critères comme :

-L'absence d'adressage fixe des nœuds tout en utilisant un adressage basé-attribut.

-L'établissement des communications multi-sauts.

-L'établissement des routes liant plusieurs sources en une seule destination pour agréger des données similaires, etc.

Parmi ces protocoles, nous citons : LEACH, SAR et SPIN etc.

I.8.4 Couche liaison de données

Elle est responsable de l'accès au media physique et la détection et la correction d'erreurs intervenues sur la couche physique. De plus, elle établit une communication saut-par-saut entre les nœuds, c'est-à-dire, elle détermine les liens de communication entre eux dans une distance d'un seul saut.

Parmi les protocoles de liaison de données, nous citons : SMACS (Self-organizing Medium Access Control for Sensor networks) et EAR (Eavesdrop And Register)

I.8.5 Couche physique

Elle permet de moduler les données et les acheminer dans le media physique tout en choisissant les bonnes fréquences. [14]

I.9 Domaines d'application des réseaux de capteurs

La miniaturisation des capteurs, le coût de plus en plus faible, la large gamme des types de capteurs disponibles ainsi que le support de communication sans fil utilisé, permettent aux réseaux de capteurs de se développer dans plusieurs domaines d'application. Ils permettent aussi d'étendre les applications existantes. 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, surveillance, environnemental, médical, domestique, commercial, etc. [15].

I.9.1 Applications militaires

Le faible coût et le déploiement rapide sont des caractéristiques qui ont rendu les réseaux de capteurs efficaces pour les applications militaires. Plusieurs projets ont été lancés pour aider les unités militaires dans un champ de bataille et protéger les villes contre des attaques, telles que les menaces terroristes. Le projet DSN (Distributed Sensor Network) [16] au DARPA (Defense Advanced Research Projects Agency) était l'un des premiers projets dans les années 80 ayant utilisé les réseaux de capteurs pour rassembler des données distribuées. Les chercheurs du laboratoire national Lawrence Livermore ont mis en place le réseau WATS (Wide Area Tracking System) [17]. Ce réseau est composé de détecteurs des rayons gamma et des neutrons pour détecter et dépister les dispositifs nucléaires. Il est capable d'effectuer la surveillance constante d'une zone d'intérêt. Il utilise des techniques d'agrégation de données pour les rapporter à un centre intelligent. Ces chercheurs ont mis en place ensuite un autre réseau appelé JBREWS (Joint Biological Remote Early Warning System) [18] pour avertir les troupes dans le champ de bataille des attaques biologiques possibles. Un réseau de capteurs peut être déployé dans un endroit stratégique ou hostile, afin de surveiller les mouvements des forces ennemies, ou analyser le terrain avant d'y envoyer des troupes (détection des armes chimiques, biologiques ou radiations). L'armée américaine a réalisé des tests dans le désert de Californie.

I.9.2 Applications à la surveillance

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

I.9.3 Applications environnementales

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

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

I.9.4 Applications médicales

Dans le domaine de la médecine, les réseaux de capteurs peuvent être utilisés pour assurer une surveillance permanente des organes vitaux de l'être humain grâce à des micro-capteurs qui pourront être avalés ou implantés sous la peau (surveillance de la glycémie, détection de cancers, etc.). Ils peuvent aussi faciliter le diagnostic de quelques maladies en effectuant des mesures physiologiques telles que : la tension artérielle, battements du cœur, etc. à l'aide des capteurs ayant chacun une tâche bien particulière. Les données physiologiques collectées par les capteurs peuvent être stockées pendant une longue durée pour le suivi d'un patient [19]. D'autre part, ces réseaux peuvent détecter des comportements anormaux (chute d'un lit, choc, cri, etc.) chez les personnes dépendantes (handicapées ou âgées).

I.9.5 Applications domestiques

Avec le développement technologique, les capteurs peuvent être embarqués dans des appareils, tels que les aspirateurs, les fours à micro-ondes, les réfrigérateurs, les magnétoscopes, etc. [20]. Ces capteurs embarqués peuvent interagir entre eux et avec un réseau externe via Internet pour permettre à un utilisateur de contrôler les appareils domestiques localement ou à distance.

Le déploiement des capteurs de mouvement et de température dans les futures maisons dites intelligentes permet d'automatiser plusieurs opérations domestiques telles que : la lumière s'éteint et la musique s'arrête quand la chambre est vide, la climatisation et le chauffage s'ajustent selon les points multiples de mesure, l'alarme est déclenchée par le capteur anti-intrusion quand un étranger veut pénétrer dans la maison.

I.9.6 Applications commerciales

Il est possible d'intégrer des capteurs au processus de stockage et de livraison dans le domaine commercial. Le réseau ainsi formé pourra être utilisé pour connaître la position, l'état et la direction d'un paquet. 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 localisation 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 [21].

I.10 Conclusion

Les réseaux de capteurs constituent un axe de recherche très fertile et peuvent être appliqués dans plusieurs domaines différents. Cependant, il reste encore de nombreux problèmes à résoudre dans ce domaine afin de pouvoir les utiliser dans les conditions réelles.

L'un des problèmes qu'on peut rencontrer dans ce genre de réseau est le routage: un protocole de routage calcule un chemin dans le réseau pour transférer les données de capteur en capteur jusqu'à atteindre la destination.

Dans le chapitre suivant, nous présenterons les protocoles de routages déployés pour les réseaux de capteurs sans fil.

II.1 Introduction

Un réseau de capteurs sans fil consiste en un ensemble de nœuds capteurs déployés à grande échelle dans un champs de captage pour détecter, recueillir et transmettre les données, concernant un phénomène observé, vers le nœud puits via les liens sans fil. Cependant, suivant le nombre de nœuds du réseau et l'étendu du champ de captage, certains nœuds ne pourront pas transmettre directement leurs messages au nœud collecteur. Ainsi, la collaboration entre les nœuds pour garantir cette transmission est une exigence. De cette manière, les messages sont propagés par les nœuds intermédiaires en établissant les chemins multi-sauts entre la source lointaine et le puits. Ce processus d'acheminement des messages d'un nœud source du réseau vers un nœud destinataire s'appelle le routage.

En tenant compte des capacités réduites des nœuds capteurs (calcul, énergie, mémoire), la communication avec le puits devrait se faire sans protocole de routage. Dans ce cas, la solution la plus simple serrait, pour chaque nœud capteur, d'envoyer ses messages par diffusions jusqu'à ce qu'ils arrivent au collecteur. Cependant cette simplicité provoque des désavantages significatifs tels que l'implosion et le chevauchement des messages. Une implosion est détectée parce que les nœuds reçoivent des copies multiples du même message (problème de redondance de données). De plus, les nœuds ne tiennent pas compte de leurs ressources pour limiter leurs opérations (émission, calcul).Ainsi, pour qu'un réseau de capteur soit efficace, la mise en place d'un algorithme de routage devient inévitable. Néanmoins, vu les contraintes imposées par ces réseaux, la mise en place d'un protocole de routage n'est pas une tâche facile.

Plusieurs travaux de recherche dans le domaine des RCSFs ont été effectués récemment et ont abouti à une multitude protocoles de routage destinés à ces réseaux.

Dans ce chapitre, nous faisons une étude bibliographique sur les protocoles de routage proposés pour les RCSFs. Vu leur multiplicité, il nous semble important de commencer par leur classification qui les groupe suivant un certain nombre de critères.

II.2 Classification des protocoles de routage pour les RCSFs

Récemment, les protocoles de routage pour les RCSFs ont été largement étudiés. Les protocoles proposés présentent les points communs et donc peuvent être classifiés suivant un certain nombre de critères. La **Figure II.1** ci-dessous résume une classification qui se base sur quatre critères : la topologie (structure) du réseau, mode d'établissement établissement des chemins, les paradigmes de communication et selon le mode de fonctionnement du protocole.

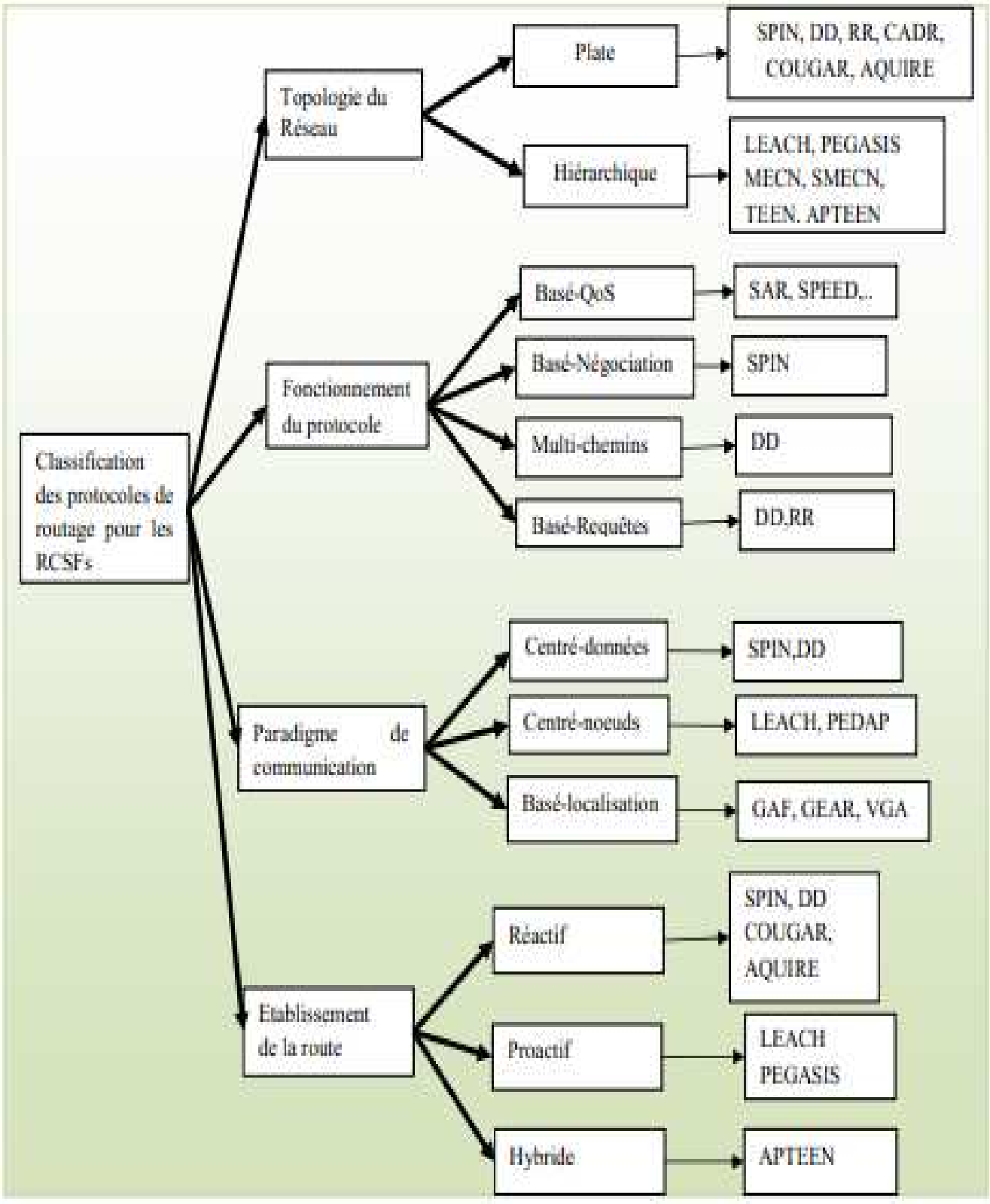

Figure II.1 Classification des protocoles de routage pour les Réseaux de capteurs sans fil.

II.2.1 Selon la topologie (structure) du réseau

La topologie détermine l'organisation des capteurs dans le réseau. Globalement, il existe deux topologies dans les RCSFs [22] : la topologie plate et la topologie hiérarchique.

II.2.1.1 Topologie plate

Dans une topologie plate, tous les nœuds capteurs possèdent le même rôle et collaborent entre eux pour accomplir la tâche de routage. Les réseaux plats sont caractérisés par : la simplicité des protocoles de routage, un coût de maintien réduit, une grande tolérance aux pannes ainsi qu'une habilité à construire de nouveaux chemins suite aux changements de topologie. Cependant, Les nœuds proches du puits participent plus que les autres aux tâches de routage. De plus, ces réseaux présentent une faible scalabilité dû au fonctionnement identique des nœuds et d'une manière distribuée nécessitant ainsi un grand nombre de messages de contrôle. La Figure suivante illustre l'organisation des capteurs dans une topologie plate.

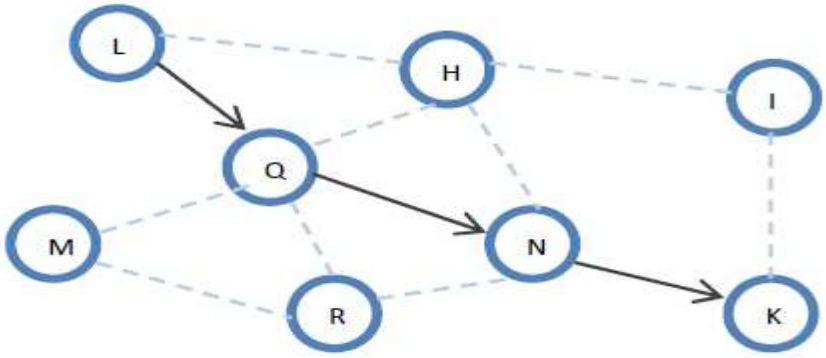

Figure II.2 Topologie plate

II.2.1.2 Topologie hiérarchique

Dans une topologie hiérarchique, les nœuds ont des différents rôles. En effet, certains nœuds sont sélectionnés pour exécuter des fonctions particulières. Une des méthodes les plus utilisées dans cette topologie est le clustering. Il consiste en un partitionnement du réseau en groupes appelés clusters. Un cluster est constitué d'un chef *(clusterhead)* et de ses membres. Suivant l'application, les membres peuvent être des voisins directs du chef ou pas.

Cette topologie présente beaucoup d'avantages tels que l'agrégation des données collectées ainsi qu'une grande scalabilité. Son inconvénient majeur est la surcharge des clusterheads qui induit un déséquilibre de la consommation d'énergie dans le réseau. Pour remédier à ce problème, clusterheads peuvent être des capteurs spécifiques avec plus de ressources énergétiques et plus de capacités de traitement ou bien ils peuvent être élus dynamiquement et ainsi garantir un équilibre de la consommation d'énergie et augmenter la tolérance aux pannes. Un exemple de cette topologie est donné dans la **Figure II.3** ci-dessous. Pour que les paquets générés par le nœud **F** atteignent le nœud **L**, ils doivent passer par les passerelles **P**, **S** et **R**.

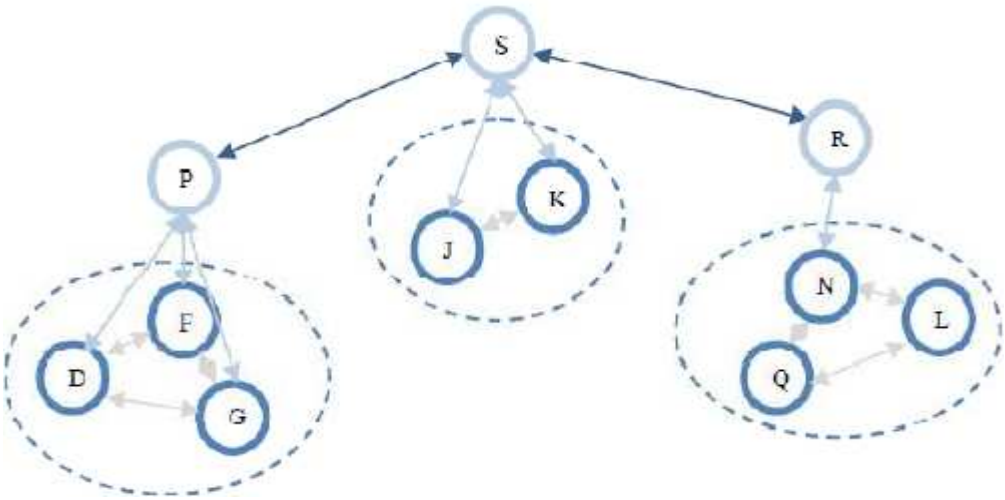

Figure II.3 Routage hiérarchique

II.2.2 Selon les paradigmes de communication

Le paradigme de communication détermine la manière dont les nœuds sont interrogés. Dans les RCSFs, il existe trois paradigmes de communication [23] : centré-nœuds, centré-données et basé sur la localisation.

II.2.2.1 Centré-nœuds (Node-centric)

Ce paradigme est celui employé dans les réseaux conventionnels, où il est nécessaire de connaître et d'identifier les nœuds communicants (comme l'adresse IP). Les réseaux ad hoc utilisent ce genre de paradigme, qui s'intègre bien avec l'utilisation de ce type d'environnement. Cependant, pour les réseaux de capteurs, un routage basé sur une identification individuelle des nœuds ne reflète pas l'usage réel du réseau. Pour cela, un autre paradigme a été introduit : data centric. Néanmoins, le paradigme node centric n'est pas à écarter totalement, car certaines applications nécessitent une interrogation individuelle des capteurs.

II.2.2.2 Centré-données (Data-centric)

Ce paradigme suppose qu'il est difficile d'avoir des identifiants comme les adresses MAC ou IP pour pouvoir communiquer entre les nœuds capteurs [24]. Ainsi, le routage ne se fait pas en fonction d'une adresse de destination, mais suivant les données disponibles au niveau des capteurs. Ces données seront propagées de proche en proche pour arriver au nœud puits.

II.2.2.3 Basé-localisation (location-based)

Dans cette approche, les décisions de routage sont établies selon la position des nœuds. La distance entre les nœuds voisins peut être estimée sur la base de la puissance du signal arrivé. Un tel type de routage nécessite que les nœuds aient connaissance de leurs positions géographiques. Par conséquent, ce type de mécanismes nécessite un déploiement d'une solution de positionnement, dont le degré de précision requis dépend de l'application ciblée. L'utilisation du GPS reste trop coûteuse pour un **RCSF.**

Néanmoins, d'autres méthodes de localisation et de positionnement des capteurs ont été développées comme par exemple la triangulation [25].

II.2.3 Selon le mode de fonctionnement du protocole

Le mode de fonctionnement définit la manière avec laquelle les données sont propagées dans le réseau. Selon ce critère, les protocoles de routage peuvent être classifiés en quatre catégories : routage basé sur la Qualité de Service "QdS" (Quality of Service "QoS" based routing), routage basé sur les requêtes (query-based routing), routage multi-chemins (Multi-path routing), et routage basé sur la négociation (Negociation based routing) [22].

II.2.3.1 Routage basé sur les multi-chemins

Dans cette catégorie, les protocoles de routage utilisent des chemins multiples plutôt qu'un chemin simple afin d'augmenter la performance du réseau. La fiabilité d'un protocole peut être mesurée par sa capacité à trouver des chemins alternatifs entre la source et la destination en cas de défaillance du chemin primaire. Pour cette raison, certains protocoles construisent plusieurs chemins indépendants, c.-à-d. : ils ne partagent qu'un nombre réduit (voir nul) de nœuds. Malgré leur grande tolérance aux pannes, ces protocoles requièrent plus de ressources énergétiques et plus de messages de contrôle.

II.2.3.2 Routage basé sur les requêtes

Dans ce type de routage, le puits génère des requêtes afin d'interroger les capteurs. Ces requêtes sont exprimées soit par un schéma valeur-attribut ou bien en utilisant un langage spécifique (par exemple SQL : Structured Query Language). Les nœuds qui détiennent les données requises doivent les envoyer au nœud demandeur à travers le chemin inverse de la requête. Les requêtes émises par le puits peuvent aussi être ciblées sur des régions spécifiques du réseau.

II.2.3.3 Routage basé sur la négociation

En détectant le même phénomène, les nœuds capteurs inondent le réseau par les mêmes paquets de données. Ce problème de redondance peut être résolu en employant des protocoles de routage basés sur la négociation. En effet, avant de transmettre, les nœuds capteurs négocient entre eux leurs données en échangeant des paquets de signalisation spéciales, appelés métadonnées. Ces paquets permettent de vérifier si les nœuds voisins disposent déjà de la donnée à transmettre. Cette procédure garantit que seules les informations utiles seront transmises et élimine la redondance des données.

II.2.3.4 Routage basé sur la qualité de service

Dans les protocoles de routage basés sur QoS, le réseau doit équilibrer entre la consommation d'énergie et la qualité de données. En particulier, le réseau doit satisfaire certaines métriques de QoS, par exemple, retard, énergie, largeur de bande passante, etc. Les protocoles de cette approche sont très recommandés pour les applications de surveillance (centrales nucléaires, applications militaires, etc.).

II.2.4 Selon le mode l'établissement des chemins

Suivant la manière de création et de maintien des chemins pendant le routage, nous distinguons trois catégories de protocoles de routage : les protocoles proactifs, les protocoles réactifs et les protocoles hybrides [26].

II.2.4.1 Les protocoles proactifs

Utilisent l'échange régulier de messages de contrôle pour maintenir au niveau de chaque nœud des tables de routage (qui associent à chaque destination ou groupe de destinations un voisin direct par lequel les paquets doivent être relayés) vers toute destination atteignable depuis celui-ci. Ces tables sont maintenues mêmes quand les routes ne sont pas utilisées. Cette approche permet de disposer d'une route vers chaque destination immédiatement au moment où un paquet doit être envoyé. Les protocoles proactifs sont adaptés aux applications qui nécessitent un prélèvement périodique des données. Et par conséquent, les capteurs peuvent se mettre en veille pendant les périodes d'inactivité, et n'enclencher leur dispositif de capture qu'à des instants particuliers. On peut citer par exemple le protocole DSDV (Destination Sequenced Distance Vector)

Le protocole de routage DSDV (Destination Sequenced Distance Vector)

Le protocole DSDV est basé sur l'algorithme distribué de Bellman-Ford. Chaque nœud du réseau maintient dans sa table de routage un ensemble d'informations pour chaque destination contenant :

– l'adresse du destinataire,

– le nombre de sauts pour l'atteindre, c'est à dire le nombre de liens de communication existant pour atteindre ces destinations

– un numéro de séquence associé au destinataire. Il est utilisé pour faire la distinction entre les anciennes routes et les nouvelles routes découvertes vers cette destination pour éviter la formation des boucles de routage.

Afin de maintenir la consistance des tables de routage dans une topologie qui change rapidement, chaque nœud du réseau transmet périodiquement sa table de routage à ses voisins directs. Le nœud peut aussi transmettre sa table de routage si le contenu de cette dernière subit des changements significatifs par rapport au dernier contenu envoyé. Afin de limiter le trafic occasionné par toutes ces mises à jour, il existe deux types de mise à jour :

– des mises à jour complètes ;

– des mises à jour incrémentales.

Dans la mise à jour complète, la station transmet la totalité de la table de routage aux voisins. Dans les mises à jour incrémentales, seules les nouvelles entrées ou celles qui ont subi un changement, par rapport à la dernière mise à jour, sont envoyées.

II.2.4.2. Les protocoles réactifs

Dits aussi les protocoles de routage à la demande, créent et maintiennent les routes selon les besoins. Lorsqu'un nœud à besoin d'une route, une procédure de découverte globale est lancée. Cette procédure s'achève par la découverte de la route ou lorsque toutes les permutations de routes possibles ont été examinées. La route trouvée est maintenue par une procédure de maintenance de routes jusqu'à ce que la destination soit inaccessible à partir du nœud source ou que le nœud source n'aura plus besoin de cette route. On dénombre essentiellement deux algorithmes :

AODV (Ad hoc On demand Distance VectorRouting) qui est un protocole de type vecteur de distance, utilise la technique Expanding Ring qui limite le nombre de sauts des paquets de recherche de route à partir du nœud source C.E. (Perkins, Belding-Royer et al., 2002).

DSR (Dynamic Source Routing) qui utilise la technique de routage par source et possède le même mécanisme qu'AODV pour limiter le nombre de sauts d'une recherche de route (Johnson, Maltz et al.).

II.2.4.3. Protocoles hybrides

Ces protocoles combinent les deux idées des protocoles proactifs et réactifs. Ils utilisent un protocole proactif pour apprendre le proche voisinage (par exemple le voisinage à deux ou à trois sauts), ainsi, ils disposent de routes immédiatement dans le voisinage. Au-delà de la zone du voisinage, le protocole hybride fait appel à un protocole réactif pour chercher des routes. Le principal protocole est **ZRP** (Zone Routing Protocol) fondé sur la technique vecteur de distance (Haas, Pearlman et al. 2001).

II.3 Exemples des protocoles de routage dans les RCSFs

II.3.1 Propagation et discussion (flooding and gossiping)

Pour le *flooding*, chaque capteur recevant un paquet de données le renvoie à tous ses voisins et ce processus continue jusqu'à ce que le paquet arrive à la destination ou le nombre maximum de sauts pour le paquet (TTL) est atteint. Ce protocole présente un ensemble d'inconvénients dont:

- ✓ *Implosion* : le même message est dupliqué plusieurs fois ; chaque nœud reçoit autant de fois la même donnée que le nombre de ses voisins ;
- ✓ *Chevauchement* : si deux nœuds observent le phénomène dans la même région, la même information sera envoyée deux fois (redondance des données) ;
- ✓ Il utilise aveuglement les ressources disponibles sans tenir en compte de leur quantité.

D'autre part, le *gossiping* est une version légèrement améliorée du *flooding* où le nœud récepteur envoie un paquet à un sous ensemble de ses voisins choisis aléatoirement. L'implosion n'est plus un inconvénient pour ce protocole. Cependant, l'envoi d'un message à tous les nœuds (un à un) demande plus de temps ;

II.3.2 Directed Diffusion(DD)

Directed Diffusion [27, 28] est un protocole de routage de catégorie *data-centric*, permettant d'utiliser plusieurs chemins pour le routage d'information. Le principe de fonctionnement du protocole DD est le suivant :

Le nœud «*Sink*» commence à envoyer, vers tous les nœuds, un message *Interest* pour démarrer une application bien déterminée. Ce paquet sera acquitté par un autre appelé *gradient*. Un gradient est un lien de réponse de la part du voisin recevant l'intérêt. En utilisant les intérêts et les gradients, plusieurs chemins peuvent être établis entre le «*Sink*» et la source. L'un de ces chemins est sélectionné par renforcement. Si ce chemin échoue un nouveau ou un alternatif doit être identifié La **figure II.4** suivante illustre les phases de fonctionnement de ce protocole

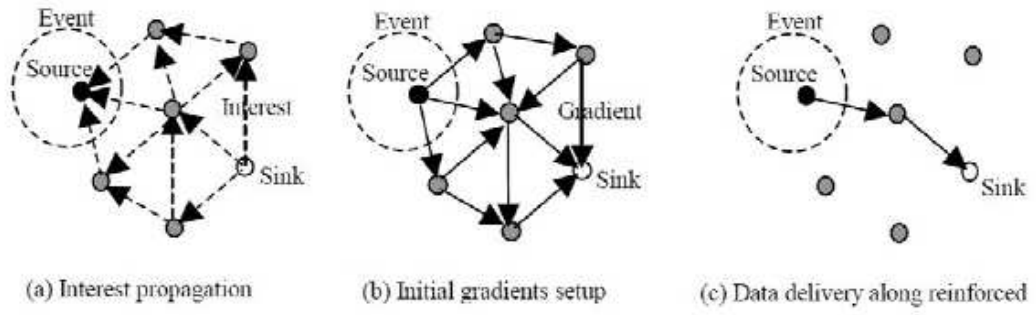

Figure II.4 Les phases de communication du protocole Directed Diffusion

II.3.3 Rumor Routing (RR)

Rumor routing [29] est une variante du protocole Directed Diffusion. Généralement Directed Diffusion utilise une forme d'inondation pour la propagation des requêtes (intérêts). Cependant, dans certains cas il y a seulement une petite quantité de données demandée des nœuds et ainsi l'utilisation d'inondation est inutile. Une approche alternative est d'inonder les événements si le nombre d'événements est petit et le nombre des requêtes est grand. Le protocole Rumor Routing essaie de trouver un compromis entre l'inondation des intérêts et la propagation des données.

L'idée est de router des requêtes aux nœuds qui ont observé un événement particulier plutôt qu'inonder le réseau entier pour récupérer des renseignements sur les événements se produisant. Pour inonder l'événement à travers le réseau, l'algorithme Rumor Routing utilise le concept d'*agent*.

Un agent est un paquet avec une grande portée (TTL) qui traverse le réseau et informe de nœud en nœud en informant ces nœuds des événements qu'il a rencontrés durant toute cette traversée.

Quand un nœud détecte un événement, il l'ajoute à sa table locale et génère un paquet *agent*. Ce paquet parcourt 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 se référant à la table d'événement.

II.3.4 Sensor Protocols for Information via Negotiation: (SPIN)

SPIN [30] est parmi les premiers protocoles de routage *data-centric* basé sur la *négociation*. SPIN est basé sur l'idée que les nœuds capteurs opèrent plus efficacement et conservent l'énergie en envoyant des données qui décrivent les données des capteurs au lieu d'envoyer les données entières, à moins que les données entières ne soient explicitement demandées. Cela permet de pallier au problème d'inondation. Pour cela, SPIN utilise trois types de messages : ADV (ADVertise), REQ (REQuest) et DATA.

Avant d'envoyer une donnée entière (message DATA), un nœud diffuse un message ADV qui contient la description, c.-à-d. méta-données, de la donnée en question. Un nœud recevant un message ADV, consulte sa base d'intérêt. S'il est intéressé par cette information, il émet un message REQ vers son voisin (émetteur d'ADV). En recevant un message REQ, l'émetteur transmet à l'intéressé la donnée sous forme d'un message DATA. Les nœuds voisins répètent ainsi cette opération, comme le montre la **figure II.5**. Comme résultat, les nœuds qui sont intéressés par la donnée en auront une copie.

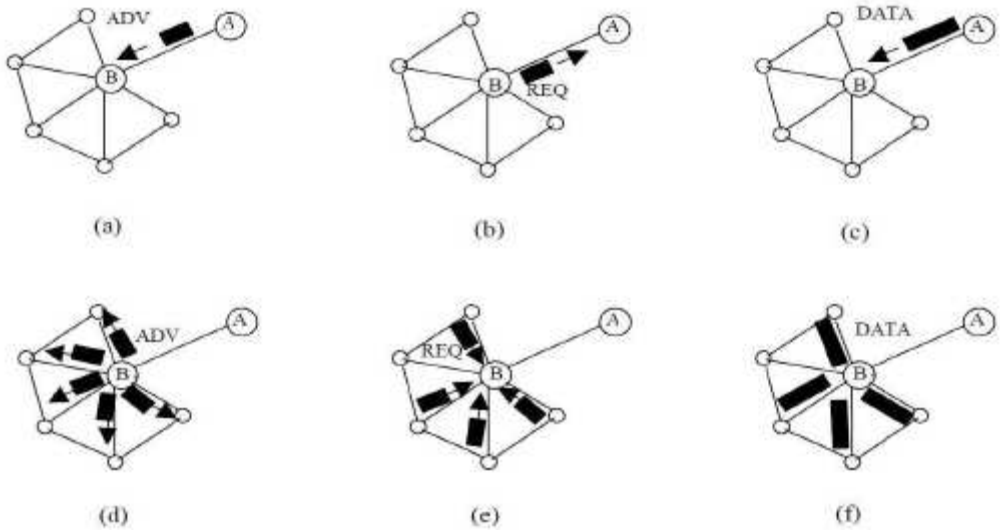

Figure II.5 Le protocole SPIN

Sur cette la figure, le nœud A commencé par aviser ses données au nœud B (a). Le nœud B répond en envoyant la requête au nœud B (b). Après avoir reçu la donnée demandée (c), le nœud B avise cette donnée à ses voisins (d), qui rendent à tour de rôle des requêtes à B (e-f).

Le protocole SPIN accommode son exécution suivant l'énergie restante du capteur, et modifie en conséquence le comportement du nœud. En effet, les nœuds contrôlent leur niveau d'énergie d'une manière continue. Lorsque le nœud s'aperçoit que son niveau d'énergie a atteint un certain seuil, il change son mode de fonctionnement, et ne répond à aucun message ADV. Notons que le format des méta-données utilisé dans SPIN n'est pas standardisé (dépend de l'application) [31].

II.3.5 Low-Energy Adaptive Clustering Hierarchy (LEACH)

LEACH [32, 33] est le premier et le plus populaire des algorithmes de routage hiérarchiques avec une efficacité énergétique, qui a été proposé pour les RCSFs [22].LEACH emploie la technique de clustering qui divise le réseau en deux niveaux : les clusterheads (CHs) et les nœuds membres. Le protocole se déroule en rounds. Chaque round se compose de deux phases : construction et communication. La durée de la phase de communication est plus longue que celle de la phase de construction afin de minimiser l'overhead.

II.3.5.1.Phase de construction

Le but de cette phase est la construction des clusters en choisissant les chefs (CHs) et en établissant la politique d'accès au média au sein de chaque groupe. Durant cette phase, chaque nœud *n* choisit un nombre aléatoire compris entre 0 et 1. Si ce nombre est inférieur à un seuil T(n), le nœud devient clusterhead. T (n) est défini comme suit [33] :

$$T(n)=\begin{cases}\frac{p}{1-p*(r\ mod\frac{1}{p})} & Si\ n \in G \\ 0 & Sinon\end{cases}$$

Avec:

p : pourcentage des nœuds désirant devenir clusterhead.

r : numéro du round courant.

G: ensemble de nœuds n'ayant pas été élus clusterheads durant les 1/P dernières périodes précédents. Par la suite, chaque nœud qui s'est élu clusterhead émet un message de notification à tous ces voisins. Les nœuds qui récoltent les messages de notification, décident leur appartenance à un cluster. La décision est basée sur l'amplitude du signal reçu : le clusterhead ayant le signal le plus fort est choisi (i.e. le plus proche). En cas d'égalité, un chef aléatoire est choisi. Chaque membre informe son chef de sa décision.

II.3.5.2. Phase de communication

Pendant cette phase, les nœuds capteurs peuvent commencer à envoyer leurs données captées au clusterhead pendant leurs propres slots. Cela leur permet d'éteindre leur interface de communication en dehors de leurs slots réservés, afin d'économiser leur énergie. Le clusterhead agrège les données reçues avant de les transmettre au collecteur (puits). Cette communication, entre un clusterhead et le collecteur, se fait d'une manière directe. Après la phase de communication, qui dure un certain temps, la phase de construction recommence.

II.3.6 Cougar [34]

Le réseau est vu comme une base de données distribuée où quelques nœuds contenant les renseignements sont temporairement inaccessibles. Les données produites par le réseau de capteurs sont modélisées comme une table relationnelle.

Dans cette table, chacun des attributs représente soit des informations sur le nœud capteur ou bien des données produites par ce nœud. Pour pouvoir interroger les nœuds du réseau, Cougar fournit une interface semblable à SQL (Structured Query Language) étendue, au niveau du puits. Cette interface permet au puits de générer des requêtes pour interroger un nœud spécial du réseau, appelé leader (chef). L'approche Cougar fournit une agrégation partielle au niveau des nœuds. Chaque nœud maintient une liste d'attente contenant les nœuds fils qui doivent lui envoyer les paquets. Le nœud n'émet le paquet agrégé au prochain saut que s'il a reçu les paquets de tous les nœuds de la liste d'attente. Cependant, un nœud peut devenir inaccessible à cause du mouvement ou d'un problème de batterie. Pour cela, Cougar utilise un timer afin d'éviter une attente indéfinie.

II.3.7 ACQUIRE (Active Query Forwarding in Sensor Networks) [35]

Cet algorithme considère aussi un RCSF comme une base de données distribuée. Dans ce plan, un nœud injecte un paquet de requête actif dans le réseau. Le voisin qui découvre que le paquet contient des informations obsolètes, émet un message de mise à jour au nœud source du paquet. Alors, le nœud choisit au hasard un voisin pour propager la requête. Comme la requête active progresse à travers le réseau, elle est progressivement résolue en plus petits composants de jusqu'à ce qu'elle soit complètement résolue. Alors, la requête est retournée au nœud source comme une réponse complète.

II.3.8 Geographic and Energy Aware Routing (GEAR) [36]

Le protocole GEAR découpe le réseau en régions. Chaque nœud connaît le coût pour atteindre chaque région. L'acheminement des paquets suit les étapes suivantes :

- ✓ Acheminer le paquet jusqu'à la région, en envoyant le paquet au nœud le plus proche de la région parmi ses voisins et ayant le niveau d'énergie résiduelle le plus élevé (fonction de distance et d'énergie),
- ✓ Acheminer le paquet dans la région de destination par une sorte de diffusion si le nombre de nœud n'est pas élevé, sinon la région est découpée en sous-région et le paquet est transmis individuellement à chaque sous-région.

Chaque paquet contient la région destination. Chaque nœud connaît sa position, son énergie résiduelle, la position et l'énergie résiduelle de ses voisins (à la demande). Un lien existe entre 2 nœuds quand ils sont à portée et leur niveau d'énergie leur permet d'effectuer l'envoi.

II.3.9 Sequential Assignment Routing (SAR)

SAR [37] est l'un des premiers protocoles de routage pour les RCSFs qui introduit la notion de qualité de service (QoS) dans les décisions de routage [38]. SAR est une approche multi-chemins qui s'efforce à réaliser l'efficacité énergétique et la tolérance aux pannes. Pour cela, SAR crée des arbres en prenant en compte les trois facteurs : métriques QoS, la ressource énergétique sur chaque chemin et le niveau de priorité de chaque paquet. En utilisant ces arbres, des routes multiples du *sink* aux capteurs sont

formées. Une ou plusieurs routes peuvent alors être empruntées. Pour assurer la tolérance aux pannes, SAR recalcule périodiquement les routes à choisir en cas de défaillance d'un nœud ou de changement de topologie [39, 40].

II.3.10 SPEED [41]

SPEED est un autre protocole de routage basé sur la QoS qui garantit le temps réel et le routage bout en bout. Une métrique supplémentaire par rapport à GEAR : le délai. En se basant sur une table de positions, SPEED estime le délai sur chaque saut en calculant le délai d'aller-retour (en retranchant le temps de traitement côté récepteur). Autrement dit, SPEED assure une vitesse de livraison des paquets constante, qu'on note Setspeed. Ceci permet de garantir des délais de livraison de bout en bout acceptables. Après avoir déterminé le délai, le prochain saut est choisi parmi les voisins qui sont plus proches de la destination.

II.3.11 TEEN et APTEEN:

TEEN (Threshold sensitive Energy Efficient sensor Network protocol) [42] est un protocole hiérarchique conçu pour être sensible aux changements improviste des attributs détectés tels que la température. L'architecture du réseau est basée sur un groupement hiérarchique où les nœuds les plus proches forment des clusters. Après la construction des clusters, le clusterhead diffuse deux seuils aux nœuds. Qui sont la valeur minimale d'un attribut pour pouvoir être transmis et le degré minimale du changement de cet attribut.

APTEEN (*Adaptive Periodic TEEN*) [43] est une extension du TEEN basée sur la capture périodique des données et la réaction aux événements temps-réel. Quand la station de base forme les clusters, les cluster-heads diffusent les attributs, les seuils et le plan de transmission à tous les nœuds et effectuent également l'agrégation des données afin d'économiser l'énergie.

II.4 Conclusion

Le routage dans les RCSFs forme un axe de recherche intéressant, avec un ensemble limité, mais en pleine croissance de résultats des recherches. Cette croissance fortement liée à deux facteurs : Les capteurs sont des véritables systèmes embarqués et donc sont conçus pour les applications spécifiques, l'expansion du champ de domaine d'application des RCSFs.

Les protocoles de routage proposés pour les RCSFs sont donc nombreux, mais ils ont tous un objectif commun : Assurer l'acheminement des données collectées par les nœuds capteurs tout en essayant d'étendre la durée du réseau. Cela nécessite la prise en compte des caractéristiques des RCSFs et des exigences des applications pour lesquelles ces réseaux sont destinés.

Dans ce chapitre, nous avons identifié et classifié quelques protocoles de routage pour les RCSFs proposés dans la littérature. De façon générale, ces protocoles sont classifiés selon la structure du réseau, le fonctionnement des protocoles, l'établissement de la route, ou les paradigmes de communication.

III.1 Introduction

Dans un réseau, les nœuds peuvent communiquer de trois façons différentes:

- **Monodiffusion :** processus consistant à envoyer un paquet d'un hôte à un autre.
- **Diffusion :** processus consistant à envoyer un paquet d'un hôte à tous les hôtes du réseau.
- **Multidiffusion :** processus consistant à envoyer un paquet d'un hôte à un groupe d'hôtes en particulier.

La monodiffusion est utilisée dans les communications normales d'hôte à hôte tant entre client et serveur que dans un réseau peer to peer. Les paquets monodiffusion utilisent l'adresse hôte du périphérique de destination comme adresse de destination, et peuvent être routés via un interréseau. En revanche, la diffusion et la multidiffusion utilisent des adresses de destination spécifiques.

Puisque le trafic de diffusion est utilisé pour envoyer des paquets à tous les hôtes du réseau, les paquets utilisent des adresses de diffusion spécifiques. Lorsqu'un hôte reçoit un paquet avec comme destination une adresse de diffusion, il traite le paquet comme s'il était adressé à son adresse monodiffusion.

La transmission de diffusion permet de localiser des services et périphériques spéciaux pour lesquels l'adresse n'est pas connue, ou lorsqu'un hôte doit fournir des informations à tous les hôtes sur le réseau. Voici quelques cas d'utilisation des transmissions de diffusion :

- Mappage des adresses d'une couche supérieure à des adresses d'une couche inférieure.
- Demande d'une adresse.
- Échange d'informations de routage entre des protocoles de routage.

Lorsqu'un hôte a besoin d'informations, il envoie une demande, appelée «requête», à l'adresse de diffusion. Tous les hôtes du réseau reçoivent et traitent cette requête. Un ou plusieurs hôtes disposant des informations requises répondent, en principe, à l'aide d'un message monodiffusion.

III.2 Diffusion

La diffusion (ou broadcast en anglais) est une manière d'acheminer les messages, de type un vers tous, dans laquelle un hôte source désire transmettre un message à l'ensemble du réseau. Aucun routage n'est nécessaire pour effectuer une diffusion, puisqu' aucune route n'est requise. Les applications de cette opération sont nombreuses, telles que la découverte de routes, la découverte de services, le lancement d'alertes au sein du réseau, la synchronisation ou encore la dissémination d'informations ou d'ordres pour un réseau de capteurs. La diffusion est donc un processus dont l'efficacité est primordiale pour le bon fonctionnement du réseau.

Les portées de communication des objets sans fil étant limitées, on considère généralement qu'il est impossible pour l'hôte source de pouvoir contacter directement l'ensemble du réseau. Il est nécessaire d'utiliser un mécanisme multi-sauts, et donc de faire participer de nombreux hôtes au processus.

La manière la plus simple pour effectuer une diffusion est connue sous le nom de diffusion aveugle (ou blind flooding en anglais). Son principe est le suivant : chaque hôte recevant pour la première fois le message à diffuser ré-émet celui-ci à destination de ses voisins. Si le réseau est connexe (il existe un chemin entre la source et n'importe quel autre hôte) et que l'on suppose l'absence de collisions, alors ce processus aboutit à une couverture complète du réseau. Malheureusement, cet algorithme très simple n'est pas efficace car il requiert la participation de tous les hôtes, alors que cela n'est pas toujours nécessaire. En conséquence, il conduit à une grande quantité de messages redondants et d'énergie gaspillée [44].

Parmi les solutions alternatives plus efficaces qui ont été proposées, beaucoup sont centralisées. Cela signifie que le protocole nécessite une connaissance globale du réseau pour fonctionner : l'existence de chaque hôte, les communications possibles entre eux et potentiellement leur position. Rassembler cette connaissance est clairement irréaliste dans un environnement tel que celui des réseaux ad hoc ou de capteurs : le coût de cette opération serait énorme, et la durée de vie des informations ainsi obtenues très courte. Des solutions dites ≪localisées≫ ont donc été explorées, qui ne nécessitent quant à elles que des informations sur le voisinage immédiat de chaque hôte

III.3 Caractéristiques de diffusion

Pour évaluer un algorithme de diffusion, il existe plusieurs caractéristiques [45]:

III.3.1 Fiabilité

Une diffusion est dite fiable lorsqu'elle permet de transmettre l'information à l'ensemble des nœuds joignables. D'un point de vue local, chaque nœud doit alors garantir que l'ensemble de son voisinage est contacté par le message de diffusion. Ainsi, si le réseau est connexe (chaque nœud est joignable), la propriété précédente garantit une couverture totale du réseau. À l'opposé, un protocole de diffusion est dit non-fiable (unreliable) lorsque l'algorithme ne garantit pas la diffusion de l'information à l'ensemble des nœuds joignables. Certains algorithmes, comme les protocoles de diffusion probabiliste. Une telle propriété peut paraître rédhibitoire mais elle possède certaines caractéristiques intéressantes. Par exemple, certains algorithmes de diffusion peuvent proposer une diffusion partielle de l'information, dans le but d'informer uniquement un sous-ensemble du réseau pour économiser certaines ressources critiques.

III.3.2 Déterminisme

Ce paramètre caractérise le comportement du mobile lors de sa décision de retransmettre à son voisinage le message de diffusion qu'il vient de recevoir. Ainsi, un algorithme de diffusion peut être déterministe ou probabiliste. Dans le premier cas, le mobile prend la décision en fonction de son état et des informations reçues dans le paquet. Le modèle probabiliste, quant à lui, décide d'une probabilité de retransmission (éventuellement à partir de certaines caractéristiques du réseau, et de l'état du mobile). L'utilisation d'un modèle probabiliste est intéressante pour son comportement en moyenne : même si certaines décisions locales ne donnent pas le meilleur résultat, la réaction de l'ensemble des nœuds permet d'obtenir des résultats acceptables en moyenne (à un niveau global). Cette approche permet de pallier le manque d'information sur le voisinage, en offrant une sélection probabiliste comme hypothèse sur la partie manquante.

III.3.3 Information sur le réseau

Chaque mobile déduit, en général, son choix de retransmission à partir d'informations sur la topologie du réseau. Ces indications peuvent être contenues dans le message de diffusion et/ou mémorisées lors des communications précédentes. On peut classer en plusieurs catégories la quantité d'information nécessaire pour chaque algorithme de diffusion.

III.3.4 Le contenu des messages

Pour assurer un bon déroulement de la diffusion, chaque nœud émet des messages concernant des informations sur le réseau, son état interne et/ou sur la diffusion en elle-même. Ce surplus se trouve dans les deux types de paquets suivant :

Les messages **HELLO** sont émis régulièrement par chaque nœud pour informer leur voisinage sur sa position s'il possède un diapositif GPS, son état interne (le nombre de nœuds qu'il peut joindre) et la liste de ses voisins.

Les messages **BROADCAST** sont émis lors de l'opération de diffusion. Chacun contient les données à transmettre à l'ensemble des nœuds, mais ils peuvent aussi posséder des informations utilisées par le protocole comme les numéros de séquence, qui permettent d'identifier de manière unique le message de diffusion. Ils peuvent également indiquer des listes de voisins (pour informer le correspondant du voisinage joint par ce message), ou un sous-ensemble de la liste des voisins dans l'émetteur (pour indiquer les voisins qui doivent retransmettre le message).

III.4 Évaluer la qualité d'un broadcast

L'efficacité d'un algorithme d'inondation est mesurable avec plusieurs métriques [46]:

Accessibilité : le pourcentage de terminaux du réseau qui sont atteints par la procédure de diffusion. Ce pourcentage doit être aussi proche de 100 que possible ;

Taux de communication : le nombre de retransmission que le protocole génère divisé par le nombre de retransmission générées par le protocole d'inondation aveugle ;

Latence : le temps qui s'écoule entre l'émission du message diffusé et la réception la plus tardive de celui-ci par un nœud de réseau.

III.5 Études des protocoles existants [47,46]

Dans les réseaux de capteurs, les nœuds ne possèdent aucune connaissance préalable sur l'organisation du réseau (topologie), la découverte de l'environnement passe par une opération de diffusion dans l'ensemble du réseau, encore appelé inondation ou flooding. Il s'agit de l'opération de communication qui permet à un terminal d'envoyer un message à destination de l'ensemble des terminaux du réseau de capteur, directement à portée radio ou accessible en utilisant les nœuds du réseau comme relais. Cette opération permet à différents messages de contrôle de parvenir à l'ensemble des nœuds du réseau ou encore d'atteindre n'importe quel terminal lorsque l'on n'a pas d'informations suffisantes sur le réseau. Bien évidemment elle implique une trop grande consommation des ressources radio du réseau pour être utilisée pour l'acheminement de données. On aura donc recours à l'inondation uniquement pour découvrir l'environnement, auto-organiser le réseau et pour permettre ainsi la mise en place d'algorithmes de routage beaucoup plus efficaces.

La plus grande partie du temps, les protocoles de routage de réseaux de capteurs emploient des protocoles très basiques pour réaliser cette opération pourtant très consommatrice de bande passante et susceptible de faire chuter considérablement les performances d'un réseau.

Les algorithmes d'inondation peuvent être classés en 4 grandes catégories : inondation aveugle, inondation probabiliste, inondation géographique et inondation sur information sur le réseau.

L'algorithme le plus trivial pour diffuser un message est **le protocole d'inondation aveugle (blind flooding),** qui consiste simplement pour le nœud émetteur à envoyer son message à ses voisins, chaque nœud du réseau émettant à nouveau tous les messages d'inondation qu'il reçoit. La seule optimisation de ce protocole consiste à étiqueter de façon unique chaque message d'inondation pour permettre qu'un nœud du réseau qui reçoit de plusieurs voisins le même message d'inondation ne l'émettre qu'une seule fois. Chaque nœud doit donc concevoir pendant un intervalle de temps supérieur de réalisation d'une inondation les messages d'inondation reçus pour les considérer comme nuls lors des éventuelles réceptions redondantes.

Bien qu'il ne nécessite qu'une table pour mémoriser les identifiants des messages d'inondation, l'algorithme d'inondation est quand même un algorithme coûteux. Malgré sa simplicité, il impose une charge énorme au réseau, car chaque voisin se doit de renvoyer le message. Ce grand nombre de mobiles essayant d'accéder en même temps entraîne une probabilité importante de collisions parmi les rediffusions. Car les

messages d'inondation sont envoyés sans accord préalable, en minimisant le temps d'attente dû à l'écoute du médium (pour réduire le temps de latence entre la réception et la réémission). Le défaut est simple, l'algorithme est naïf. Lorsque deux mobiles très proches se décident à réémettre le même message, une grande partie des voisins de ces deux mobiles vont recevoir exactement la même information. Cette utilisation doublon est pourtant inutile. De plus, tous les voisins vont essayer de réémettre le message, entraînant ainsi des contentions pour l'accès au médium.

Dans **les méthodes probabilistes**, chaque nœud du réseau ne retransmettra un message d'inondation qu'avec une probabilité p fixée par l'algorithme. Lorsque les réseaux sont très denses, ce type de protocole peut donner de bonnes performances. On peut également ajouter aux messages de la table des dernières diffusions reçus un champ permettant de compter le nombre de fois qu'un message m a été reçu ; le terminal retarde la transmission de m pendant un temps fixé par l'algorithme et même renonce à cette retransmission si m a été reçu plus de k fois, k étant fixé préalablement par l'algorithme (on suppose que si k voisin ont déjà retransmis le message, il est très fortement probable que de nouvelles retransmissions n'atteindront aucun nouveau nœud).

Même s'il offre une accessibilité correcte (supérieure à 90%) dans le cas de très forte densité, il est néanmoins nettement en position de faiblesse, car chaque nœud possède les mêmes chances de réémettre quelle que soit sa position par rapport au voisin qui lui transmet le message de diffusion, sans tenir compte de la topologie du voisinage.

On peut encore également baser la décision de retransmettre sur des critères de distance (j'ai reçu le message d'un voisin très proche de moi, alors je ne retransmets pas). Si un de ces messages arrive d'une distance inférieure à D alors il annule son émission. Dans le cas contraire, et au bout du temps d'attente, il réémet le message de diffusion.

Le défaut provient du fait que même si un mobile a entendu plusieurs fois le message de diffusion, il peut quand même le réémettre.

Enfin, on peut également améliorer les performances des protocoles de diffusion par la construction d'une structure organisant les nœuds du réseau. Cette structure devra être construite localement (uniquement entre les voisins proches) et offrir des caractéristiques globales permettant d'améliorer les algorithmes de diffusion et même ceux de routage. On peut citer les schémas suivant :

- ***Schéma fondé sur le comptage (Counter-Based Scheme)*** : Un mobile ne réémet pas un message s'il l'a déjà entendu plus de C fois. Plus précisément, un mobile attend un certain temps entre la réception et la réémission. Durant cette période, s'il reçoit le nouveau message d'inondation il reprogramme à plus tard sa transmission. Lorsqu'il doit finalement émettre, il regarde s'il a reçu moins de C fois le message d'inondation, auquel cas il émet le message.

- ***Schéma fondé sur la position (Location-Based Scheme)*** : Un mobile ne réémet pas un message si la couverture supplémentaire par l'envoi de son message est inférieure à un seuil A. Encore une fois, un délai d'attente entre la réception et l'émission permet d'optimiser le nombre de voisins qui émettent leur message. Ce modèle obtient les meilleurs résultats, avec une excellente économie de messages (de l'ordre de 40% à 60%), une accessibilité proche de 100% et un faible temps de latence. Il est le plus efficace car il utilise l'information la plus précise possible (une information de position). Enfin, il faut noter que ces dispositifs peuvent nécessiter un dispositif de positionnement, qui n'est pas nécessairement disponible.
- ***Schéma fondé sur les groupes*** : L'idée sous-jacente dans les groupes est celle de hiérarchie parmi les nœuds, en les séparant en plusieurs groupes. Le réseau est alors décomposé en un ensemble de domaines, chacun pouvant contenir une hiérarchie de sous-domaines. Lorsqu'une requête est initiée par un nœud, elle est transmise au chef de groupe, puis peut remonter dans la hiérarchie jusqu'à joindre un chef de niveau supérieur qui pourra la rediriger vers le sous-domaine de destination, pour enfin joindre le ou les correspondants. Au vu de la structuration du réseau, il est logique que certains nœuds aient des statuts spécifiques. Certains sont chargés de maintenir les communications dans le groupe qui leur est affecté, tandis que d'autres se chargent de relayer les messages entre groupes. Une passerelle (Gateway) est un nœud chargé de faire communiquer les groupes entre eux. Elle appartient donc à deux groupes différents et doit être capable de joindre les chefs de chacun de ces groupes. Pour effectuer une diffusion dans cette architecture, la méthode la plus simple proposée est de n'autoriser que les chefs de groupe et les passerelles à réémettre le message de diffusion.

  Cette idée demande une phase de création des groupes. Plus grave, ce modèle est très sensible à la mobilité et le phénomène d'effet en chaîne provoque un encombrement réseau important dû à la recréation des groupes et la perte des messages d'inondation. Le modèle de grappes, avec sa structure hiérarchique sous-jacente, se trouve handicapé par la nécessité de reconstruction des groupes lors des changements topologiques.
- ***Schéma fondé sur les ensembles dominants*** : Un ensemble dominant (dominating set) est une notion de la théorie de graphes. Un sous-ensemble D de V est dit dominant si et seulement si tout nœud de V est soit dans D soit voisin d'un nœud de D. Plus précisément, l'ensemble D est dominant si et seulement si :

$$\forall i \in V \quad i \in D \cap (\exists j \in D \qquad i \in N(j))$$

Dans un réseau de capteur, l'ensemble dominant doit être connexe pour assurer une diffusion complète. Un graphe est dit connexe si tous les nœuds sont joignables, c'est-à-dire qu'il existe toujours un chemin constitué d'arcs reliant deux nœuds du graphe. Cette propriété est très importante dans le cas des ensembles dominants, car elle garantit que chaque nœud de l'ensemble dominant peut joindre n'importe quel autre dans ce même ensemble. L'algorithme de diffusion est alors très simple : seuls les nœuds de l'ensemble dominant réémettent le message. Chaque nœud de l'ensemble dominant couvre alors l'ensemble de ses voisins, garantissant ainsi une couverture complète du réseau. Un nœud u est dit « couvert » par un sous-ensemble S de son voisinage si et seulement si les 3 conditions suivantes sont respectées [Dai et Wu]:

- l'ensemble S est connecté,
- tout voisin de u est le voisin d'au moins un nœud de S,
- tous les nœuds de S ont un identifiant supérieur à celui de u.

Un nœud appartient à un ensemble dominant si et seulement s'il n'existe pas de sous-ensemble qui le couvre complètement.

- ***Schéma fondé sur l'élimination des voisins*** : Par la nature même de la diffusion, chaque nœud reçoit les messages émis par l'ensemble des mobiles, même si ceux-ci ne lui étaient pas destinés. Il peut donc prendre conscience des informations que chaque nœud a reçues. Pour garantir une accessibilité quasi-parfaite, chaque mobile peut ajouter à son message d'inondation sa liste de voisins. Ainsi, chaque nœud, en comparant sa liste de voisinage à celle contenue dans le message d'inondation, peut déduire quels sont les mobiles qui ont déjà reçu le message. Au bout d'un certain temps, lorsque le message d'inondation a été diffusé dans tout le réseau (ou tout au moins, dans l'ensemble du voisinage), chaque nœud regarde la liste de ses voisins. Si au moins un des nœuds n'a pas été contacté par le message d'inondation, il peut réémettre son message.

  Ce procédé n'est pas parfait. Par exemple, si un des voisins (B) d'un nœud (A) a été contacté par un mobile éloigné et qu'il ne réémet pas le message d'inondation, alors le mobile A peut croire que ce mobile n'a pas été joint et donc se forcer à réémettre son message d'inondation. Ce surcoût peut dépendre de la densité du réseau et du protocole de diffusion utilisé.

Dans le cas de protocoles d'inondation intégrés à des protocoles de routage, il est difficile d'envisager des protocoles ayant des paramètres très dépendants du réseau (densité de nœuds, topologie, sans perdre en généricité). C'est pour cela que les algorithmes de diffusion proposés dans un contexte purement de réseau ad hoc ne sont guère plus sophistiqués que l'inondation aveugle. Ce qui induit à des conséquences, et le travail présenté propose une nouvelle idée basée sur l'utilisation d'algorithmes des approches probabilistes.

III.6 Counter Based Efficient Flooding (CBEF)

Le problème c'est que chaque nœud diffuse le paquet a tout ces voisins ce qui va créer une surcharge de la bande passante « inondation aveugle » et va aussi gaspiller énormément d'énergie au niveau des nœuds vu que tous ces derniers participeront a cette inondation jusqu'à atteindre la destination désirée, ce qui risque de les faire disparaitre du réseau au cas où leurs batteries s'éteindront.

Ainsi, on pourra dire qu'avec ce type d'inondation, plus le nombre des nœuds est important dans le réseau lors de la procédure de recherche de route, plus y aura des risques d'affaiblissement des batteries avec une surcharge importante au niveau de la bande passante.

Afin de faire face aux problèmes créer par l'inondation aveugle (surcharge de bande passante, consommation d'énergie,…) ils ont proposés une amélioration concernant la diffusion et cela en ajoutant quelques conditions que doit vérifier chaque nœud du réseau lors de la réception d'un paquet RREQ par son voisin afin de réduire les problèmes soulevé par l'inondation aveugle et tout cela en économisant l'énergie des batteries et en libérant la surcharge de la bande passante.

- ✓ **Comparaison entre inondation aveugle et inondation probabiliste**

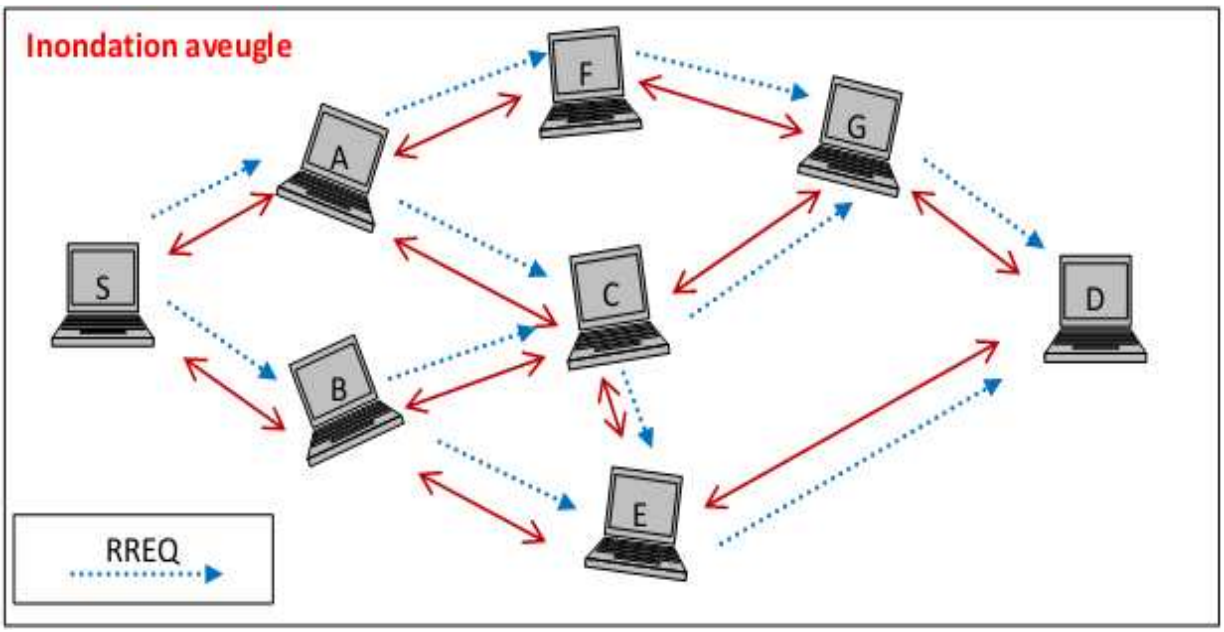

Figure III.1 La procédure de diffusion de RREQ

Depuis le schéma précédent, on voit bien que le nœud S inonde le réseau en envoyant les paquets à tous ces voisins qui sont dans sa portée de communication, et qui eux aussi l'envoient à leur tour jusqu'à arriver au nœud (D) qui est la cible de cet envoi, et comme résultat de la procédure de diffusion de route, certes on aura le chemin qui mène depuis la source vers la destination, mais en conséquence, on aura aussi du gaspillage d'énergie et de la bande passante surchargée.

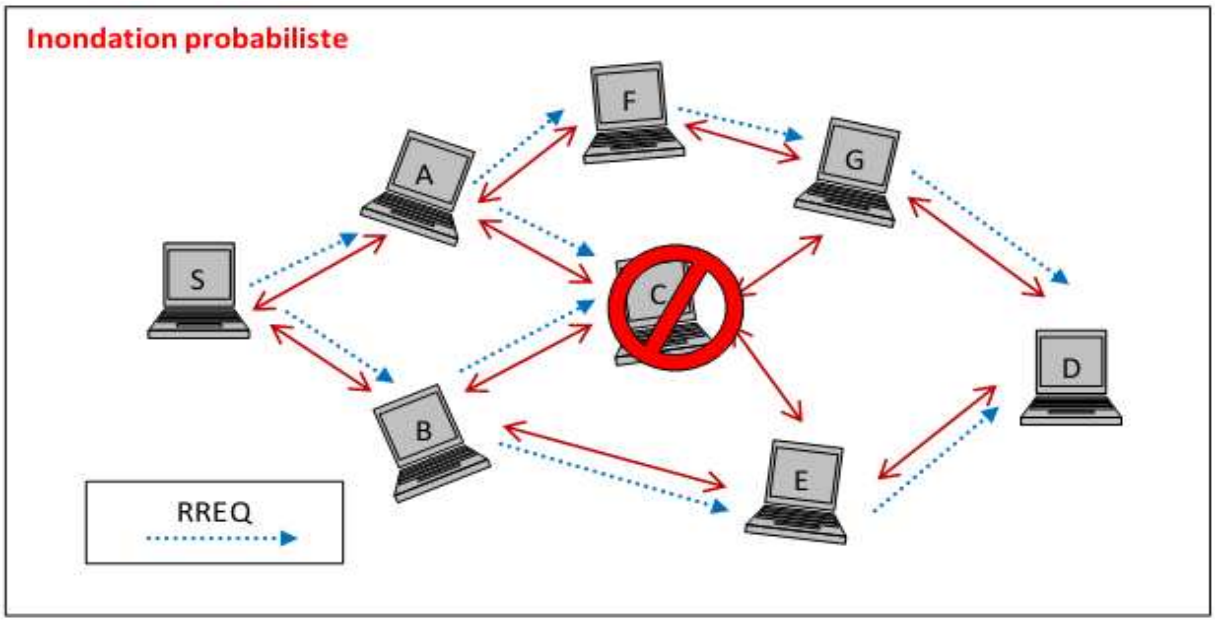

Figure III.2 La procédure de diffusion de RREQ de leur proposition

Dans cet exemple, le nœud C se met en silence et n'envoie pas de paquet (RREQ) à ses voisins et laisse les nœuds E et G le diffuser et tout cela en vérifiant les deux propriétés suivantes :

1. Le nombre de ses voisins : si le nombre de ses voisins est important, il se réduit au silence, et peut ne pas diffuser et laisser ses voisins le faire.

2. Le nombre de silence : si le nœud n'effectue aucune diffusion et qu'il atteint le nombre limite de silences : il doit obligatoirement le diffuser. L'inondation probabiliste sur le comptage représente quelques avantages par rapport à l'inondation aveugle notamment en termes d'énergie et celui de la surcharge de la bande passante.

Le but principal de leur proposition est d'essayer de mettre fin aux problèmes issus de l'inondation aveugle et cela en remplaçant l'inondation aveugle par l'inondation probabiliste ou sur le comptage, et cela s'effectue en améliorant la diffusion en lui ajoutant quelques techniques de vérification lors de la réception d'un paquet (RREQ)avant son envoi aux voisins afin que les nœuds puissent garder leur énergie le plus longtemps possible et sans oublier de dégager la bande passante le plus de fois possible, pour que telles conditions soient réalisables.

III.7 Conclusion

Dans ce chapitre, on a étudié l'inondation aveugle et la probabiliste, pour amélioré cette diffusion en remplaçant l'une par l'autre.

Dans le chapitre suivant, nous implémenterons la méthode d'optimisé la diffusion CBEF pour les réseaux de capteurs sans fil.

IV.1 Introduction

TinyOS est un système d'exploitation open-source conçu pour des réseaux de capteurs sans fil. Il respecte une architecture basée sur une association de composants, réduisant la taille du code nécessaire à sa mise en place. Cela s'inscrit dans le respect des contraintes de mémoires qu'observent les réseaux de capteurs.

Pour autant, la bibliothèque de composant de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. L'ensemble de ces composants peut être utilisé tel quel, il peut aussi être adapté à une application précise.

En s'appuyant sur un fonctionnement événementiel, TinyOS propose à l'utilisateur une gestion très précise de la consommation du capteur et permet de mieux s'adapter à la nature aléatoire de la communication sans fil entre interfaces physiques.

IV.2 Propriétés de la plateforme TinyOS

TinyOS a été conçu pour la programmation des réseaux de capteurs et il est caractérisé par :

- Disponibilité et sources : TinyOS est un système principalement développé et soutenu par l'université américaine de Berkeley, qui le propose en téléchargement sous la licence BSD et en assure le suivi;
- Event-driven : Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des évènements qui se produisent instantanément. Ainsi, l'activation de tâches, leur interruption ou encore la mise en veille du capteur s'effectue à l'apparition d'évènements, ceux-ci ayant la plus forte priorité. Ce fonctionnement basé sur les évènements (Event-driven) s'oppose au fonctionnement dit temporel (time-driven) où les actions du système sont gérées par une horloge donnée ;
- Langage : TinyOS a été programmé en langage NesC que nous allons détailler plus tard;
- Préemptif : Le caractère préemptif d'un système d'exploitation précise si celui-ci permet l'interruption d'une tâche en cours. TinyOS ne gère pas ce mécanisme de préemption entre les tâches mais donne la priorité aux interruptions matérielles. Ainsi, les tâches entre-elles ne s'interrompent pas mais une interruption (sous forme d'un évènement) peut stopper l'exécution d'une tâche ;
- Temps réel : Lorsqu'un système est dit « temps réel » celui-ci gère des niveaux de priorité dans ses tâches permettant de respecter des échéances données par son environnement. Dans le cas d'un système strict, aucune échéance ne tolère de dépassement contrairement à un système temps réel mou. TinyOS se situe au-delà de ce second type car il n'est pas prévu pour avoir un fonctionnement temps réel ;

- Consommation : TinyOS a été conçu pour réduire au maximum la consommation en énergie d'un nœud capteur. Ainsi, lorsqu'aucune tâche n'est active, il se met automatiquement en mode veille.

Le tableau ci-dessous résume ses propriétés :

Propriété	Valeur pour TinyOS
Type	Event-driven
Disponibilité	Open source
Langage	NesC
Préemptif	Non
Temps réel	Non
Source	Fournies

Tableau IV.1 : Propriété de TinyOS

IV.3 Gestion Allocation de la mémoire

Il est important de préciser de quelle façon un système d'exploitation aborde la gestion de la mémoire. C'est encore plus significatif lorsque ce système travaille dans un espace restreint.

TinyOS a une empreinte mémoire très faible puis qu'il ne prend que 300 à 400 octets dans le cadre d'une distribution minimale. En plus de cela, il est nécessaire d'avoir 4 Ko de mémoire libre qui se répartissent suivant le schéma ci-contre :

- La pile : sert de mémoire temporaire au fonctionnement du système notamment pour l'empilement et le dépilement des variables locales ;
- Les variables globales : réservent un espace mémoire pour le stockage de valeurs pouvant être accessible de puis des applications différentes ;
- La mémoire libre : pour le reste du stockage temporaire.

La gestion de la mémoire possède de plus quelques propriétés. Ainsi, il n'y a pas d'allocation dynamique de mémoire et pas de pointeurs de fonctions. Bien sur cela simplifie grandement l'implémentation. Par ailleurs, il n'existe pas de mécanisme de protection de la mémoire sous TinyOS ce qui rend le système particulièrement vulnérable aux crash et corruptions de la mémoire.

IV.4 Structure logicielle

Le système d'exploitation TinyOS s'appuie sur le langage NesC. Celui-ci propose une architecture basée sur des composants, permettant de réduire considérablement la taille mémoire du système et de ses applications. Chaque composant correspond à un élément matériel (LEDs, timer, ADC . . .) et peut être réutilisé dans différentes applications. Ces applications sont des ensembles de composants associés dans un but précis. Les composants peuvent être des concepts abstraits ou bien des interfaces logicielles aux entrées-sorties matérielles de la cible étudiée (carte ou dispositif électronique). L'implémentation de composants s'effectue en déclarant des tâches, des commandes ou des évènements.

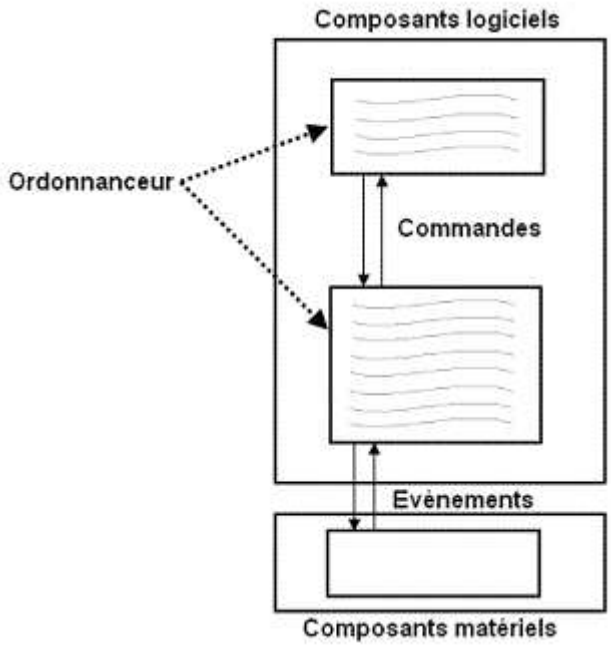

Figure IV.1 Interactions internes au système TinyOS

Les tâches sont utilisées pour effectuer la plupart des blocs d'instruction d'une application. A l'appel d'une tache, celle-ci va prendre place dans une file d'attente de type FIFO (First In First Out) pour y être exécutée. Comme nous l'avons vu, il n'y a pas de mécanisme de préemption entre les tâches et une tache activée s'exécute en entier. Ce mode de fonctionnement permet de bannir les opérations pouvant bloquer le système (inter-blocage, famine, . . .). Par ailleurs, lorsque la file d'attente des taches est vide, le système d'exploitation met en veille le dispositif jusqu'au lancement de la prochaine interruption (on retrouve le fonctionnement event-driven).

Les commandes correspondent à l'exécution d'une fonctionnalité précise dans un autre composant.

Les évènements sont prioritaires par rapport aux tâches et peuvent interrompre la tache en cours d'exécution. Ils permettent de faire le lien entre les interruptions matérielles (pression d'un bouton, changement d'état d'une entrée, . . .) et les couches logicielles que constituent les taches. Dans la pratique, NesC permet de déclarer 2 types de composants : les modules et les configurations:

- Les modules constituent les briques élémentaires de code et implémentent une ou plusieurs interfaces.
- Une application peut faire appel à des fichiers de configuration pour regrouper les fonctionnalités des modules. Un fichier top-level configuration permet de faire le lien entre tous les composants.
- Les interfaces sont des fichiers décrivant les commandes et évènements proposés par le composant qui les implémente. L'utilisation des mots clefs « Use » et « Provide » au début d'un composant permet de savoir

respectivement si celui-ci fait appel à une fonction de l'interface ou redéfinit son code.

IV.5 L'ordonnanceur TinyOS

Nous allons détailler le fonctionnement précis de l'ordonnanceur TinyOS qui est au cœur de la gestion des tâches et des évènements du système. Le choix d'un ordonnanceur déterminera le fonctionnement global du système et le dotera de propriétés précises telles que la capacité à fonctionner en temps réel.

L'ordonnanceur TinyOS c'est :

- 2 niveaux de priorité (bas pour les tâches, haut pour les évènements)
- 1 file d'attente FIFO (disposant d'une capacité de 7)

Par ailleurs, entre les tâches, un niveau de priorité est défini permettant de classer les tâches, tout en respectant la priorité des interruptions (ou évènements). Lors de l'arrivée d'une nouvelle tache, celle-ci sera placée dans la file d'attente en fonction de sa priorité (plus elle est grande, plus le placement est proche de la sortie). Dans le cas ou la file d'attente est pleine, la tâche dont la priorité est la plus faible est enlevée de la FIFO.

IV.6 Package TinyOS

TinyOS est prévu pour fonctionner sur une multitude de plateformes, disponibles dès l'installation. En effet, TinyOS peut être installé à partir d'un environnement Windows (2000 et XP) ou bien GNU/Linux (RedHat essentiellement, mais d'autres distributions sont également possibles). Deux principales versions de TinyOS sont disponibles : la version stable (v. 1.1.0) et la version actuellement en cours de tests (v.1.1.15); la première présente moins de risques mais est nettement moins récente.

- Windows : un guide propose l'installation de tous les principaux outils nécessaires au bon fonctionnement du système, notamment Cygwin (couche d'émulation de l'API Linux) qui permet d'avoir une interface Unix sous Windows. Le JDK Java 1.4 de Sun est nécessaire afin d'effectuer la procédure d'installation.
- GNU/Linux : des packages RPM sont proposés au téléchargement, un guide explique la marche à suivre. Les distributions Linux ayant un autre gestionnaire de paquet peuvent utiliser un programme (comme « Alien ») pour installer les packages, ou compiler directement à partir des sources. Le JDK de IBM est nécessaire.

IV.7 Cibles possibles pour TinyOS

Il existe de nombreuses cibles possibles pour ce système d'exploitation embarqué. Malgré leurs différences, elles respectent toutes globalement la même architecture basée sur un noyau central au tour duquel s'articule les différentes

interfaces d'entrée-sortie, de communication et d'alimentation. Voici un schéma représentant cette architecture:

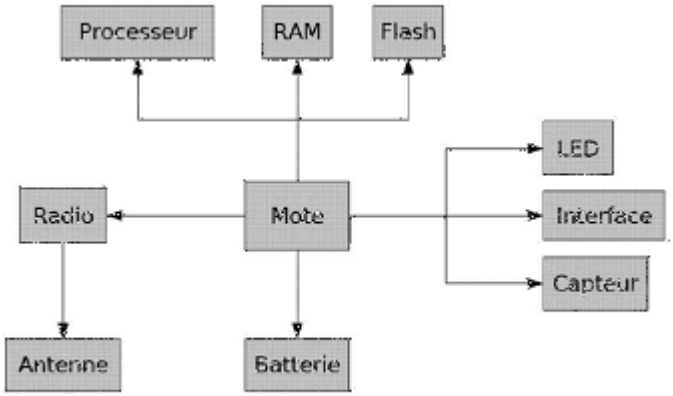

Figure IV.2 Architecture générale des cibles utilisant TinyOS

Mote, processeur, RAM et Flash : On appelle généralement Mote la carte physique utilisant TinyOS pour fonctionner. Celle-ci a pour cœur le bloc constitué du processeur et des mémoires RAM et Flash. Cet ensemble est à la base du calcul binaire et du stockage, à la fois temporaire pour les données et définitif pour le système TinyOS.

Radio et antenne : TinyOS est prévu pour mettre en place des réseaux sans fils, les équipements étudiés sont donc généralement équipés d'une radio ainsi que d'une antenne afin de se connecter à la couche physique que constitue les émissions hertziennes.

LED, interface, capteur : TinyOS est prévu pour mettre en place des réseaux de capteurs, on retrouve donc des équipement bardés de différents types de détecteurs et autres entrées.

Batterie : Comme tout dispositif embarqué, ceux utilisant TinyOS sont pourvus d'une alimentation autonome telle qu'une batterie.

IV.8 Counter Based Efficient Flooding for Wireless Sensor Network (CBEF-WSN)

Voila un diagramme d'activité qui illustre les différents points ajoutés au niveau de la couche mac à une proposition pour le fonctionnement du CBEF - WSN.

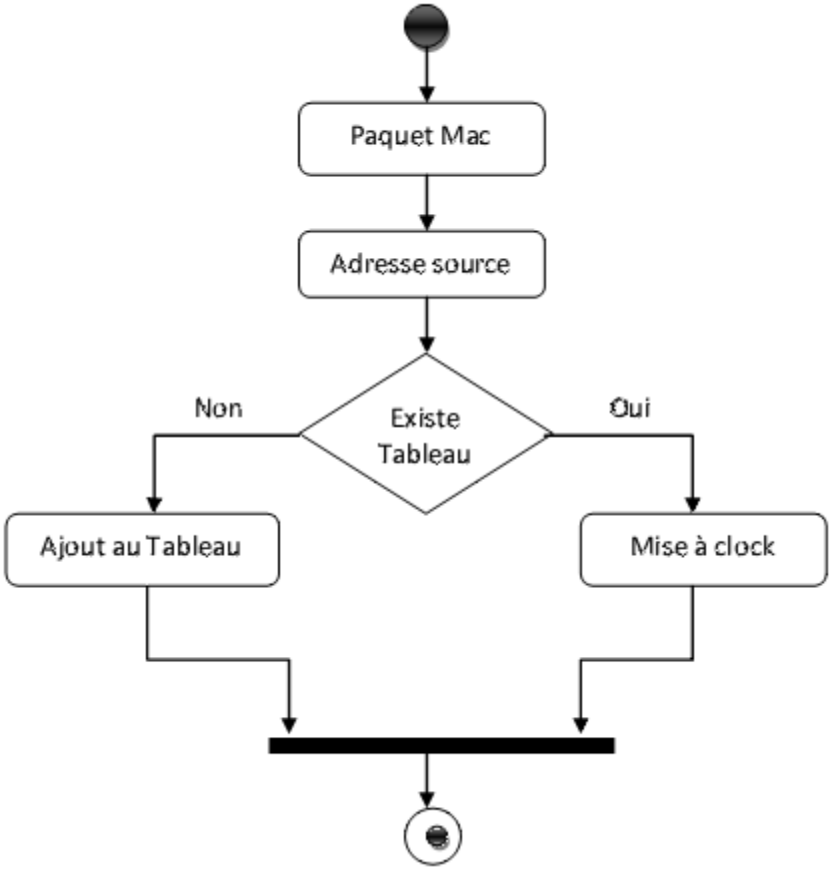

Figure IV.3 Diagramme d'activité de la couche mac.

Ensuite le nœud peut savoir le nombre de ces voisins s'il compte le nombre des nœuds enregistrés dans l'enregistrement et transmet ce nombre à la couche supérieure pour calculer le rapport du nombre de voisins sur le nombre de RREQ envoyé par l'ensemble des voisins.

Un mobile quand il reçoit une RREQ ne réémet pas le message directement il attend un certain temps entre la réception et la réémission. Durant cette période, s'il reçoit un nouveau paquet d'inondation, il reprogramme à plus tard sa transmission et incrémente le nombre des RREQ reçu pour la même RREQ (i.e. la même source et destination ainsi que le même nombre de séquences enregistrées dans le paquet de RREQ). Il s'agit la d'un autre enregistrement qui contient un champ pour sauvegarder le paquet, un champ pour compter le nombre des RREQ reçus pour ce paquet et un autre champ pour identifier la source et la destination du paquet.

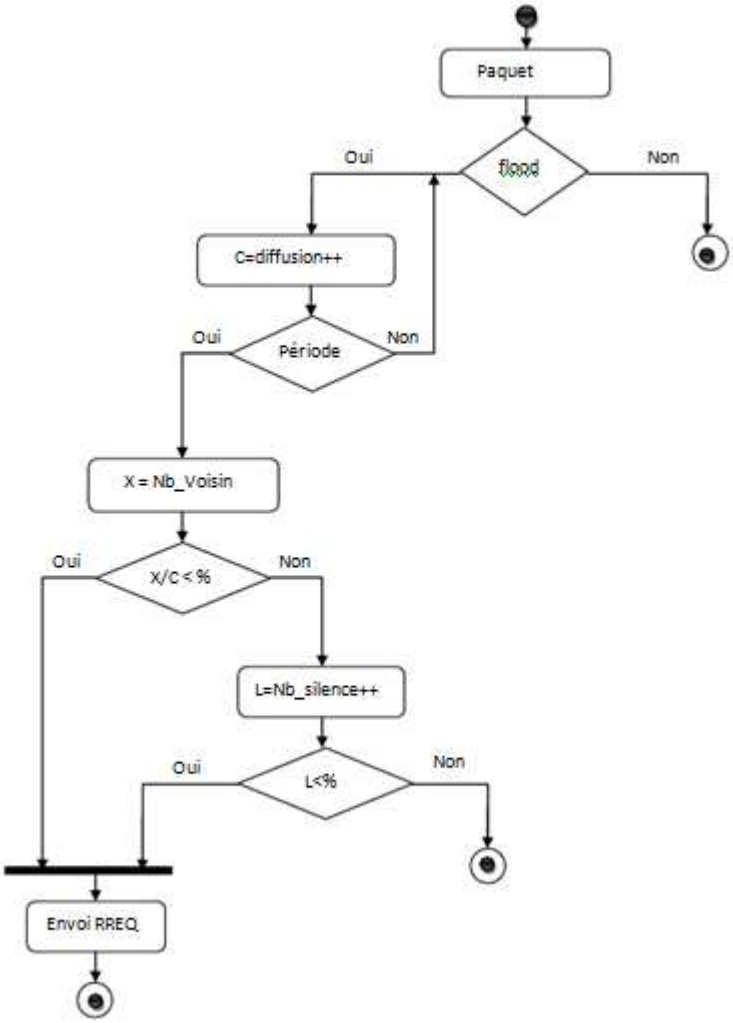

Figure IV.4 Diagramme d'activité de flood au niveau du paquet

Lorsqu'il doit finalement émettre, il regarde le rapport de nombre de voisins sur le nombre des RREQ reçus, s'il est inférieur à un certain seuil fixé à la simulation, il émet le message et initialise le compteur des RREQ reçus à zéro.

On peut améliorer notre proposition par un autre compteur qui calcule le nombre de silence de nœud afin de le comparer à un autre seuil. Si ce compteur dépasse le seuil il doit obligatoirement réémettre la RREQ pour qu'il ne soit pas mis trop au silence et prouver son existence pour pouvoir contribuer sur à la transmission et les échanges des messages dans le réseau.

Un autre point aussi très important dans notre proposition, si un nœud reçoit une RREQ et contient dans son cache le chemin vers la destination, il ne doit pas se mettre en silence et compte sur ses voisins pour la retransmission du RREQ même si le rapport du nombre de voisins sur le nombre des RREQ reçus est inferieur au seuil, il crée un paquet de type Route Reply et ajoute ce chemin à la Route Reply et envoie le paquet vers la source.

Voila un diagramme d'activité qui illustre les différents points ajoutés au niveau du paquet pour le fonctionnement du CBEF - WSN.

IV.9 Capteur utilisé

Au cours de nos expériences, nous utilisons des capteurs sans fil de type MTM-CM5000-MSP, de « MAXFOR TECHNOLOGY ING » .Chaque module radio est contrôlé par un microcontrôleur MSP430 fabriqué par Texas instruments qui est équipé de 10 Kilobits de mémoire vive (RAM), 48 Kilobits de mémoire flash, et fonctionne à 2.4 GHZ. De plus, chaque senseur est muni de capteurs pour détecter la lumière, la température et l'humidité. Pour la communication radio, les senseurs utilisent une puce RF émetteur-récepteur CC2420. Cette puce est conformée au standard IEEE 802.15.4 qui sert d'intermédiaire entre le MSP430 et l'antenne et qui permet aussi de mesurer la puissance du signal reçu.

La figure suivante illustre des sur ce type des capteurs :

Figure IV.5 Spécifications de capteur **MTM-CM5000MSP**.

IV.10 Implémentation de CBEF

TinyOS a été utilisé comme environnement de développement dans cette recherche. Les librairies et les applications du système TinyOS sont écrites en NesC, une version de C qui a été conçue pour programmer les systèmes embarqués. Dans NesC, les programmes sont construits à partir des composants qui sont connectés ensemble pour former le programme entier.

IV.10.1 Structure.h

```
nx_uint16_t type;
nx_uint16_tsrc;
nx_uint16_t inter;
nx_uint16_t dst;
nx_uint16_thopes;
nx_uint16_tseq;
nx_uint16_t data;
```

IV.10.2 Flood.nc

```
#include "structure.h"
moduleTestAMC {
uses {
interface Packet;
interfaceAMPacket;
interfaceLeds;
interface Boot;
interface Receive;
interfaceAMSend;
interface Timer<TMilli> as MilliTimer;
interface Timer<TMilli> as Timer0;
interfaceSplitControl as AMControl;
  }
}
implementation {
message_t packet;
bool locked;
uint16_tseq = 0;
uint16_t flood=1;
uint16_t data=2;
uint16_t voisins[20];
uint16_t nbr_vois=0;
uint16_t diffusion=0;
uint16_t seuil=0;
uint16_t packet_seq=0;
uint16_t packet_hopes=0;
uint16_t packet_src=0;
uint16_t id=2;
event void Boot.booted() {
callAMControl.start();
}
//-----------------------------------------------------------------
void flooding() {
```

```
MSG_t* msg = (MSG_t*)callPacket.getPayload(&packet, sizeof(MSG_t));
msg->type=flood;
seq++;
msg->seq=seq;
msg->data=23;
msg->hopes=0;
msg->src=id;
msg->dst=AM_BROADCAST_ADDR;
msg->inter=id;
if (locked) {
return;
     }
else    if    (call    AMSend.send(AM_BROADCAST_ADDR,    &packet,
sizeof(MSG_t)) == SUCCESS) {
call Leds.led0On();
locked = TRUE;
}
return;
 }
//-----------------------------------------------------------------
voidvoisinage(uint16_t addr) {
uint8_t i;
for(i=0;i<nbr_vois;i++){
if(voisins[i]==addr) return;
}
voisins[nbr_vois]=addr;
nbr_vois++;
}
//-----------------------------------------------------------------
event void MilliTimer.fired() {
flooding();
}
//-----------------------------------------------------------------
eventmessage_t* Receive.receive(message_t* bufPtr,
                void* payload, uint8_t len) {
```

```
MSG_t* msg = (MSG_t*)payload;
voisinage(msg->inter);//appel a la fonction voisinage pour compter le nombre de voisins
if(msg->type==flood){
if(msg->seq>seq){
seq=msg->seq;
msg->hopes++;
diffusion=1;
packet_seq=msg->seq;
packet_hopes=msg->hopes;
packet_src= msg->src;
call Timer0.startOneShot( 250 );
     }
else diffusion++;
}
returnbufPtr;
  }
//-----------------------------------------------------------------
event void AMSend.sendDone(message_t* bufPtr, error_t error) {
if (&packet == bufPtr) {
locked = FALSE;
call Leds.led0Off();
    }
  }
//-----------------------------------------------------------------
event void Timer0.fired()
  {
seuil=2;
if (diffusion <seuil)
{
MSG_t* msg = (MSG_t*)callPacket.getPayload(&packet, sizeof(MSG_t));
     msg->type=flood;
     msg->seq=packet_seq;
     msg->data=0;
     msg->hopes=packet_hopes;
```

```
      msg->src=packet_src;
      msg->dst=AM_BROADCAST_ADDR;
      msg->inter=id;
      if (locked) {
      return;
             }
      else   if   (call   AMSend.send(AM_BROADCAST_ADDR,   &packet,
sizeof(MSG_t)) == SUCCESS) {
      call Leds.led1Toggle();
      locked = TRUE;
      }
   }
 }
//-------------------------------------------------------------------
event void AMControl.startDone(error_t err) {
callMilliTimer.startPeriodic(5000);//  ce  timer  permet  d'envoyer
periodiquement un message
  }
//-------------------------------------------------------------------
event void AMControl.stopDone(error_t err) {
}
}
```

IV.10.3 Explication de programme

- La fonction module *TestAMC* :

La première partie du programme signifie qu'il s'agit d'un module appelé *TestAMC*, et déclaré les interfaces qu'il utilise. Le module *TestAMC* utilise les interfaces:

Packet : forme de message diffusé.

Leds : permettent de suivre le déroulement du programme en allumant les Leds du capteur à des moments pertinents.

Receive : permet de recevoir les messages recueillis via la liaison sans fil.

AMSend: permet d'envoyer les messages via une liaison sans fil.

Dans ce programme, on a utilisé deux « timer » pour déclencher les événements périodiques:

MilliTimer : Ce timer permet d'envoyer périodiquement un message de position aux restes membres de groupe, il est égale 5000 ms.

Timer0 : Ce timer est programmé à 250 ms.

- La fonction *implémentation* :

Dans cette fonction on initialise les paramètres du structure de message par exemple seq initialiser à 0 ; flood=1; numéro data=2; tableaux de 20 élément; nbr_vois=0; diffusion=0; seuil=0 et chaque capteur a une identification « id ».

- La fonction *flooding* :

On commence un nouveau flood (diffusion), en fixant les paramètres du message : le type à flood, on incrémente seq , data à 23, hopes à 0, src à id du capteur(adresse), et en suite on envoi le message.

- La fonction *voisinage* :

Si l'adresse de voisin n'existe pas dans le tableau on l'ajoute et ensuite on incrémente le nombre de voisin.

- La fonction *receive* :

Elle fait appel à la fonction *voisinage* pour vérifier est ce que l'émetteur se trouve dans la table voisins.

Après elle teste est ce que le message est de type flood, si oui, elle teste le numéro de séquence du message est supérieur au dernier numéro de séquence enregistré, si oui – c.à.d. une nouvelle diffusion- elle remplace le numéro de séquence du programme par celui du message, augmente la variable hopes du message, met variable diffusion à 1(réinitialisation des variables) et on d'éclanche le timer0 qui est responsable sur le temps avant la diffusion ; Si non –c.à.d. une ancienne diffusion- elle incrémente la variable diffusion

- Fonction *AMSend.sendDone :*

Pour éteindre le voyant du capteur si le packet est envoyé

- La fonction *Timer0* :

Ce Timer est d'éclanche chaque 250ms, son rôle et de vérifier si diffusion <seuil, si oui il envoi le message de diffusion reçu ; si non il ne fait rien (silence)

- *La fonction AMControl* :

Elle fait appel à *MilliTimer.startPeriodic(5000)* : est le temps qui permet d'envoyer périodiquement un message avec une période de 5 seconds (pour éviter le silence absolu)

IV.10.4 Installation de l'application

- **Capteurs :**

Les commandes pour installer l'application dans un capteur

Cd /opt/tinyos-2.x/apps/flood

Make telosb

Make telosb reinstall bsl, /dev/ttyUSB0

Les trois lignes de commandes montrent respectivement, la méthode d'acces au répertoire (flood) qui contient l'application, la compilation de cette application et l'installation de cette dernière dans le capteur.

- **Station de base :**

Les commandes pour installer le programme de la station de base dans un capteur

Cd /opt/tinyos-2.x/apps/BaseStation

Make telosb

Make telosb reinstall bsl, /dev/ttyUSB0

L'outil Listen Java capture les paquets reçus et il affiche le contenu en binaire. La commande à exécuter pour lancer le programme :

$ javanet.tinyos.tools.Listen -comm serial@/dev/ttyUSB0:telosb

Le format général des messages de l'application BlinkToRadioC est donc(en ignorant le premier octet00):

- Destination address (2 bytes)
- Active Message handler ID (1 byte)
- Group ID (1 byte)
- Message length (1 byte)
- Payload (up to 29 bytes):
    - source mote ID (2 bytes)
    - samplecounter (2 bytes)
    - ADC channel (2 bytes)
    - ADC data readings (10 readings of 2 bytes each): dans le cas de notre application ce champ contient: flood, source, destination, sequence, hopes, inter, data (cette structure est définie dans le fichier structure.h)

Ainsi, nous pouvons interpréter le paquet de données comme suit:

dest addr	handlerID	groupID	msg len	source addr	counter	channel	readings
7e 00	0a	7d	1a	01 00	14 00	01 00	96 03 97 03 97 03 98 03 97 03 96 03 97 03 96 03 96 03 96 03

Après l'exécution du programme java 'Listen' on obtient la fenêtre suivante :

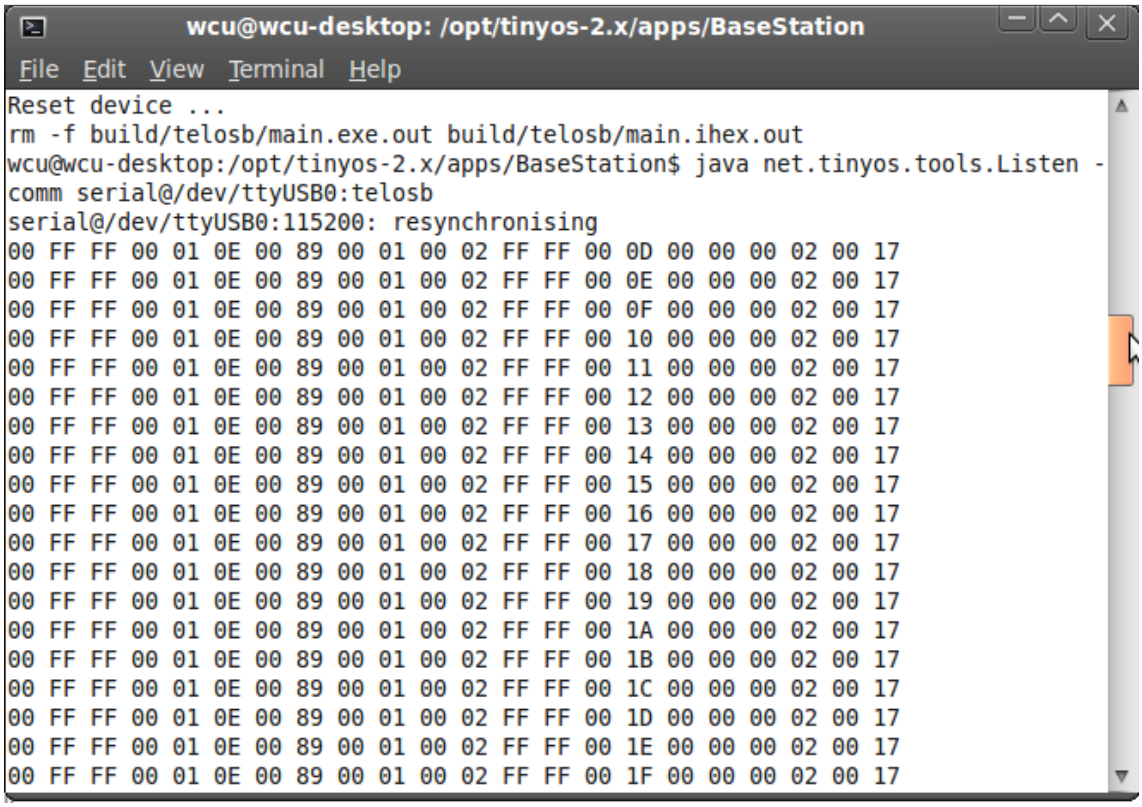

Figure IV.6 : Station de base (hexadécimale)

Les données collectées sont représentés en hexadécimale, et pour rendre les résultats plus lisibles, on utilise le programme amélioré qui permet de rendre les résultats plus lisibles (**Figure IV.7**).

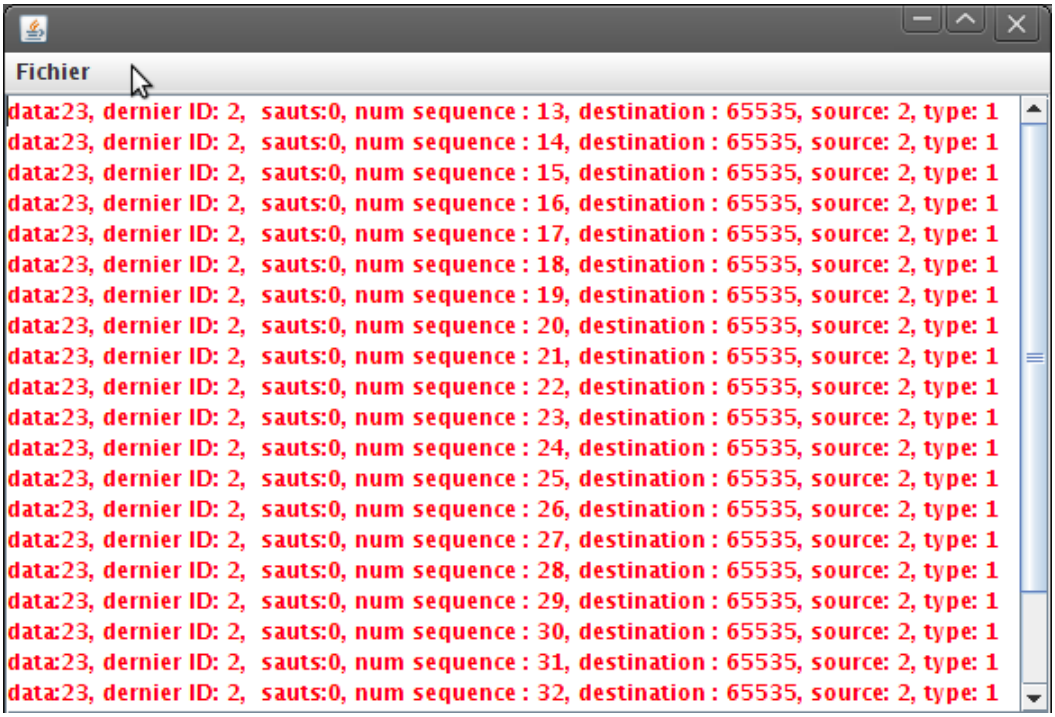

Figure IV.7 : Station de base (interface améliorée)

Data : Donnée.

Derniere ID : dernier nœud avant d'atteindre la station de base.

Sauts : Nombre de sauts pour arriver à la station de base

Num sequence : définit le numéro de séquence utilisé par les capteurs afin d'éviter qu'un message soit traiter deux fois.

Destination : l'adresse de destination (dans le notre cas c'est l'adresse de diffusion).

Source : L'adresse de l'émetteur

Type : Flood (La diffusion).

IV.11 Conclusion

Dans ce chapitre, on a présenté les outils logiciels et matériels ainsi que la démarche à suivre pour réaliser une application d'amélioration de la diffusion

Cette implémentation pour but d'optimiser les diffusions dans les réseaux de capteurs, ce qui permet de minimiser la charge dans les réseaux, et surtout elle permet d'économiser l'énergie pour les capteurs qui sont des dispositifs à faible ressources.

Elle exige des outils bien particuliers qui sont développés pour exploiter les systèmes à ressources limitées tels qu'un système d'exploitation léger « TinyOs », un langage orienté composant « NesC ».

Les réseaux de capteurs sont des réseaux formés d'un grand nombre de nœuds capteurs qui se collabore entre eux pour fournir un service bien déterminé.

Cependant, il faut savoir que la caractéristique principale des RCSFs réside dans l'absence d'une infrastructure centralisée et qui fait du routage un problème très compliqué. Des grands travaux ont été réalisés et des études sont faites pour résoudre ce problème.

Comme perspective de ce travail on peut proposer le teste d'application à grande échelle pour valider la méthode proposée et aussi fixer les paramètres optimale pour la décision d'émission.

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

ACQUIRE	Active Query Forwarding in Sensor Networks
ADV	ADVertise
AODV	Ad hoc On demand Distance VectorRouting
APTEEN	Adaptive Periodic TEEN
CBEF	Counter Based Efficient Flooding
CHs	Clusterheads
DARPA	Defense Advanced Research Projects Agency
DD	Directed Diffusion
DSDV	Destination Sequenced Distance Vector
DSN	Distributed Sensor Network
DSR	Dynamic Source Routing
EAR	Eavesdrop And Register
FIFO	First In First Out
GEAR	Geographic and Energy Aware Routing
IEEE	Institute of Electrical and Electronics Engineers
IrDA	Infrared Data Association
JBREWS	Joint Biological Remote Early Warning
LEACH	Low-Energy Adaptive Clustering Hierarchy
MEMS	Micro Electro-Mechanical Systems
NES	Mécanisme d'élimination de voisins
QdS	Qualité de Service
Rc	Rayon de Communication
RCSF	Réseau de Capteur Sans Fil
REQ	REQuest
RR	Rumor Routing
Rs	Rayon de Sensation
SAR	Sequential Assignment Routing
SMACS	Self-organizing Medium Access Control for Sensor networks
SMP	Sensor Management Protocol
SQL	Structured Query Language

LISTE DES ABRÉVIATIONS

SPIN	Sensor Protocols for Information via Negotiation
TADAP	Task Assignement and Data Advertisement Protocol
TCP	Transmission Control Protocol
TEEN	Threshold sensitive Energy Efficient sensor Network protocol
UDP	User Datagram Protocol Like
UWB	Ultra Wide Band
WATS	Wide Area Tracking System
WSN	Wireless Sensor Network
ZRP	Zone Routing Protocol

[1] Berrachedi Amel,Diarbakirli Amina «Sécurisation du protocole de routage hiérarchique LEACH dans les réseaux de capteurs sans fil», 2008/2009 .

[2] Séverine Sentilles, «Achitecture logitiellle pour capteurs sans-fil en réseau», Master TI 2e année, mälardalen University, Sweden, Janvier-Juin 2006.

[3] Yacine Challal «Réseaux de capteurs sans fils», Supports de cours, Systèmes Intelligents pour le transport, Université de Technologie de Compiègne, France,17 Novembre 2008.

[4] Samira Allam «Approche multi agents pour contrôler l'inondation dans un réseau de capteurs», Projet de fin d'étude, Ecole nationale Supérieure d'Informatique ESI,Algérie,2008/2009.

[5] Antoine Gallais, François Ingelrest, Jean Carle, David Simplot-Ryl. Maintien de la couverture de surface dans les réseaux de capteurs avec une couche physique non idéale. CFIP, Colloque Francophone sur l'Ingénierie des Protocoles. 2006.

[6] Gallais, Antoine. Ordonnancement d'activité dans les réseaux de capteurs : l'exemple de la couverture de surface. Université des sciences et technologies de Lille. 2007. Rapport de thèse.

[7] Imrich Chlamtac, Iacopo Carreras et Hagen Woesner "From internets to bionets: biogical kinetic servive oriented networks". The case study of Bionetic Sensor Networks. CREAT-NET Research Consortium, Trento, Italy 2005.

[8] Khelladi, L. et N. Badache. «Réseaux de capteurs : état de l'art.» Rapport de recherche. 2004.

[9] Hakima CHAOUCHI et Maryline LAURENT- MAKNAVICIUS ; Rapport Technique «Les réseaux sans fil et la sécurité» ; l'Institut national de télécommunications ; 10-2007

[10] C. Karlof, D. Wagner, «Secure routing in wireless sensor networks: Attacks and countermeasures», in Proceedings of First IEEE International Workshop on Sensor Network Protocols and Applications, Mail 2003.

[11] K. Baumgartner, Réseaux de capteurs sans fil, IBCOM, Décember 2005.

[12] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. ASurvey on Sensor Network.In IEEE Communications Magazine, Volume: 40, Issue: 8 On pages: 102-114, ISSN: 0163-6804, August 2002.

[13] I.F.Akyildiz, W.Su , Y.Sankarasubramaniam, E.Cayirci «A survey on Sensor Networks», Georgia Institute of Technology, IEEE Communications Magazine, Août 2004.

[14] Yassine Barka «Sécurisation du protocol de routage LEACH dans les réseaux de capteurs»,2009/2010.

[15] I. F. Akyildiz, W. Su , Y. Sankarasubramaniam , E. Cayirci. Wireless sensor networks: a survey. 2002, Vol. 38, pp. 393-422.

[16] C.Y. Chong, S.P. Kumar. Sensor Networks : Evolution, Opportunities, and Challenges. Proceedings of the IEEE. 2003, Vol. 91, pp. 1247-1256.

[17] T.B. Gosnell, J.M. Hall, C.L. Ham, D.A. Knapp, Z.M. Koenig, S.J. Luke, B.A. Pohl, A.Schach von Wittenau, and J.K. Wolford. Gamma-Ray Identification of Nuclear Weapon Materials. Lawrence Livermore National Lab. Livermore CA, USA : s.n., February, 1997. Technical repport DE97053424.

[18] Brown, M. J. Users Guide Developed for the JBREWS Project. Los Alamos National Laboratory of California University. 1999. Technical repport LA-UR-99-4676.

[19] Andrews, P. Johnson and D.C. Remote continuous monitoring in the home. Telemedicine and Telecare. June 1996, Vol. 2, 2, pp. 107-113.

[20] E.M. Petriu, N.D. Georganas, D.C. Petriu, D. Makrakis, and V.Z. Groza. Sensor-based information appliances. IEEE Instrumentation Measurement Magazine. December 2000, Vol. 3, 4, pp. 31-35.

[21] Michael Fitzgerald. Technnology Review : Tracking a Shopper's Habits. Technology Review.04 August 2008.

[22] J.N Al-Karaki et A. E. Kamal, «Routing Techniques in Wireless Sensor Networks: A Survey», Magazine: IEEE Communications, vol. 11, N° 6, pp. 6-28, Dec. 2004.

[23] D. Niculescu, « Topics In Ad-Hoc Networks: Communication Paradigms for Sensor

Networks », NEC Laboratories America, IEEE Communications Magazine, Mars 2005.

[24] Gérard CHALHOUB, “Les réseaux de capteurs sans fil ”

[25] C. Townsend et S. Arms, “Wireless Sensor Networks: Principles and Applications”,MicroStrain, Inc.

[26] R. Jurdak, « Wireless Ad Hoc and Sensor Networks:A Cross-Layer Design perspective»,University College Dublin,2007.

[27] C. Intanagonwiwat, R. Govindan et D. Estrin, « Directed diffusion: A scalable and robust communication paradigm for sensor networks», Proceedings ACM MobiCom'00, pp. 56-67, Boston, MA, Août 2000.

[28] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann et F. Silva, « Directed diffusion for wireless sensor networking», IEEE/ACM Transactions on Networking, vol. 11, N°1, pp.2-16,Fevrier 2003,.

[29] D. Braginsky et D. Estrin, « Rumor Routing Algorithm for Sensor Networks » dans les démarches du premier atelier sur les réseaux de capteurs et applications (WSNA), Atlanta, GA, Octobre 2002.

[30] J. Kulik, W. Heinzelman et H. Balakrishnan, « Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks», Wirel. Netw, Vol. 8, pp.169-185,2002.

[31] K. Akkaya et M. Younis, « A Survey on Routing Protocols for Wireless Sensor Networks». Elsevier, 2003

[32] W.R. Heinzelman, A. Chandrakasan et H. Balakrishnan, «Energy-efficient

Communication Protocol for Wireless Microsensor Networks», dans les démarches de la société d'ordinateur d'IEEE à la trente troisième conférence internationale d'Hawaï sur les sciences de systèmes (HICSS '00),Vol. 8, pp. 8020, Washington DC, USA, Janvier 2000.

[33] W.R. Heinzelman, A. Chandrakasan et H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks" in IEEE Tmnsactions on Wireless Communications (October 2002), vol. 1(4), pp. 660-670.

[34] Y. Yao et J. Gehrke, « The cougar approach to in-network query processing in sensor networks », dans l'enregistrement de SIGMOD , Septembre 2002.

[35] N. Sadagopan, B.Krishnamachari et A. Helmy "The ACQUIRE Mechanism for Efficient Querying in Sensor Networks", In Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA), pp. 149–155,Anchorage,AK, May, 2003.

[36] Y. Yu, D. Estrin et R. Govindan, «Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks» rapport technique du department d'informatique de l'université UCLA, May 2001.

[37] I. F. akyildiz, W. Su, Y. Sankarasubramaniam et E. Cayirci, « Wireless Sensor networks: a Survey » Elsevier , Computer Networks, Vol. 38, N° 4, pp. 393-422, 2002,

[38] S. Kumar Singh, M P Singh et D K Singh, « Routing Protocols in Wireless Sensor Networks –A Survey » revue international de l'informatique et de l'enquête d'ingenierie (IJCSES) Vol.1, N°2, Novembre 2010.

[39] L J G Villalba et A L S Orozco, A T Cabrera et C J B Abbas, « Review: Routing Protocols in Wireless Sensor Networks », 26 Octobre 2009.

[40] M. Ilyas et I. Mahgoub, «Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems», CRC Press LLC, 2005.

[41] T. He et al. «SPEED: A stateless Protocol for real-time communication in Sensor networks», International Conference on Distributed Computing Systems ICDCS, 2003.

[42] A. Manjeshwar et D.P. Agrawal, « TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks », 15ème colloque international sur le traitement parallèle et distribute (IPDPS-01), 2001.

[43] A. Manjeshwar et D.P. Agrawal, « APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks », 16ème Colloque International sur le traitement parallèle et distribué (IPDPS-02), 2002.

[44] S. Y. Ni, Y. C. Tseng, Y.S. Chen, and J. P. Sheu. The broadcast storm problem in a mobile ad hoc network. In Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom), Seattle, USA, August 1999.

[45] FrançoisIngelrest, Thèse de Doctorat de l'Université des sciences et technologies de Lile : Protocoles localisés de diffusion et économie d'énergie dans les réseaux AD HOC et de capteur, juin 2006.l

[46] Julien CARTIGNY, Thèse de Doctorat de l'Université des sciences et technologies de Lile : Contributions à la diffusion dans les réseaux ad hoc, décembre 2003.

[47] Houda Labiod, Réseaux mobiles ad hoc et réseaux de capteur sans fil, 2006.

Résumé

Les RCSFs sont utilisé de plus en plus dans le monde des télécommunications, par ailleurs le routage dans ces réseaux reste limité vu les problèmes issues des caractéristiques de ses réseaux. Optimiser ce genre de problèmes dans ces réseaux est une perspective pour augmenter leur utilité.

Le problème qui pose, c'est que chaque nœud diffuse le paquet a tout ces voisins ce qui va créer une surcharge de la bande passante « inondation aveugle » et va aussi gaspiller énormément d'énergie au niveau des nœuds vu que tous ces derniers participeront a cette inondation jusqu'à atteindre la destination désirée, ce qui risque de les faire disparaitre du réseau au cas où leurs batteries s'éteindront.

Afin de mettre fin à ce problème on utilise CBEF – WSN qui remplace cette inondation par une inondation probabiliste pour améliorer le routage et ainsi avoir meilleur utilisation des ressources avec ce type de réseaux.

Les capteurs fonctionnent donc à basse tension et ceci est géré par un système d'exploitation spécialisé : TinyOS. Enfin, il doit exister un environnement de développement logiciel afin d'importer des applications sur les capteurs. Le langage utilisé pour ce développement est le nesC.

Mots clés: Réseaux de capteurs, Protocole de routage, Diffusion.

LISTE DES TABLEAUX