LA REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université MOULOUD MAMMERI de Tizi Ouzou

Faculté de génie électrique et d'Informatique

Département d'Informatique

Spécialité : Système Informatique

Pour l'obtention d'un diplôme de master en informatique

Thème intitulé :

L'amélioration des techniques de mise en veille dans le protocole LEACH

Encadré par : M^r^ Mehammed Daoui

Présenté par : M^elle^ Benamara Sarah

21/09/2011

Remerciements

Mes premiers remerciements et ma grande gratitude s'expriment envers mon encadreur Mr Mehammed Daoui.

Ses conseils et ses encouragements ont permis à ce travail d'aboutir à la réussite. Ses capacités scientifiques et ses compétences étaient mon grand support.

La liberté qu'il m'a accordée et les responsabilités qu'il m'a confiées ont beaucoup contribué à ma formation et à mon autonomie de travail.

Ses critiques et suggestions ont été d'une précieuse aide pour la réalisation de ce projet.
Je le remercie chaleureusement pour sa pédagogie, sa patience, sa disponibilité et son dévouement.

Faire ce projet sous sa direction était pour moi un grand honneur et un immense bonheur.

J'exprime ma gratitude et ma reconnaissance à messieurs les membres de jury Mrs Hameg, Hamrioui et Mme Belkadi, ainsi je les remercie chaleureusement d'avoir pris le soin et le temps de bien juger ce travail.

J'adresse également mes sincères remerciements à ma famille et mes amis de m'avoir aidé à surmonter tous les obstacles et à me forger à travers les difficultés vécues durant toute cette période de travail.

A ceux qui ont été continuellement présents, qui m'ont épaulée par leur aide, soutien et encouragement.

Soyez assurés de mon profond respect.

M^elle Benamara Sarah

Liste des tableaux

Liste des figures

Résumé :

Les progrès technologiques qui ont été réalisés durant ces dernières années ont permis de développer de nouveaux types de capteurs équipés de moyens de communication sans fil qui sont peu onéreux et pouvant être configurés dans le but de former des réseaux autonomes dits réseaux de capteurs sans fil.

Durant ces dernières années, la conservation d'énergie est un problème important, donc plusieurs travaux se concentrent sur la conservation d'énergie dans les RCSF.

Le principe utilisé dans la plupart de ces travaux est de permettre aux nœuds capteurs de se mettre en veille au lieu de rester actif, ce qui consomme beaucoup d'énergie.

Dans ce mémoire, nous nous sommes tout d'abord focalisés sur les techniques de mise en veille dans les réseaux de capteurs sans fil, puis nous avons conçu une solution qui permet de réduire le nombre de nœuds mis en veille de façon à avoir un Cluster_Head suivi de son sous Cluster_Head dans le but de maximiser la durée de vie du réseau.

Abstract:

Technological advances which have been realized in the recent years have developed a new types of sensors equipped with wireless communication means which are inexpensive and can be configured in order to form networks called autonomous wireless sensor networks.
In recent years, energy conservation is an important issue, so more work will focus on energy conservation in WSN.

The principle used in most of this work is to allow sensor nodes go to sleep instead of staying active, which is energy intensive.

In this paper, we first focused on the techniques of sleep in wireless sensor networks, then we have designed a solution that reduces the number of nodes paused in order to have a Cluster_Head followed by its sub Cluster_Head in order to maximize the lifetime of the network.

Introduction générale

Introduction générale

Depuis quelques années, plusieurs technologies ont contribué à la production des composants de type capteurs plus petits en termes de taille avec un prix faible.
Ces composants se réunissent ensemble pour former des réseaux de capteurs sans fil dits aussi RCSF dont ils sont importants pour un certain nombre d'applications stratégiques comme la détection d'une cible et la surveillance d'une zone d'intérêt [6][KM07].

La gestion d'énergie dans ces réseaux est cruciale puisque les capteurs ont de sévères contraintes énergétiques.

À cet égard, utiliser des techniques efficaces optimisant la consommation d'énergie est indispensable pour réaliser des économies énergétiques significatives dans un RCSF.

Par conséquent, les protocoles conçus pour les RCSF doivent judicieusement utiliser les ressources énergétiques finies et doivent faire fonctionner les capteurs uniquement en cas de besoin, autrement, tous les capteurs doivent être en mode veille.

Donc, l'objectif de notre travail est de proposer un protocole qui réalise des réductions supplémentaires dans la consommation d'énergie.

Chapitre I :

Les concepts généraux sur les réseaux de capteurs sans fil

1. Introduction :

Dans le milieu industriel et scientifique, il est devenu nécessaire, depuis quelques décennies, d'observer et de contrôler certains phénomènes physiques tels que la pression atmosphérique, la température, le degré de pollution de l'air…etc.

Ceci a été rendu possible grâce aux industries qui proposent alors des capteurs sans fil qui peuvent renseigner l'utilisateur sur plusieurs données.

En effet, les progrès réalisés en microélectronique et micromécanique ont permis la conception d'unités miniatures possédant un dispositif de stockage ainsi qu'un système d'exploitation.
Celles-ci étant miniatures sont communément appelées capteurs.
Ils peuvent aussi être reliés ensemble pour former un réseau sans fil qui se base sur des protocoles pour communiquer et propose des programmes et des réseaux embarqués.

Lors de cette thèse plus particulièrement dans ce chapitre, nous allons découvrir les caractéristiques essentielles d'un tel réseau.

En effet, un aspect primordial à ne pas négliger dans les réseaux de capteur sans fil est bien la consommation énergétique à base de batterie qui est ni rechargeable, ni remplaçable.
La contrainte principale alors serait d'introduire des techniques de mise veille pour réduire la consommation d'énergie afin de maximiser la durée de vie du réseau.
Les capteurs fonctionnent donc à basse tension et ceux-ci sont gérés par un système d'exploitation spécialisé appelé : TinyOS.

Donc, afin d'importer des applications sur ce type de capteurs, nous allons utiliser un environnement de développement supervisé par un système d'exploitation TinyOS accompagné d'un langage de développement qui est le NesC.

Il en a découlé l'obligation de simuler virtuellement un réseau de capteurs, ainsi, il sera possible d'analyser le comportement des différents éléments d'un capteur, la mise en réseau de ceux-ci et leur durée de vie afin de choisir les meilleurs compromis.

2. Généralités sur les RCSF :

2.1. Qu'est ce qu'un capteur (senseur)?

C'est un système qui sert à détecter, sous forme de signal souvent électrique, un phénomène physique (température, humidité ou l'intensité lumineuse) afin de le représenter.

Les capteurs sont de petits appareils dotés d'une batterie, capables de communiquer entre eux et de détecter des événements s'ils se trouvent à l'intérieur de leur rayon de perception [9].

2.2. Définition d'un Réseau de Capteur Sans Fil « RCSF » :

Les nœuds capteurs sont des capteurs intelligents "smart sensors", capables d'accomplir les tâches complémentaires : le relevé d'une grandeur physique, le traitement éventuel de cette information et la communication avec d'autres capteurs tout en utilisant une consommation à base d'énergie. L'ensemble de ces capteurs, déployés pour une application, forme un réseau de capteurs qui a pour but la surveillance d'une zone géographique, et parfois d'agir sur celle-ci (il s'agit alors de réseaux de capteurs-actionneurs).
Ou bien la détection de feu de forêt, ou surveillance de la solidité d'un pont après un tremblement de terre.

Toute intervention humaine après le déploiement des nœuds capteurs est la plupart du temps exclue, le réseau doit donc s'autogérer.
Les capteurs sont placés de manière plus ou moins aléatoire (par exemple à l'aide d'un hélicoptère) dans des environnements pouvant être dangereux.

Afin que les nœuds capteurs travaillent d'une façon coopérative, les informations recueillies sont partagées entre eux par voie hertzienne. Le choix du lien radio plutôt que du lien filaire permet un déploiement facile et rapide dans un environnement pouvant être inaccessible pour l'être humain [9, 10].

2.3. Architecture d'un nœud-capteur :

Un nœud capteur est composé de quatre unités principales dont chacune correspond à une tâche particulière qui peut être de capture, de traitement, de communication ou d'énergie [1] :

2.3.1. L'unité de capture (Sensing unit) :

Cette unité est composée de deux sous-unités, un dispositif de capture physique qui prélève l'information de l'environnement local et un convertisseur analogique/numérique appelé ADC (Analog to Digital Converters).

2.3.2. L'unité de traitement des données (Processing unit) :

Les données captées sont communiquées au processeur où elles sont stockées dans la mémoire. Ces processeurs qui sont utilisés dans le cadre de réseaux de capteurs sont à faible consommation d'énergie, leurs fréquences sont assez faibles, moins de 10 MHz pour une consommation de l'ordre de 1 mW.
Une autre caractéristique est la taille de leur mémoire qui est de l'ordre de 10 Ko de RAM pour les données et de 10 Ko de ROM pour les programmes [6, 3].
Cette mémoire consomme la majeure partie de l'énergie allouée au microcontrôleur, c'est pourquoi on lui adjoint souvent une autre mémoire moins coûteuse en énergie.

2.3.3. Unité de communication (Transceiver unit) :

Elle est composée d'un émetteur/récepteur (module radio) permettant la communication entre les différents nœuds du réseau.

2.3.4. Unité d'énergie (Power unit) :

Pour des réseaux de capteurs sans fil autonomes, l'alimentation est une composante cruciale.

Il y a essentiellement deux aspects : premièrement, stocker l'énergie et la fournir sous la forme requise ; deuxièmement, tenter de reconstituer l'énergie consommée par un réapprovisionnement grâce à une source externe au nœud-capteur telles les cellules solaires. Le stockage énergétique se fait traditionnellement en utilisant des piles.

À titre indicatif, ce sera souvent une pile AA normale d'environ 2,2 à 2,5 Ah fonctionnant à 1,5 V [6].

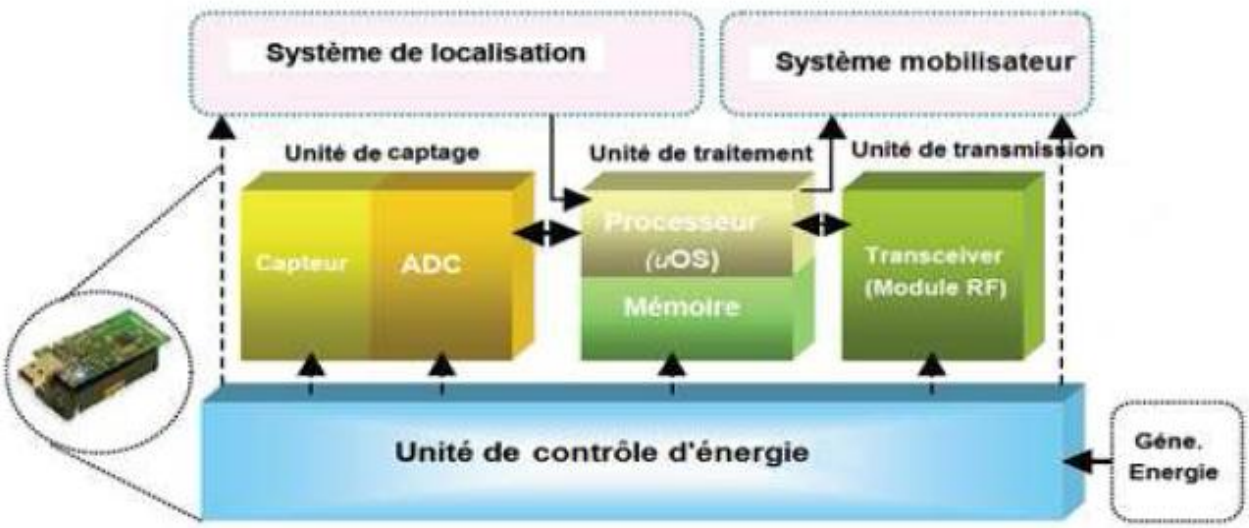

Figure 1 : Architecture d'un nœud capteur [39]

2.4. Architecture de communication d'un RCSF :

Le réseau de capteur sans fil est constitué d'un champ de captage (field), un ensemble de nœuds capteurs, une station de base et un centre de traitement de données : [11,12]

L'espace de collecte ou champ de captage est vue comme étant le centre d'intérêt pour le phénomène capté.

Les nœuds capteurs représentent le cœur du réseau ils sont présents dans le but de collecter les données afin de les router vers la station de base (Sink).

La station de base dite aussi puit ou sink est un nœud particulier chargé de stocker, récolter et traiter les différentes données qui proviennent des autres nœuds.

En effet, ce nœud est soit doté d'une interface réseau tel que le GPRS ou Ethernet soit une entité extérieure au réseau tel que le PDA ou l'ordinateur portable.

Il reste toujours actif dans le but de recevoir les informations alors sa source énergétique doit être limitée pour ne pas s'épuiser rapidement.

Le gestionnaire de tâche (centre de traitement de données) :

Son rôle est la réception de données collectées par la station de base, les regrouper puis les traiter pour ne garder que les informations utiles.

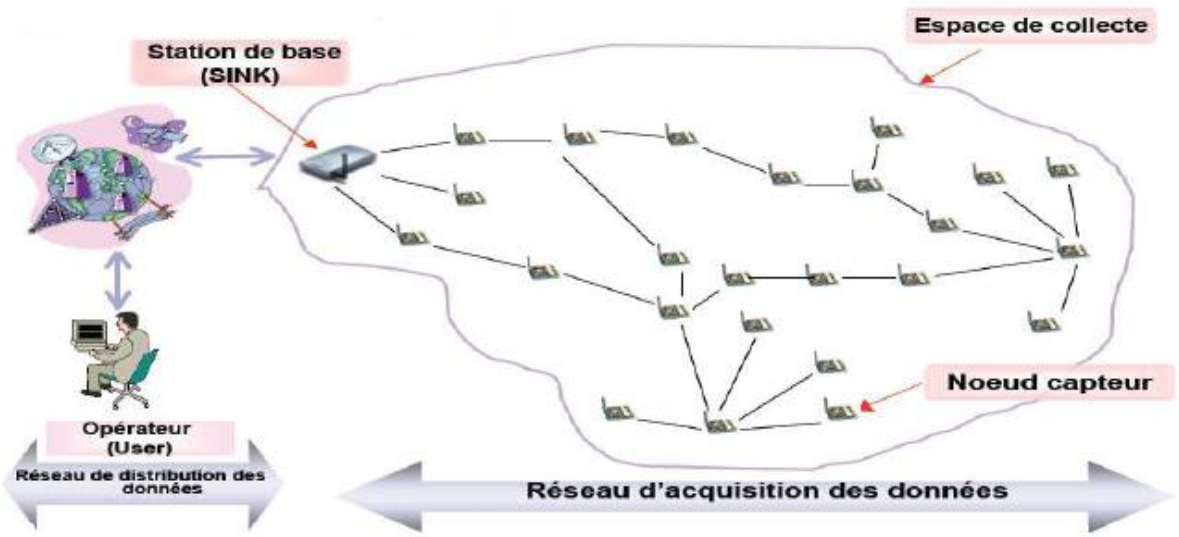

Figure 2 : Architecture des RCS [39]

2.5. Les types d'architecture pour les réseaux de capteurs sans fil :

Il existe deux types de réseaux de capteurs sans fil représentés comme suit :

- Les réseaux de capteurs sans fil plats :

sont des réseaux homogènes dans lesquelles les nœuds possèdent le même niveau en terme de capacité et fonctionnalité concernant le captage, seule la station de base joue un rôle différent celui d'une passerelle qui collecte les données issues des différents nœuds pour les transmettre à l'utilisateur.

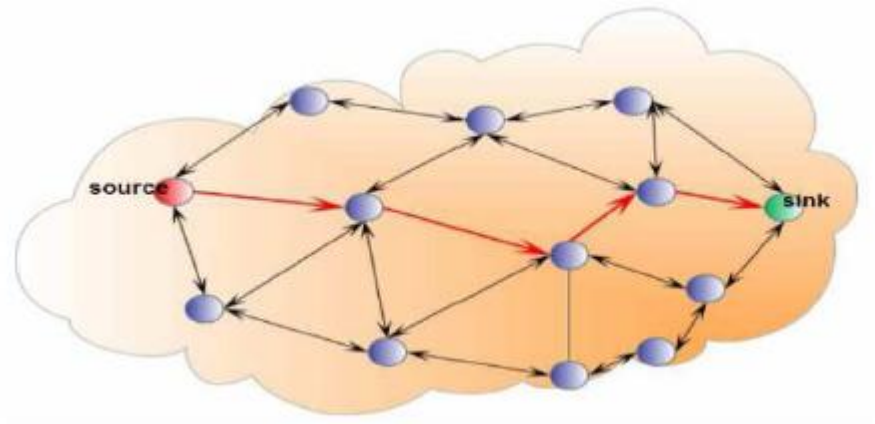

Figure 3 : Architecture des RCSF Plats [40]

- Type de RCSF hiérarchiques :

C'est un ensemble de réseaux hétérogènes dans lesquels la capacité des nœuds est différente en termes de ressource énergétique qui peut être plus importante dans certains nœuds que dans d'autres, et en termes d'une plus grande portée de communication et puissance de calcul.

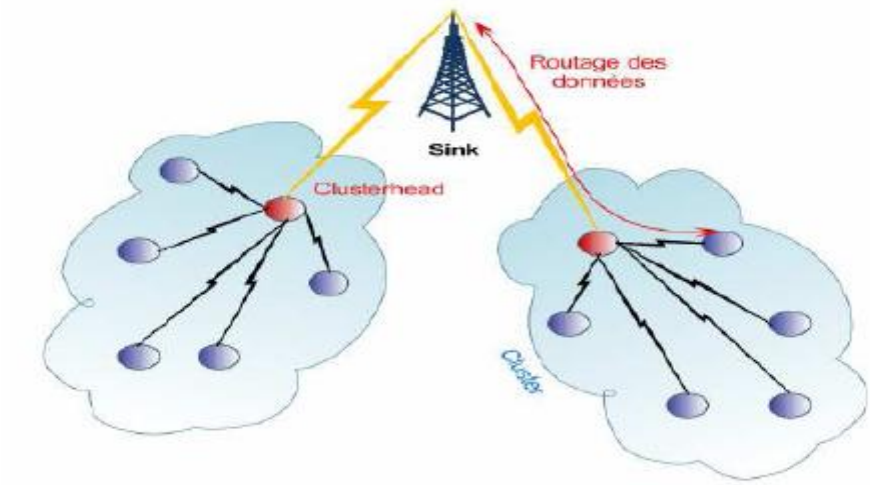

Figure 4 : Architecture hiérarchique [40]

2.6. Les différents facteurs de conception de RCSF:

De nombreux facteurs comme la tolérance aux pannes, les coûts de production, la consommation d'énergie, l'environnement ou la topologie du réseau influencent la conception des réseaux de capteurs sans fil.

Nous citons parmi eux :

2.6.1. Le facteur d'échelle dit aussi (Scalabilité) :

Le réseau doit être capable de fonctionner avec un très grand nombre de capteurs qui peut atteindre des milliers de nœuds tout en permettant l'augmentation de ce nombre et la concentration (densité) des ces derniers dans une région (pouvant dépasser 20 noeuds/m³) [1].

Un nombre aussi important engendre beaucoup de transmissions inter nodales (implémentation d'une détection d'erreur, d'un contrôle de flux,..) et nécessite que le puits

soit équipé de beaucoup de mémoire pour stocker les informations reçues.

2.6.2. La topologie dynamique :

La topologie des réseaux de capteurs peut changer au cours du temps pour les raisons qui suivent :

- Suite à l'expiration de l'énergie d'un nœud capteur, celui-ci peut devenir non opérationnel.
- La mobilité des nœuds ainsi que la station de base rendent le réseau dynamique.
- La défaillance d'un nœud capteur est, donc très probable car il peut être déployé dans des environnements hostiles (champ de bataille par exemple).

2.6.3. La durée de vie du réseau :

C'est l'intervalle de temps qui sépare l'instant de déploiement du réseau de l'instant où l'énergie du premier nœud s'épuise. Selon l'application, la durée de vie exigée pour un réseau peut varier entre quelques heures et plusieurs années.

2.6.4. Tolérance aux pannes :

La panne peut être fréquente suite à une erreur de fabrication ou plus fréquemment à un manque d'énergie. Les interférences ou chocs (interactions externes) peuvent aussi être la cause des dysfonctionnements.

2.6.5. Coût de fabrication :

Souvent, les réseaux de capteurs sont composés d'un très grand nombre de nœuds dont le prix d'un est critique afin de pouvoir concurrencer un réseau de surveillance traditionnel.

Actuellement un nœud ne coûte souvent pas beaucoup plus que 1$. Par contre un nœud Bluetooth, pourtant déjà connu pour être un système low-cost, revient à environ à 10$.

2.6.6. Topologie du réseau :

Il faut que les nœuds-capteurs soient capables d'adapter leur fonctionnement afin de maintenir la topologie souhaitée.

On distingue généralement trois phases dans la mise en place d'un réseau ainsi que son évolution:

- Déploiement :

Les nœuds sont soit répartis de manière prédéfinie soit de manière aléatoire alors, il faut alors que ceux-ci s'organisent de manière autonome.

- Post-Déploiement - Exploitation :

Durant la phase d'exploitation, suite à des pannes ou à la modification de la position des nœuds, la topologie du réseau peut être soumise à des changements.

- Redéploiement :

La mise à jour du réseau s'effectue aussi lors de l'ajout de nouveaux capteurs dans celui-ci.

2.6.7. Consommation d'énergie :

La contrainte primordiale dans un réseau de capteur sans fil est bien l'énergie.

En effet, la recharge des sources énergétique est souvent trop coûteuse et parfois impossible. Il faut donc que les capteurs économisent l'énergie afin de pouvoir maximiser la durée de vie du réseau.

Les réseaux de capteurs fonctionnant selon un mode de routage par saut, chaque nœud joue un rôle important dans la transmission de données ainsi le mauvais fonctionnement de l'un implique un changement dans la topologie et impose une réorganisation du réseau.

2.6.8. Les médias de transmission :

Dans un réseau de capteurs, les nœuds sont reliés par une architecture sans-fil.

En effet le média de transmission doit être normé pour effectuer des opérations sur ces réseaux dans le monde entier.

Dans le plus souvent des cas, on utilise l'infrarouge (qui est robuste aux interférences, et peu onéreux), le Bluetooth et les communications radio ZigBee.

2.7. Applications des RCSF :

Les applications des RCSF sont classées en quatre grandes catégories [8].

- Applications orientées temps :

Dans ce type d'applications l'acquisition et la transmission des données capturées sont liées au temps : période d'acquisition qui peut être de quelques secondes jusqu'à quelques heures voire des jours.
Ainsi, la quantité de données échangées dans le réseau dépend de la périodicité des mesures à effectuer sur l'environnement local.

La collecte de données environnementales peut représenter un bon exemple de cette classe d'application dans des domaines variés : agriculture, expérimentation scientifique, etc.

- Applications orientées requêtes

Cette classe d'application est destinée aux applications adaptées à l'utilisateur qui peut requérir des informations à partir de certaines régions dans le réseau ou interroger les capteurs pour acquérir des mesures d'intérêts.
Dans ce cas, un capteur ne peut envoyer une information que si la station de base le lui demande.

- Applications orientées événements

Cette classe d'application est destinée par exemple à la surveillance des feux de forêts dans laquelle un capteur envoie des alarmes à la station de base dès que la température dépasse un certain seuil.

Elle a été conçue dans le domaine militaire pour la surveillance du déplacement d'objets dans le champ de bataille.
Par la suite, elle a rapidement trouvé de nouvelles perspectives comme le contrôle industriel, le contrôle médical des patients, la surveillance d'édifices (barrages, ponts, voies de chemins de fer, etc.).

Dans ce cas, les capteurs ne peuvent envoyer des données uniquement si un événement spécifique se produit.

- Applications hybrides

Ce type d'application met en œuvre les trois modes de fonctionnement décrits précédemment.
Par exemple, le réseau peut combiner entre un réseau de type (time driven) c'est-à-dire orienté temps et un réseau de type (event driven) orienté événement.

2.8. Le domaine Applications des RCSF :

Les réseaux de capteurs sans fil possèdent beaucoup d'applications citées comme suit :

- Détection d'intrusions :

Les nœuds capteurs seront placés dans de différents points stratégiques dans le but de prévenir des cambriolages ou des passages de gibier sur une voie de chemin de fer (par exemple) sans avoir à recourir à de coûteux dispositifs de surveillance vidéo.

Figure 5: application des RCSF pour la découverte d'intrusion

- Découvertes de catastrophes naturelles :

Les capteurs dispersés dans la nature peuvent ainsi signaler des événements tels que les feux de forêts, tempêtes ou inondations. Ceci permet une intervention beaucoup plus rapide et efficace des secours pour minimiser les dégâts.

Figure 6 : application des RCSF pour le relevé météo logique.

- Contrôle de pollution :

Dans ce domaine, on disperse les capteurs au-dessus d'un emplacement industriel pour détecter et contrôler les fuites de gaz ou les produits chimiques.

- Agriculture :

On incorpore dans le domaine de l'agriculture les nœuds sous terre pour pouvoir questionner le réseau de capteurs sur l'état du champ afin de déterminer par exemple les secteurs les plus secs dans le but de les arroser en priorité.

Pour reconnaître également la position des troupeaux de bétail à tout moment, on les équipe de capteurs afin d'éviter aux éleveurs d'avoir recours à des chiens de berger.

- Contrôle d'édifices :

On inclut des capteurs entre les sacs de sables formant une digue de fortune. La détection rapide d'infiltration d'eau peut servir à renforcer le barrage en conséquence. Cette technique peut aussi être utilisée pour d'autres constructions tels que les ponts, les voies de chemins de fer, les routes de montagnes, les bâtiments et autres ouvrages d'art.

- Surveillance médicale :

Pour pouvoir surveiller la progression d'une maladie, on implante sous la peau de mini capteurs vidéo, pour recevoir des images en temps réel d'une partie du corps sans aucune chirurgie pendant environ 24h.

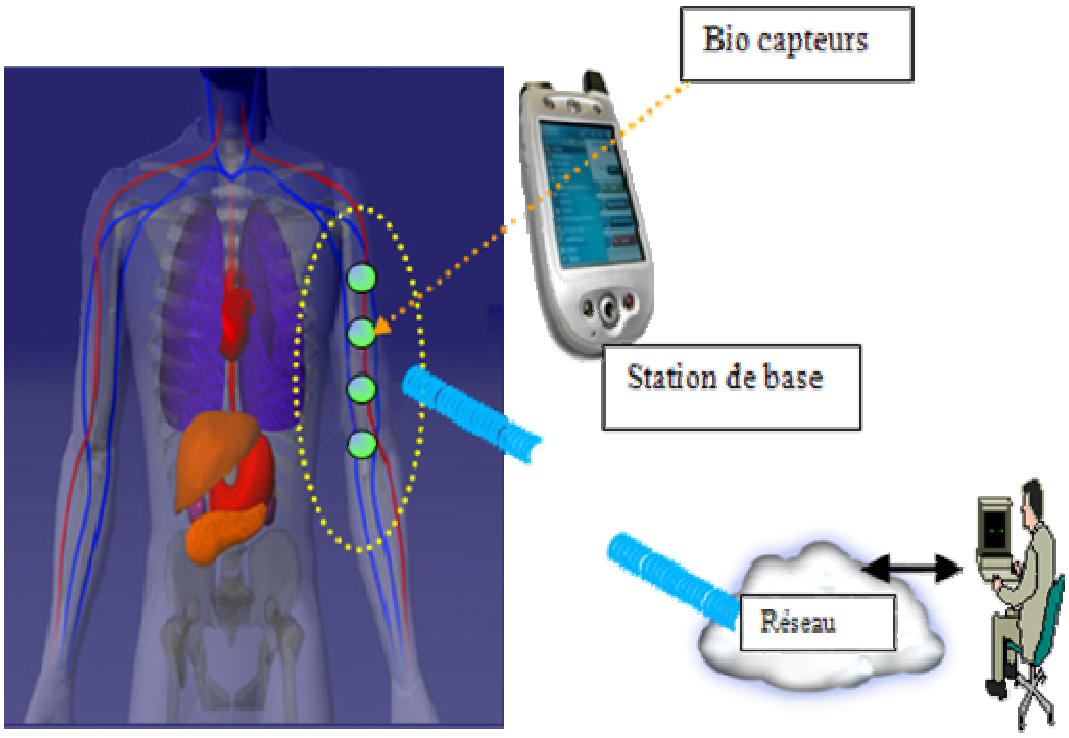

Figure 7: application des RCSF dans le domaine médical

2.9. La pile protocolaire :

La pile protocolaire [1] est illustrée dans la figure ci-dessous. Elle comprend la couche application, la couche transport, la couche réseau, la couche liaison de données, la couche physique, le plan de gestion de l'énergie, le plan de gestion de la mobilité et le plan de gestion des tâches.

On commence par les définir une à une :

2.9.1. La couche physique :

Cette couche est celle du niveau bas du modèle OSI, elle spécifie les caractéristiques matérielles, les fréquences porteuses, et assure la transmission et la réception des données au niveau bit etc...

2.9.2. La couche liaison :

Le protocole MAC (Media Access Control) de cette couche assure la gestion de l'accès au support physique. Cette couche permet de montrer comment les données sont expédiées entre deux nœuds/routeurs. Elle est responsable du multiplexage des données, du contrôle d'erreurs, de l'accès au media. Elle assure la liaison point à point et multi-point dans un réseau de communication.

2.9.3. La couche réseau :

La couche réseau prend soin de router les données fournies par la couche transport (son but principal est de trouver une route et une transmission fiable des données captées, des nœuds capteurs vers le puits "sink" en optimisant l'utilisation de l'énergie des capteurs).

2.9.4. La couche transport :

Cette couche permet de transport les données, gérer le flux si le réseau de capteurs l'exige. Elle permet de réordonne et rassemble les segments venus de la couche réseau pour les envoyer à la couche application.

2.9.5. La couche application :

C'est l'interface avec les applications, Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par les logiciels. De plus de ces couches c'est-à-dire celles du modèle OSI, nous disposons également de trois couches supplémentaires qui sont citées comme suit:

2.9.6. Plans de gestion :

Le but de ces plans est la longévité du réseau en permettant aux nœuds de collaborer entre eux, acheminer les données dans le réseau mobile et partager les ressources de manière efficace tout en réduisant la consommation énergétique.

2.9.7. Plan de gestion de mobilité :

D'après son nom « mobilité», cette partie permet de détecter et enregistrer le mouvement des nœuds. Ainsi, le nœud peut garder trace de ses nœuds voisins. En déterminant leurs voisins, les nœuds capteurs peuvent balancer l'utilisation de leur énergie et la réalisation de tâche.

2.9.8. Plan de gestion d'énergie :

contrôle l'utilisation de la batterie. Par exemple, après la réception d'un message, le capteur éteint son récepteur afin d'éviter la duplication des messages déjà reçus. En outre, si le niveau d'énergie devient bas, le nœud diffuse à ses voisins une alerte les informant qu'il ne peut plus participer au routage. L'énergie restante est réservée au captage.

2.9.9. Plan de gestion de tâche :

Cette gestion permet de balancer et ordonnancer les différentes tâches de captage de données dans une région spécifique.

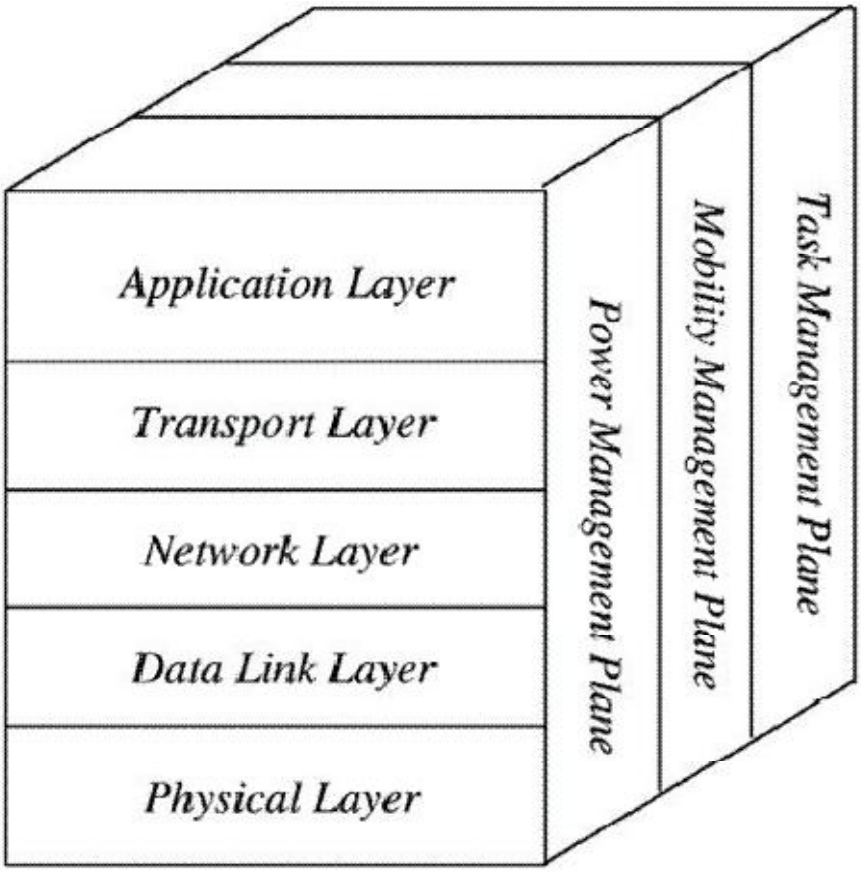

Figure 8 : Modèle en couches pour la communication dans les RCSF

3. La comparaison entre les réseaux de capteurs sans fil et les réseaux MANET :

Les RCFS sont souvent comparés aux réseaux ad hoc traditionnels. En effet, Les MANET partagent beaucoup de points communs avec les réseaux de capteurs sans fil. Nous commençons en premier lieu par les points communs essentiels qui sont :

- L'absence d'infrastructure :

Tous les deux sont des réseaux ad hoc, c'est `à dire qu'ils fonctionnent sans avoir besoin d'une infrastructure pour la gestion des échanges, d'où la nécessité de l'auto-configuration.

- Communication sans fil :

Tous les deux sont des réseaux sans fil, ce qui fait que la portée des communications est limitée par la capacité de rayonnement des antennes utilisées et les puissances mises en jeu.

- L'utilisation de protocoles multi-sauts :

Ainsi, les nœuds dans ces types de réseaux sont souvent dans des configurations multi-sauts ce qui induit la mise en place de protocoles de routage.

- Le médium :

Le médium utilisé pour l'échanger d'informations entre les nœuds est l'air. Ainsi, les protocoles d'accès au médium des réseaux MANET et des RCSF sont très proches et sont typiquement en mode half-duplex (les entités étant incapables de recevoir et d'´emettre en même temps).

- La consommation énergétique à base de batterie :

Les entités de ces réseaux sont souvent alimentées par des batteries qui sont ni rechargeables ni remplaçable. Il existe malgré les points en commun de ces deux types de réseaux plusieurs points différents qui se résument dans le tableau juste en dessous :

Les réseaux de capteurs sans fil	Les réseaux MANETs
1-Les entités dans les réseaux de capteurs sans fil interagissent essentiellement avec la nature ou l'environnement ou entre-elles. 2-Le fait que les nœuds du réseau dans un RCSF sont souvent déployés dans des environnements hostiles (forêts, volcans, etc.) cela augmente le risque que ces nœuds tombent en panne beaucoup plus souvent. 3-Les topologies dans les RCFS sont assez dynamiques et doivent être robustes pour palier aux problèmes de pannes. 4-Les nœuds dans sont déployés pour des durées exprimées en mois, voire en années, sans l'intervention humaine. Dans la plupart des cas, ces nœuds sont alimentés par des batteries. Cette contrainte affecte le fonctionnement de ces nœuds et les oblige à travailler selon un cycle d'activité intermittent. Ceci a un effet direct sur la disponibilité des nœuds : un nœud en état inactif ne peut pas communiquer avec les autres nœuds du réseau. Ainsi, la conception des protocoles dans un RCSF doit prendre en compte cette contrainte. 5- Les échanges de données dans les applications d'un réseau RCSF sont souvent du type collecte de données. Les nœuds doivent envoyer vers un puits des informations sur des phénomènes observés, ceci soit à la demande du puits, périodiquement, au déclenchement d'une alarme, ou bien un mélange des trois.	1-Les entités d'un réseau MANET sont utilisées directement par des êtres humains, comme les portables, les PDA, etc. 2-Dans un réseau MANET, vu que les nœuds ne sont pas déployés dans des zones hostiles, cela diminue le risque que ces derniers tombent en panne plus souvent. 3-contrairement aux réseaux MANET, le défi essentiel des protocoles du réseau est typiquement la mobilité des nœuds. 4- Contrairement aux réseaux MANETs, qui ne prennent pas en considération le fait qu'un nœud mis en veille qui est hors du cycle d'activité ne puisse communiquer avec un autre nœud. 5- les applications des réseaux MANET sont plus orientées calcul distribué et donc le trafic circule entre tous les nœuds du réseau et dans tous les sens.

Tableau 1 : Comparaison entre les RCSF et réseaux Ad Hoc

4. Conclusion :

Dans ce chapitre introductif, nous avons procédé à l'étude de concepts généraux sur les réseaux de capteurs sans fil.

Nous avons posé les briques de base et fédéré quelques concepts nécessaires à la compréhension de nos problématiques dans la suite de cette thèse.

Cela fait des années que les réseaux de capteurs suscitent un engouement important dans la

recherche.

Nous avons remarqué à travers nos lectures que minimiser la consommation énergétique d'un nœud capteur est le cheval de bataille de toutes les solutions et protocoles proposés.

En effet, lorsque ce n'est pas l'objectif principal, alors c'est sûrement un critère de performance capitale.

Nous avons évoqué la définition de quelques notions primordiales pour acquérir les connaissances à propos du mémoire.

Dans les prochaines sections, nous conjuguerons cela avec la notion de longévité dans les réseaux de capteurs sans fil et nous dresserons un panorama de techniques de mise en veille proposées dans ces types de réseaux, puis nous élaborerons une approche dite PWLEACH dont le rôle est de réduire la consommation énergétique pour la maximisation de la survie du réseau.

Chapitre II :

La consommation d’énergie dans les réseaux de capteurs sans fil

1. Introduction :

Les capteurs sans fil sont vus comme étant des dispositifs équipés d'une source énergétique à faible capacité [3].

En effet, la durée de vie d'un capteur, et par conséquent celle du réseau est étroitement liée à celle de sa batterie (son module d'alimentation).

Les travaux de recherche se focalisent sur la conception de protocoles et d'algorithmes à faible consommation pour les RCSF.

La consommation d'énergie d'un capteur peut donc être distribuée sur trois unités:

- Unité de capture
- Unité de traitement des données
- Unité de communication.

La minimisation de la consommation joue un rôle primordial, et c'est essentiellement pour cette raison que la contrainte d'énergie est considérée comme étant celle de premier ordre dans ce type de réseaux car la durée de vie d'un capteur est liée à sa réserve d'énergie et à la façon dont celle-ci est utilisée [19].

Mais un nœud, dont l'énergie est épuisée n'impliquerait pas forcément la mort du réseau dans son ensemble. Alors, certains algorithmes n'hésitent pas à sacrifier quelques nœuds en épuisant l'intégralité de leur réserve sur une période donnée pour augmenter significativement la durée de vie globale du RCSF.
Donc l'objectif de ce chapitre consiste à présenter les principaux facteurs et contraintes qui influencent la conception des réseaux de capteurs sans fil. Par la suite, nous décrirons la problématique de la consommation d'énergie dans les réseaux de capteurs sans fil.

Finalement, nous présenterons une vue globale de solutions proposées pour chaque type d'énergie pour la gestion de la consommation d'énergie.

2. La consommation d'énergie :

Les nœuds capteurs sont des composants micro-électroniques qui sont équipés de sources énergétiques limitées.

De plus, dans certaines applications, ces nœuds ne peuvent pas être dotés de mécanismes de rechargement d'énergie, par conséquent, la durée de vie d'un nœud capteur dépend fortement de la durée de vie de la batterie associée.

Comme les réseaux de capteurs sont basés sur la communication multi-sauts, chaque nœud joue à la fois un rôle d'initiateur de données et de routeur également.

Le dysfonctionnement d'un certain nombre de nœuds entraine un changement significatif sur la topologie globale du réseau, et peut nécessiter un routage de paquets différents et une réorganisation totale du reseau.

C'est pour cela que le facteur de consommation d'énergie est d'une importance primordiale dans les réseaux de capteurs.

La majorité des travaux de recherche menés actuellement se concentrent sur le problème de la contrainte énergétique afin de concevoir des algorithmes et protocoles spécifiques à ce genre de réseau qui consomment le minimum d'énergie.

Dans les réseaux de capteurs, l'efficacité en consommation d'énergie représente une métrique de performance significative, qui influence directement la durée de vie du réseau en entier.

Pour cela, les concepteurs peuvent, au moment du développement des protocoles négliger les autres métriques de performances telles que la durée de transmission et le débit, au profit du facteur de consommation d'énergie [13].

En effet, La consommation d'énergie dans un RCSF, peut être divisée selon l'ordre décroissant suivant [14] : La communication (émission et réception), traitement de données, et capture.

3. Modèle de consommation d'énergie :

Heinzelman et al. [19] proposent un modèle radio de consommation d'énergie.

(Ainsi, les énergies nécessaires pour émettre des messages ETx(s,d) et recevoir ERx(s) sont données par les formules suivantes:

• Pour émettre un message de s bits vers un récepteur loin de d mètres, l'émetteur consomme:

$$ETx(s, d) = ETx\ elec(s) + ETx\ amp(s, d)$$

$$ETx(s, d) = (Eelec * s) + (Eamp * s * d^2)$$

• Pour recevoir un message de s bits, le récepteur consomme :

$$ERx(s) = ERx\ elec(s)$$

$$ERx(s) = Eelec * s$$

Eelec et Eamp représentent respectivement l'énergie de transmission électronique et d'amplification.

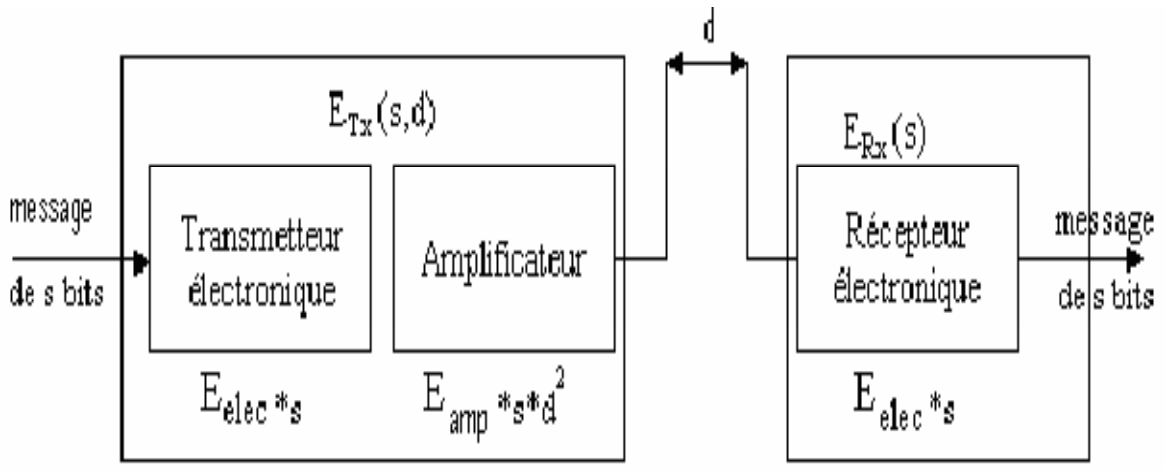

Figure 9: Modèle de consommation d'énergie [19].

4. Consommation d'énergie d'un nœud-capteur

4.1. Formes de dissipation d'énergie

Les nœuds-capteurs sont alimentés principalement par des batteries. Ils doivent donc fonctionner avec un bilan énergétique frugal. En outre, ils doivent le plus souvent avoir une

durée de vie de l'ordre de plusieurs mois, voire de quelques années, puisque le remplacement des batteries n'est pas une option envisageable pour des réseaux avec des milliers de nœuds.

Afin de concevoir des solutions efficaces en énergie, il est extrêmement important de faire d'abord une analyse de différents facteurs provoquant une dissipation de l'énergie d'un nœud-capteur [52].

Cette dissipation d'énergie se fait de manière générale selon plusieurs modes, nous citons :

- Le MCU2 : Généralement les MCUs possèdent divers modes de fonctionnement tel

que le mode actif, veille, idle (allumé mais ne fonctionne pas). Chaque mode est caractérisé par une quantité différente de consommation d'énergie. Par exemple, le MSP4303 consomme 3 mW en mode actif, et seulement 15 _W dans le mode sommeil.

Toutefois, la transition entre les modes de fonctionnement implique un surplus d'énergie et de latence. Ainsi, les niveaux de consommation d'énergie des différents modes, les coûts de transition entre les modes mais encore le temps passé par le MCU dans chaque mode ont une incidence importante sur la consommation totale d'énergie d'un nœud-capteur.

- La radio : la radio opère dans quatre modes de fonctionnement : émission, réception,

actif, et sommeil. Une observation importante dans le cas de la plupart des radios est que le mode idle induit une consommation d'énergie significative, presque égale à la consommation en mode réception [53].

Ainsi, il est plus judicieux d'éteindre complètement la radio plutôt que de passer en mode "idle" quand l'on a ni à émettre ni à recevoir de données. Un autre facteur déterminant est que, le passage de la radio d'un mode à un autre engendre une dissipation d'énergie importante due à l'activité des circuits électroniques.

Par exemple, quand la radio passe du mode sommeil au mode émission pour envoyer un paquet, une importante quantité d'énergie est consommée pour le démarrage de l'émetteur lui-même [52].

- Le détecteur ou le capteur proprement dit : il y a plusieurs sources de consommation

d'énergie par le module de détection, notamment l'échantillonnage et la conversion des signaux physiques en signaux électriques, le conditionnement des signaux et la conversion analogique-numérique.

Étant donné la diversité des capteurs, il n'y a pas de valeurs typiques de l'énergie consommée.

En revanche, les capteurs passifs (température, sismiques, ...) consomment le plus souvent peu d'énergie par rapport aux autres composants du nœud-capteur. Notons, les capteurs actifs tels que les sonars, les capteurs d'images, etc. peuvent consommer beaucoup d'énergie.

Un autre aspect non négligeable est le phénomène d'auto-décharge de la batterie. En effet, cette dernière se décharge d'elle même et perd de sa capacité au fil du temps.

Il est difficile d'apporter ici une étude quantitative et comparative précise de la consommation

de chaque composant d'un nœud-capteur en raison du grand nombre de plates-formes commerciales existantes. Cependant, des expérimentations ont montré que c'est la transmission de données qui est la plus consommatrice en énergie [52].

4.2. Facteurs qui interviennent dans la consommation d'énergie :

Nous appelons surconsommation d'énergie toute consommation inutile que l'on peut éviter afin de conserver l'énergie d'un nœud-capteur. Les sources de cette surconsommation sont nombreuses, elles peuvent être engendrées lors de la détection lorsque celle-ci est mal gérée (par exemple par une fréquence d'échantillonnage qui est mal contrôlée).
La surconsommation concerne également la partie communication. En effet, cette dernière est sujette à plusieurs phénomènes qui surconsomment de l'énergie surtout au niveau MAC où se déroule le contrôle d'accès au support sans fil.
Plusieurs facteurs entrent dans la consommation d'énergie, et on cite parmi eux les suivants [19]:

- Les collisions :

Les collisions sont la première source de perte d'énergie. Quand deux trames sont émises en même temps et se heurtent, elles deviennent inexploitables et doivent être abandonnées.
Les retransmettre par la suite, consomme de l'énergie. Tous les protocoles MAC essayent à leur manière d'éviter les collisions.

Ø La retransmission :

La retransmission des paquets perdus peut engendrer une perte significative de l'énergie car, les nœuds capteurs n'ont qu'une seule antenne radio et n'utilisent qu'un canal de transmission, alors la transmission simultanée des données qui proviennent de plusieurs capteurs engendrent des problèmes de collisions et ainsi une perte d'information transmise.

Ø Le « idle listening » ou écoute active :

Ceci se produit éventuellement lorsque la réception attend un paquet qui ne sera pas reçu, ce qui peut engendrer une perte importante de la capacité des nœuds en énergie.

Ø Etat du module radio

On distingue quatre états des composants radio (transmetteur et récepteur) : actif, réception, transmission et sommeil [24].

• Etat actif : la radio est allumée, mais non employée car le nœud capteur n'est ni en train de recevoir ni de transmettre.
En effet, suite à l'écoute inutile du canal de transmission, cet état provoque une perte d'énergie.

• Etat sommeil : la radio est mise en veille.

• Etat transmission : la radio transmet un paquet.

• Etat réception : la radio reçoit un paquet.

En effet, l'énergie de transition est le vrai problème qui se pose car le passage fréquent de l'état actif à l'état sommeil cause une consommation d'énergie plus importante que de laisser le module radio allumé.

Ceci est dû à la puissance nécessaire pour la mise sous tension du module radio. Alors, il est souhaitable d'arrêter complètement la radio plutôt que de transiter dans le mode sommeil.

Ø Le « Overhearing » ou surécoute:

La surécoute conduit à une perte d'énergie additionnelle à cause de l'implication des autres capteurs dans la réception des données. Ceci se produit quand un nœud reçoit un paquet qui ne lui est pas destiné.

Puisque ce facteur engendre l'implication d'autres capteurs dans la réception de données, alors la perte énergétique serait additionnelle.

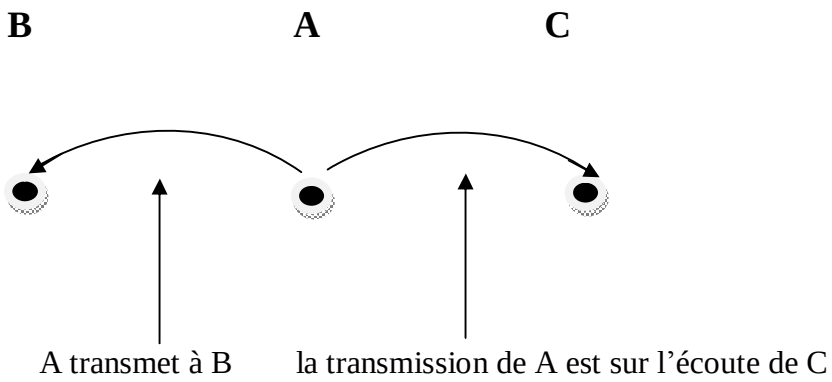

Figure 10 : La surécoute dans une transmission.

Ø Le « Overemitting » ou surémission :

La surémission se produit lorsqu'un nœud capteur envoie les données à un destinataire qui n'est pas prêt à les recevoir, suite a cela, les messages envoyés sont considérés comme inutiles puisqu'ils consomment une énergie additionnelle.

Ø La surcharge :

Plusieurs protocoles de la couche MAC fonctionnent par échange de messages de contrôle (overhead) pour assurer différentes fonctionnalités : signalisation, connectivité, établissement de plan d'accès et évitement de collisions. Tous ces messages nécessitent une énergie additionnelle.

- La taille des paquets :

L'impacte de la taille des messages échangés dans le réseau a un effet sur la consommation énergétique. Ainsi, celle-ci ne doit être ni trop élevée ni trop faible.
Si elle est petite, le nombre de paquets de contrôle (acquittement) générés augmente.
Dans le cas contraire, une grande puissance de transmission est nécessaire pour des paquets de grande taille.

4. La minimisation d'énergie dans les RCSF :

L'énergie que le nœud consomme peut être due à de nombreuses opérations essentielles représentées comme suit :

A. Energie de capture :

Une tâche est effectuée par l'unité de capture qui traduit les phénomènes physiques en signal électrique pouvant être digital ou analogique. Il existe plusieurs types de ce composant qui mesurent les paramètres d'environnement comme la température, le son, l'image, la pression, etc. En effet, cette énergie est dissipée pour accomplir les tâches suivantes :

- L'échantillonnage.
- Le traitement de signal.
- La conversion analogique/numérique.

En général, l'énergie de capture représente un faible pourcentage de l'énergie totale consommée par un nœud. Prenons l'exemple des capteurs de température ou de tremblement de terre qui sont moins consommateurs d'énergie par rapport à ceux d'imagerie ou de vidéo [15].

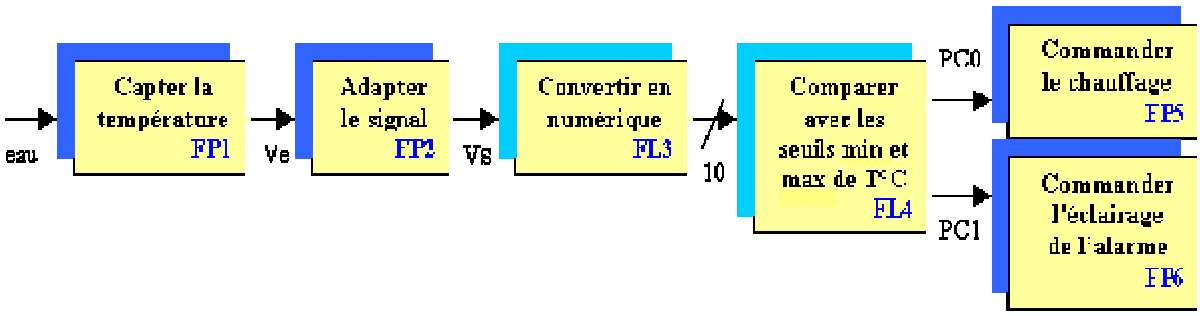

Figure 11: schéma fonctionnel d'un capteur de température

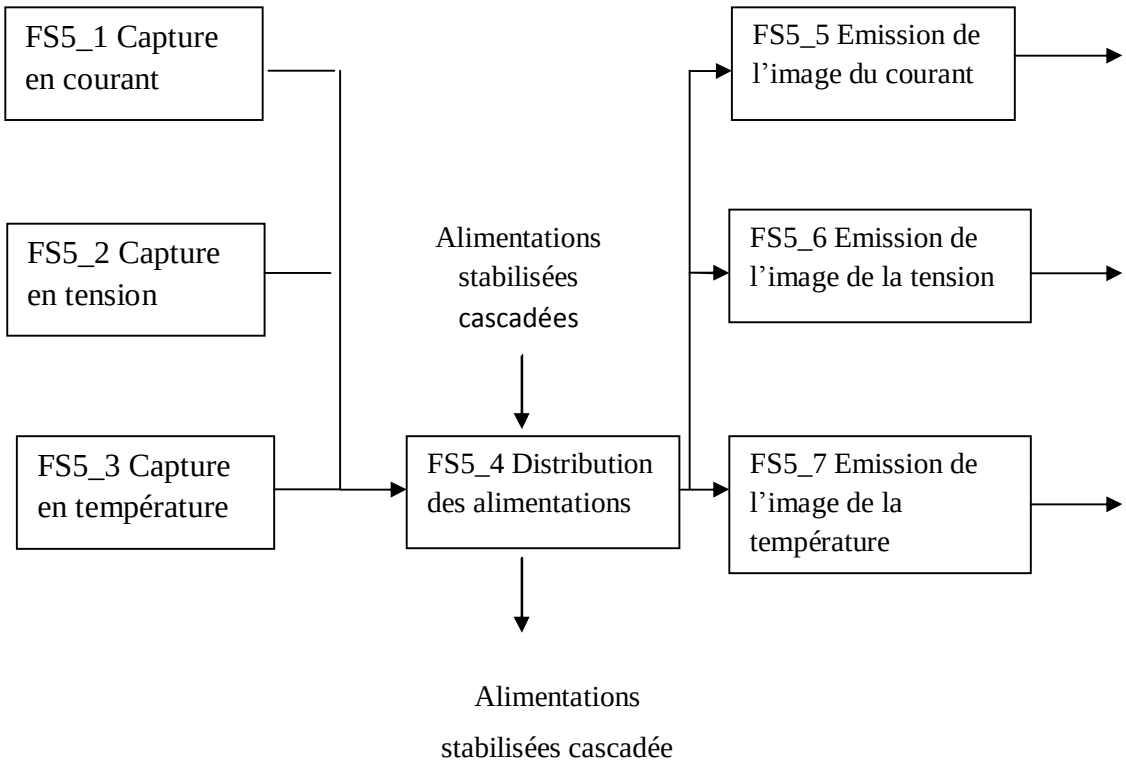

Figure 12 : schéma fonctionnel d'un capteur de vidéo

Ainsi, la solution serait d'utiliser des composants à faible consommation mais ne réduisant pas leur performances dans le but de minimiser la consommation d'énergie lors de la capture. Ou bien réduire les durées de capture par la suppression de capture inutile.

B. Energie de traitements :

En effet, cette énergie se divise en deux parties :

- l'énergie de commutation
- l'énergie de fuite

L'énergie de commutation est déterminée par la tension d'alimentation. Par contre, l'énergie de fuite correspond à l'énergie consommée lorsque l'unité de calcul est en état repos (c'est-à-dire qui n'effectue aucun traitement).
Cette énergie est faible par rapport à celle nécessaire pour la communication.
Pour pouvoir minimiser la consommation d'énergie on illustre deux techniques ou approches lors du traitement de données par un nœud capteur

ü L'approche de partitionnement du système :

qui permet de transférer un calcul prohibitif (abusif) en termes de temps de calcul vers une station de base qui possède une grande capacité de calcul qui n'a pas de contraintes énergétiques dans le but de minimiser la consommation d'énergie qui est si primordiale dans ce type de réseau.

ü L'approche Dynamic Voltage Scaling « DVS » :

qui ajuster la tension d'alimentation et la fréquence du microprocesseur pour économiser la puissance de calcul sans dégradation de performances [18].

C. Energie de communication :

L'énergie de communication représente la plus grande proportion de l'énergie totale consommée au niveau d'un nœud [16]. Elle se décline en deux parties :

- L'énergie de l'émission
- L'énergie de réception

En effet, la distance de transmission ainsi que la quantité des données à communiquer déterminent ce type d'énergie. La puissance du signal caractérise son émission, donc quand

cette puissance augmente, le signal aura une grande portée et l'énergie consommée sera plus élevée.

En effet, la perte d'énergie due à un mauvais acheminement des paquets de données influe sur la durée de vie du réseau.
Donc les protocoles sont utilisés pour trouver les chemins optimaux en consommation d'énergie.

La minimisation de celle-ci pendant la communication est liée aux protocoles développés pour la couche réseau et la couche MAC [17], et qui se basent sur plusieurs techniques :

- Négociation :

Ce mécanisme permet aussi de régler le problème de chevauchement, d'implosion et d'inondation.
Et pour les éviter, on précède l'émission d'une donnée par sa description, en utilisant la notion de métadonnées, ainsi le récepteur aura le choix par la suite d'accepter la donnée ou non.

- CSIP(Collaborative Signal and Information Processing) :

Est une technique qui combine plusieurs domaines:

- La communication.
- Le calcul à basse puissance.
- Le traitement de signal.
- Les algorithmes distribués.
- La tolérance aux fautes.

L'objectif de cette technique est de réduire le nombre d'émission/ réception des messages.
En ce qui concerne le contrôle de la topologie [23], ce dernier permet l'ajustement de la hiérarchisation.

- La hiérarchisation :

est un terme qui signifie le regroupement de nœuds capteurs, ainsi que l'organisation du réseau en structure à plusieurs niveaux.

Prenons l'exemple du clustering qui organise « divise » le réseau en cluster (groupe) avec des membres supervisés par un « Cluster Head » [20].

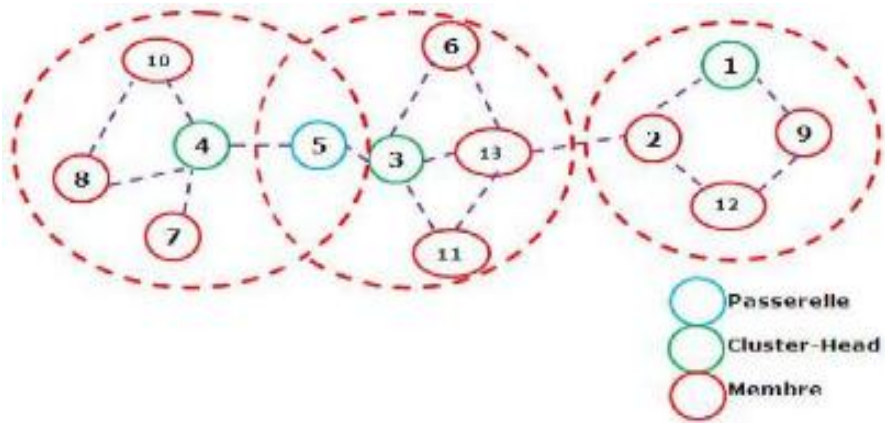

Figure 13 : principe de Clustering

L'autre solution qui a été proposée dans [25] consiste à prendre la densité élevée des capteurs comme un atout primordial pour mettre en veille ou endormir certains nœuds capteurs pour qu'ils ne soient pas actifs en même temps.

C'est une solution assez maligne pour pouvoir éviter la grande consommation d'énergie dans les RCSF.

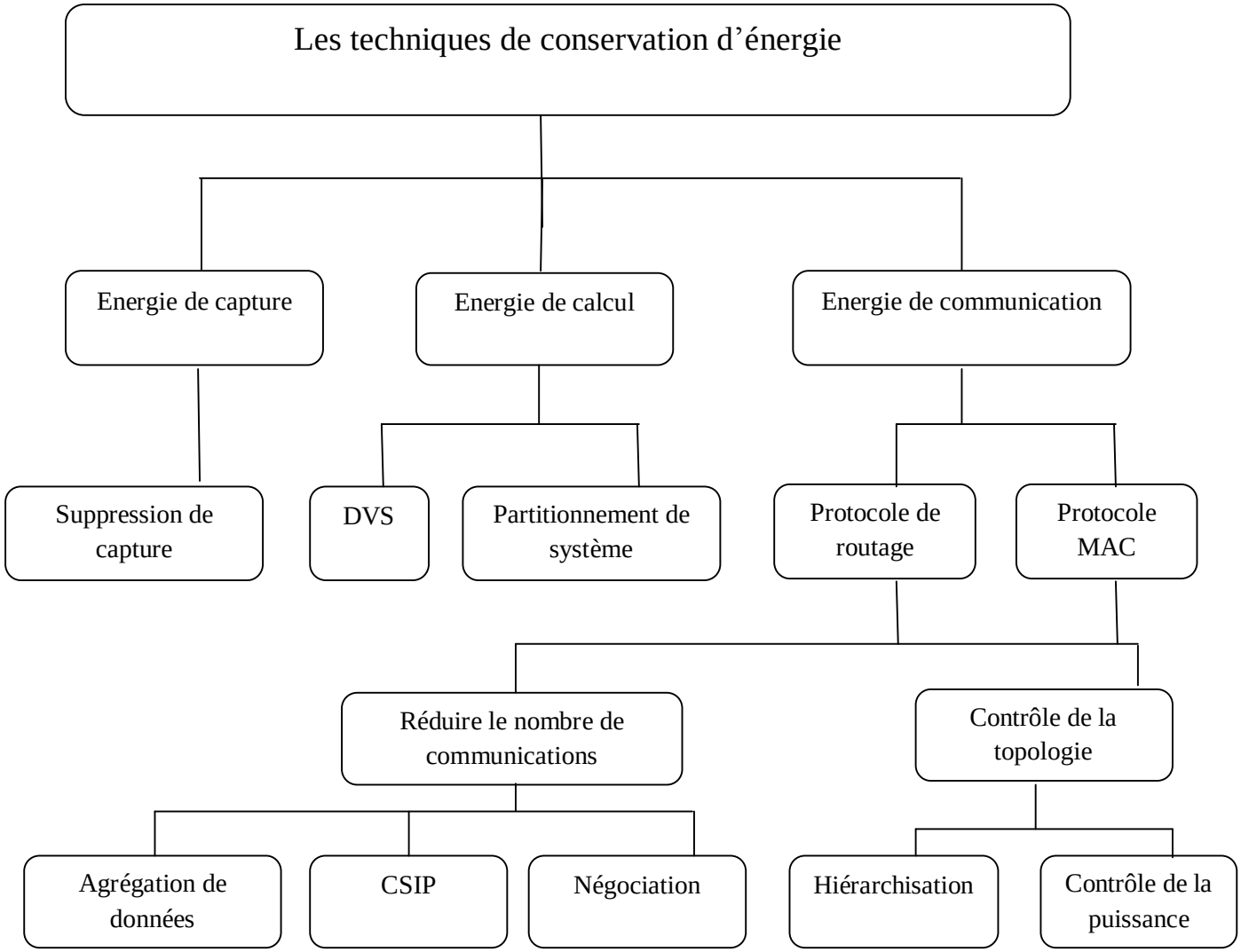

Figure 14 : Approches utilisées pour minimiser la conservation d'énergie [21]

Il existe également d'autres mécanismes de la conservation d'énergie tels que :

- Traitement local :

Cette stratégie de traitement local permet de réduire sensiblement le trafic, car l'idée de cette technique est que la source peut se censurer. Ainsi une programmation événementielle semble être bien adaptée aux réseaux de capteurs. Seuls les changements significatifs de l'environnement devraient provoquer un envoi de paquets dans le réseau.

Dans le même état d'esprit, une grande collaboration est attendue entre les capteurs d'une même région en raison de leur forte densité et dans la mesure où les observations ne varient

presque pas entre des voisins très proches. Ainsi les données pourront être confrontées localement et agrégées au sein d'un seul et unique message.

- Organisation des échanges :

Ce procédé revient à limiter les problèmes de retransmission dus aux collisions, par l'introduction de protocoles tels que : le S-MAC (Sensor MAC) [47] qui est une méthode d'accès au canal de type CSMA-CA avec le mécanisme RTS/CTS (Request to Send, Clear to Send) qui permet d'éviter les collisions et le problème de la station cachée. La principale innovation, apportée par ce protocole, est d'avoir un mécanisme de mise en veille distribué sur chaque nœud du réseau dans le but de réduire la consommation d'énergie.

- Limitation des accusés de réception :

L'acquittement systématique est mal adapté à des réseaux denses : il provoque une surcharge de celui-ci et donc des collisions et des interférences avec les données échangées.

- La répartition de la consommation d'énergie :

En effet, dans le cas d'une transmission directe vers l'observateur 2.15(a), les capteurs éloignés vont plus rapidement manquer d'énergie et les autres nœuds peuvent être sujets au phénomène d'overhearing.

Au contraire, dans le cas d'une transmission par saut 2.15(b), les nœuds proches de l'observateur vont être vite en rupture de batterie car ils seront plus sollicités pour relayer les messages des autres. La solution consiste à hiérarchiser les échanges en divisant la zone d'observation en clusters 2.15(c). Un chef dit Cluster_Head est élu pour chaque cluster dont le rôle est la récupération d'informations auprès des capteurs de son cluster [HCB00].

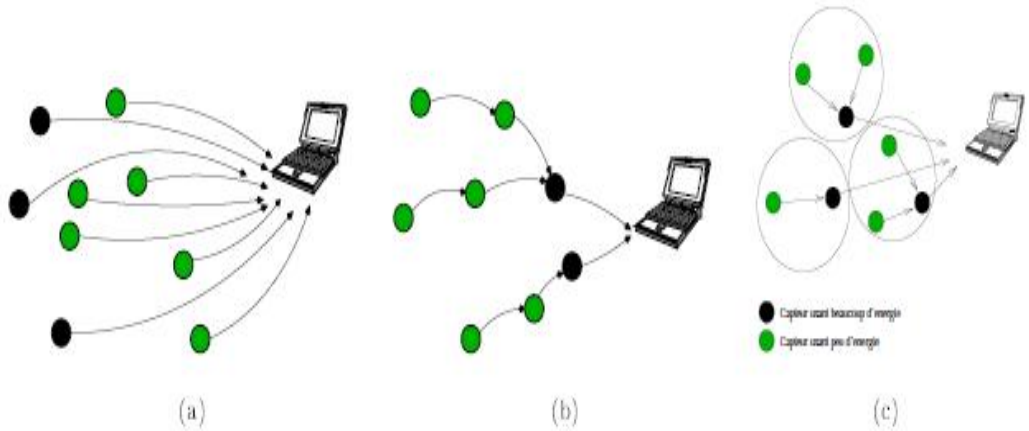

Figure 15 : À gauche - Transmissions directes. Au centre - Transmission saut par saut. À droite - Hiérarchisation en clusters.

- La conservation de l'énergie par clustering ou création de grappe:

La formation de grappes est une technique qui consiste à subdiviser le réseau en sous ensembles de nœuds, appelés grappes. Les capteurs sont alors groupés en sous-ensembles ayant chacun un capteur désigné comme TG (Tête de Grappe) et des nœuds membres connectés à la TG.

Les nœuds membres transmettent leurs données à la TG qui, à son tour, les achemine jusqu'au centre de traitement, soit directement [4817], soit en multi-sauts via des passerelles ou d'autres TG voisines [4933]. Les TG peuvent effectuer certaines opérations comme le filtrage et l'agrégation des données collectées, ce qui permet de limiter la quantité de données transmises à la station de base et ainsi alléger la bande passante et sauvegarder l'énergie des nœuds collecteurs.

Dans certains cas, la formation de grappes offre des économies d'énergie importantes, [50,34] puisque seules les TG sont impliquées dans les tâches de collecte, routage et agrégation des données.

Plusieurs protocoles de formation de grappes ont été proposés dans la littérature et ont généralement le prolongement de la durée de vie du réseau comme objectif principal.

Pour cela, la formation de grappes est typiquement basée sur l'énergie résiduelle des nœuds.

Par exemple, Low-Energy Adaptive Clustering Hierarchy (LEACH) [48,17] est l'un des premiers protocoles de formation de grappes proposés pour les RCSF, il est réparti, proactif et dynamique. Dans LEACH, la formation des grappes se base sur la puissance du signal reçu pour évaluer le degré d'un nœud et sa proximité de ses voisins.

Chaque nœud prend sa décision de devenir ou non TG en se basant sur le nombre de fois qu'il a joué ce rôle dans le passé. Les TG sont les seuls nœuds qui peuvent router les données vers la station de base, ce qui sauvegarde l'énergie des autres capteurs.

- Par l'ajustement optimisé des puissances de transmission

Dans un RCSF où les capteurs sont équipés de plusieurs puissances de transmission ou d'une puissance ajustable, l'énergie consommée dépend directement de l'énergie de transmission. Certains travaux se sont penchés sur l'optimisation de la puissance de transmission des capteurs.

Par exemple, dans [51], les chercheurs ont proposés un protocole réparti qui permet d'ajuster dynamiquement la puissance de transmission de chaque capteur pour contrôler la taille des grappes. L'idée proposée est que chaque capteur ajuste sa puissance de transmission en fonction du nombre de ses voisins.

- Par la planification optimisée des états des capteurs :

Beaucoup de travaux se sont intéressés à la planification optimale des états des capteurs et à l'ordonnancement optimal des activités de ces derniers, dans le but de minimiser l'énergie consommée par le réseau. Il s'agit, d'une façon générale, d'alterner entre un (ou des) état(s) où le capteur est activé, et donc consomme un certain niveau d'énergie, et un autre où il est éteint (en veille) et donc consomme une quantité d'énergie négligeable. Beaucoup de travaux proposent des techniques de mise en veille des capteurs pour préserver leur énergie.

5. Conclusion :

Dans ce chapitre nous avons procédé à l'étude de la consommation énergétique des réseaux de capteurs sans fil.

Nous avons posé les briques de base et fédéré quelques concepts nécessaires à la compréhension de nos problématiques dans la suite de ce manuscrit.

En effet, les réseaux de capteurs sans fil présentent un intérêt considérable et une nouvelle étape dans l'évolution des technologies de l'information et de la communication.

Ils doivent intégrer des mécanismes qui permettent aux utilisateurs de prolonger la durée de vie du réseau en entier, car chaque nœud est alimenté par une source d'énergie limitée et généralement irremplaçable.

Nous avons remarqué à travers nos lectures que minimiser la consommation d'énergie d'un nœud capteur est le cheval de bataille de toutes les solutions et protocoles proposés.

De ce fait, la communauté de recherche est en train de développer et de raffiner plusieurs techniques de conservation d'énergie.

Dans le chapitre qui suit, nous conjuguerons cela avec la notion de la durée de vie du réseau et nous dresserons un panorama des techniques de mise en veille dans les réseaux de capteurs sans fil proposées dans la littérature afin de prolonger la durée de vie des réseaux de capteurs.

1. Introduction

Dans les réseaux de capteur sans fil, la conservation de l'énergie est un problème important.

Plusieurs travaux se concentrent sur la conservation de l'énergie dans la communication à l'aide de protocoles de contrôle d'accès au medium (MAC) particulièrement pour ces réseaux.

Le principe utilisé dans la majorité de cette thèse est de rendre la mise en veille des nœuds capteurs possibles au lieu de rester en mode actif tout le temps, ce qui cause une grande consommation énergétique.

Mais les transitions fréquentes entre les modes "veille" et « réveil » consomme également de l'énergie.

À cet effet, quelques protocoles de mise en veille tel que (ASLEEP, Duty-Cycling, sleep / wakeup) ont été représentés pour une gestion efficace de l'énergie dans les réseaux de capteurs sans fil.

Donc l'objectif de ce chapitre est de proposer une nouvelle approche pour les réseaux des capteurs sans fil qui évite la grosse consommation d'énergie dans ce type de réseau par le biais de techniques de mise en veille.

2. Notion de durée de vie d'un réseau

La vie d'un réseau de capteurs correspond à la période de temps durant laquelle le réseau peut, selon le cas : maintenir assez de connectivité, couvrir le domaine entier, ou garder le taux de perte d'information en-dessous d'un certain niveau. La vie du système est donc liée à la vie nodale, même si elle peut en différer. La vie nodale correspond à la vie d'un des nœuds du réseau. Elle dépend essentiellement de deux facteurs : l'énergie qu'il consomme en fonction du temps et la quantité d'énergie dont il dispose.

Selon la discussion de Akyildiz et al. dans [3], la quantité prédominante d'énergie est consommée par un nœud-capteur durant la détection, la communication puis le traitement des données. Il existe différentes définitions pour la durée de vie d'un réseau de capteurs (fondées sur la fonctionnalité désirée).

Elle peut être définie comme suit:

1. La durée de vie du réseau persiste jusqu'à ce que le premier nœud épuise toute son énergie.

2. Ou bien jusqu'à ce que le premier Cluster_Head épuise toute son énergie.

3. La durée jusqu'à ce que tous les capteurs épuisent leur énergie.

3. La notion de mise en veille :

Le nœud capteur est un dispositif doté d'une batterie qui à certains moment est épuisable et non rechargeable, donc la bonne gestion énergétique est vraiment cruciale pour pouvoir maximiser la durée de vie du réseau.

La solution la plus efficace pour conserver de l'énergie serait de mettre le capteur de l'émetteur en veille quand la communication de celui-ci n'est pas nécessaire, et de l'activer quand c'est nécessaire.

4. La taxonomie des techniques de mise en veille dans les réseaux de capteurs sans fil

4.1. La technique du Duty Cycling :

La technique du Duty Cycling a été introduite dans les réseaux de capteur sans fil dans le but de minimiser la consommation d'énergie, car le moyen le plus efficace de conserver la consommation énergétique serait de mettre le capteur de l'émetteur / récepteur en mode veille (low-power) à chaque fois que la communication n'est pas nécessaire.

Ainsi, le nœud capteur doit être éteint (le passage vers le mode sommeil) lorsqu'il n ya plus de transmission ou de réception de paquet, et devrait être prêt dès qu'un nouveau paquet de données doit être envoyé ou reçu.

Ceci engendre une alternance entre la période active et celle de mise en veille en fonction de l'activité du réseau. Ce comportement est généralement dénommé Duty-cycling.

4.1.1. Définition d'un Duty-cycle :

Un Duty-cycle est vu comme étant la fraction de temps où les nœuds sont actifs et accompagnés d'un algorithme d'ordonnancement de mise en veille/Réveil.

4.2. Le protocole Sleep/Wakeup :

Il s'agit généralement d'un algorithme distribué qui peut être défini pour un composant donné (le module Radio) du nœud-capteur. Il repose sur les dates auxquelles les nœuds décident de transiter entre l'état actif et l'état sommeil et permettent aux nœuds voisins d'être actifs en même temps, ce qui rend possible l'échange de paquets.

Le protocole est divisé en trois grandes catégories :

- A la demande
- Le rendez-vous programmés
- Les régimes asynchrones.

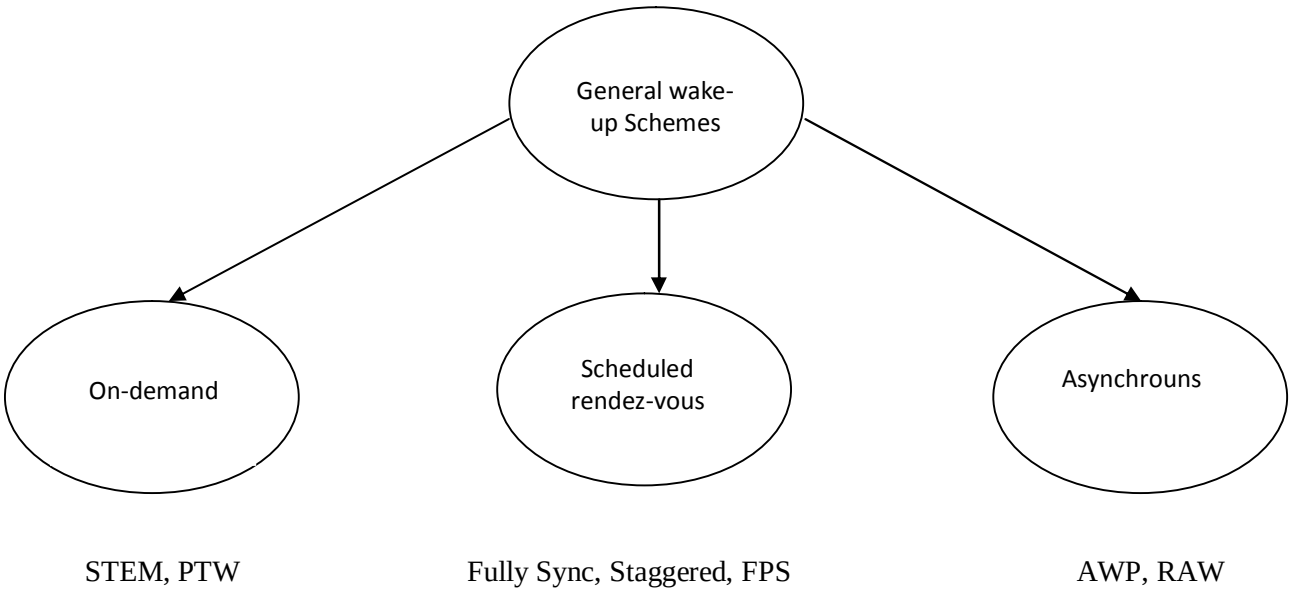

Figure 16: Le protocole Sleep/Wakup

- Les protocoles à la demande (on-demand):

Le but de cette technique de base est qu'un nœud mis en veille devrait se réveiller seulement quand un autre nœud veut communiquer avec lui.
Cette technique engendre un problème principal qui est de savoir comment informer un nœud en sommeil qu'un autre nœud est disposé à communiquer avec lui.

- L'approche de rendez-vous programmés :

Cette approche la, permet aux différents nœuds mis en veille de se réveiller simultanément que leurs voisins, ils se réveillent suivant un ordonnancement de réveil et restent actifs pour communiquer avec leurs voisins pendant un court intervalle de temps.
Ensuite, ils se remettent en veille (se rendorment) jusqu'au prochain rendez-vous.
Parmi les techniques de mis en veille de l'approche de rendez-vous programmés, nous citons :

A. le TAG : Fixed Staggered Scheme dans lequel l'intervalle TI « Talk Interval » est fixé et égale par rapport à tous les nœuds.

B. Adaptive Staggered sLEEP (ASLEEP) : permet d'ajuster dynamiquement les programmes de mise en veilles de façon à avoir des intervalles de temps TI ajustés.

Ø Le protocole sleep/wakeup asynchrone :

Avec la technique de Sleep/wakup asynchrone, les nœuds se réveillent quand ils veulent et tant qu'ils sont capables de communiquer avec leurs voisins [38].

5. Les Protocoles du niveau MAC

Plusieurs protocoles MAC pour les réseaux de capteurs sans fil ont été proposés, et nous nous concentrons principalement sur les questions de gestion d'énergie plutôt que sur les méthodes d'accès au canal.

5.1. Le protocole MAC : (Medieum Access Control) :

Le contrôle d'accès au medium MAC est une partie de la couche liaison de données située au dessus de la couche physique du modèle OSI

Les algorithmes de contrôle d'accès au médium sont utilisés pour permettre à plusieurs utilisateurs de partager simultanément un médium de communication.

Leur but est de maximiser l'utilisation du canal avec un minimum d'interférences et de collisions.

Le protocole MAC détermine le temps où un nœud atteint le médium pour tenter de transmettre, de contrôler ou de faire passer un paquet à un autre nœud ou à un ensemble de nœuds.

Dans cette section, nous allons présenter les protocoles MAC dédiés aux RCSF. Ces protocoles se basent sur les modes d'accès au canal suivant : CSMA/CA, NP-CSMA-PS,

TDMA et un mécanisme hybride, mais nous focaliserons nos recherches sur le mode CSMA/CA.

5.2. Aperçu de MAC :

La couche MAC IEEE 802.15.4 :

Ø Utilise deux modes d'adressage IEEE 64-bit & 16-bit

Ø Accès canal CSMA-CA

- Utilise une structure de trame simple
- Permet d'utiliser le mécanisme de beaconing; réveil périodique, vérification de l'arrivé d'un beacon (jeton pour accéder au canal).
- Economise de l'énergie à travers la mise en veille entre deux beacons, et les nœuds ne devant pas router ou recevoir les données aléatoirement peuvent se mettre en veille.
- Assure une transmission fiable de données
- Offre une sécurité AES-128

Le Beaconing est un mécanisme qui identifie le réseau et indique la présence de la donnée.
Il est optionnel, présent uniquement lorsque le réseau est actif et décrit la super-trame qui est composée de deux parties :

- Inactive: dans laquelle toutes les stations dorment
- Active: avec une Période active composée de 16 slots.

5.3. Le protocole S-MAC (Sensor-MAC) :

Le protocole S-MAC, [37] est basé sur la méthode CSMA/CA et repose sur le mécanisme RTS (Request To Send)/CTS (Control To Send) dans le but de traiter le problème des nœuds cachés. Ce protocole introduit une période d'activité et de mise en veille.
Pour pouvoir communiquer, les nœuds doivent être synchronisés et organisés en clusters virtuels, dont un représente le chef et se nomme Cluster Head « CH » et les autres sont vus comme des membres.
Lors de la transmission d'un long message, le protocole S-MAC utilise le message passing qui permet d'envoyer les fragments de celui-ci sous forme de rafale.

Chaque nœud diffuse périodiquement aux autres nœuds du cluster sa période d'activité et celle de mise en veille sous forme de programme de scheduling dans un paquet SYNC.

S-MAC ajoute dans chaque fragment et dans chaque ACK la durée restante de la transmission Ce qui permettra aux nœuds qui se réveillent au milieu de la transmission de se rendormir.

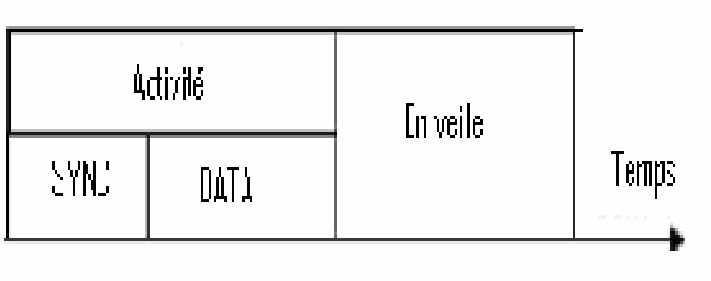

Figure 17 : La période d'activité et de mise en veille dans S-MAC [37].

S-MAC est considéré comme étant un protocole simple, avec une réduction de la perte énergétique pendant l'écoute du canal libre, ceci est réalisé par l'introduction du cycle de mise en veille.

Le problème qui se pose c'est que le protocole S-MAC n'assure pas de scalabilité c'est-à-dire que l'augmentation du nombre de nœuds implique directement l'augmentation du nombre de schedule sauvegardé dans chaque nœud.
Et comme le principe ici est que les messages doivent attendre la période d'écoute pour être envoyés, alors cela engendrerait une augmentation de la latence.

L'utilisation d'une période d'écoute fixe qui est déterminée avant le déploiement des nœuds cause l'écoute d'un canal libre quand il n'y a pas de données à transmettre pendant cette période. Apres ces problèmes étudiés précédemment, le protocole T-MAC est utilisé pour remédier aux problèmes de la période fixe dans S-MAC [34].

5.4. T-MAC (Timeout MAC) :

Dans ce protocole et contrairement à S-MAC, la période d'écoute se termine quand il n'y a plus d'activité sur le canal pendant une période adaptive TA, qui représente le temps minimum d'écoute d'un canal libre.

En utilisant cette période adaptive, les nœuds sauvegardent leurs énergies en minimisant l'écoute du canal libre

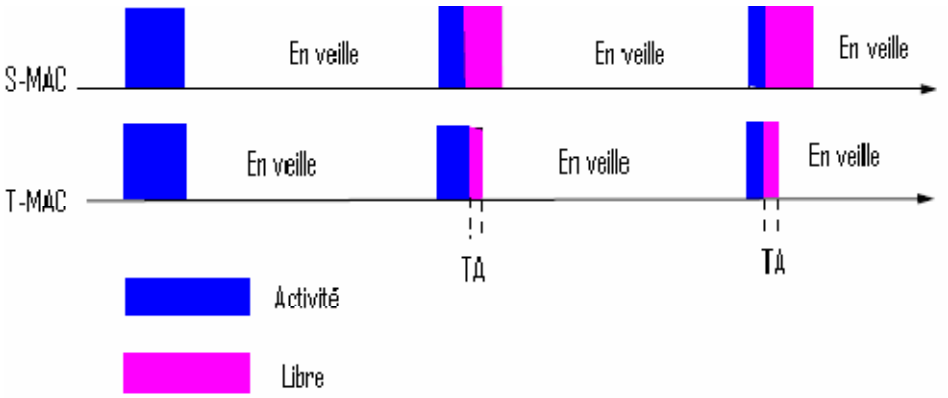

Figure 18 : La comparaison des deux protocoles

5.4.1. Avantages de ce mode d'accès :

T-MAC conserve l'énergie plus que S-MAC puisqu'il évite l'écoute d'un canal libre même dans la période d'écoute.

Il est aussi simple que le protocole S-MAC.

T-MAC contient beaucoup de messages de contrôle (overhead).

5.4.2. Inconvénients :

T-MAC comme S-MAC n'est pas scalable c'est-à-dire que l'augmentation du nombre de nœuds implique directement l'augmentation du nombre de schedule sauvegardé dans chaque nœud.

6. Prolonger la vie des réseaux de capteurs sans fil Par une mise en veille adaptative

6.1. Le protocole ASLEEP (Adaptive staggered SLEEP) :

6.1.1. La description du protocole ASLEEP :

En effet, le protocole ASLEEP ajuste dynamiquement les programmes de mise en veille des nœuds (régler automatiquement la longueur de l'intervalle TI) et n'exige pas a priori la connaissance de la topologie du réseau.

Ici, les nœuds capteurs sont supposés former une arborescence logique de routage pour la transmission de données. La communication entre un parent et ses fils se produit avec ce qu'on appelle Communication Period (CP) qui se répète périodiquement.

Chaque CP est divisé en deux parties:

- un intervalle dit Talk Interval (TI), dans lequel les nœuds communiquent en utilisant le protocole MAC.

- un intervalle dit Silence Interval (SI) dans lequel les nœuds sont endormis (en veille).
- Chaque TI partage avec les nœuds une relation parent-fils.

7. Conclusion :

Nous nous sommes intéressés dans ce chapitre à la contrainte énergétique et l'amélioration de la durée de vie des réseaux de capteurs.

En effet, ces réseaux présentent un intérêt considérable et une nouvelle étape dans l'évolution des technologies de l'information et de la communication.

Cette nouvelle technologie suscite un intérêt croissant vu la diversité de ces applications : santé, environnement et industrie.

Cependant, nous avons remarqué que plusieurs facteurs et contraintes compliquent la gestion de ce type de réseaux qui se caractérisent par une capacité énergétique limitée rendant l'optimisation de la consommation d'énergie dans les réseaux une tâche critique pour prolonger la durée de vie du réseau.
Ce problème est causé par le fait que les réseaux de capteurs sont déployés dans des environnements inaccessibles et que leurs batteries à faible puissance ne peuvent être ni rechargées ni remplacées.

Donc, nous avons présenté dans ce chapitre quelques techniques de mise en veille qui permettent de mieux gérer la consommation d'énergie afin de maximiser la duré de vie du réseau.

Chapitre IV :

La conception et l’implémentation du protocole LEACH et de la solution proposée

1. Introduction

L'objectif primordial de notre travail est l'amélioration de la technique de mise en veille proposée par le protocole standard LEACH « Low Energy Adaptive Clustering Hierarchy » et l'implémentation d'une nouvelle version de celui-ci baptisée PWLEACH « PoWer LEACH» qui permet un gain en énergie en se chargeant de réduire le nombre de nœud mis en veille pour réduire la consommation des Cluster_Head afin d'aboutir par la suite à un chef suivi de son sous chef.
En effet, l'inconvénient majeur du protocole LEACH est que le Cluster_Head épuise son énergie à un certain moment donné ou peut tomber en panne, donc les nœuds deviendront autonomes et que chacun d'entre eux se chargera lui-même de l'envoie d'informations récoltées à la station de base, chose qui augmente la consommation d'énergie et diminue la survie de vie du réseau.

Apres avoir étudié les problèmes de ce protocole, nous avons proposé une nouvelle approche qui prend en considération les inconvénients du protocole LEACH pour pouvoir l'améliorer en proposant une nouvelle version.

Les simulations effectuées ont démontré l'efficacité de cette approche par rapport au protocole précédant « le LEACH » en termes de mise en veille et d'énergie en passant par la réduction de nœuds mis en veille via l'implémentation et la simulation des deux protocoles.

Dans ce chapitre, nous entamons notre approche par l'exposition d'outils nécessaires à l'implémentation et à la simulation de ces deux protocoles, ensuite, nous décrirons le fonctionnement du protocole LEACH que nous avons implémenté en montrant par la suite l'efficacité de l'approche proposée par rapport à LEACH ainsi que son apport lié à notre thème.

Nous achevons ce chapitre par une représentation des résultats prélevés lors des simulations.

2. Environnement de simulation

Dans ce qui suit, nous présenterons les outils utilisés lors de l'implémentation des deux protocoles en commençant tout d'abord par TinyOS, le système d'exploitation conçu pour les RCSF, puis nous parlerons du langage de programmation NesC avec lequel nous avons effectué notre programmation.

Nous concluons cette section par la présentation d'un simulateur adéquat aux réseaux de capteur sans fil TOSSIM qui offre deux mécanismes permettant d'émuler le réseau :

- Une interface graphique TinyViz pour visualiser le déroulement de la simulation.
- Un simulateur PowerTossim pour simuler et évaluer la consommation d'énergie.

2.1. TinyOS

De nombreux contributeurs ont développé un système d'exploitation dit Tinyos destiné au RCSF afin de faciliter l'implémentation et l'exécution de protocoles dédiés à ce type de réseaux. Celui-ci 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.

2.1.1. Pourquoi un nouveau système d'exploitation :

Les OS classiques sont généralement conçus pour un usage générique. Ils sont ainsi conçus en supposant une disponibilité sans limite des ressources. Leur objectif est la facilité d'usage, la rapidité et l'efficacité, mais ils ne sont pas appropriés aux "motes" (noeuds capteurs), vu que ces derniers sont caractérisés par :

- Des ressources énergétiques basses
- Une mémoire limitée avec une CPU lente
- Une réduction de la taille mémoire
- Une communication radio qui se base sur une faible bande-passante avec une portée radio courte.

La solution était d'utiliser un système d'exploitation plus approprié : « le TinyOS » qui se caractérise par Concurrence :

- Une architecture orientée événement avec un concept de modularité

- Une Application composée de composants

- Un OS et une application compilés en un seul exécutable

- L'utilisation d'un modèle event/command

- L'utilisation d'un ordonnancement FIFO non préemptif

2.1.2. Les propriétés de la plate forme TinyOS :

- Le langage : TinyOS a été programmé en langage NesC que nous allons détailler plus tard.

- La disponibilité et sources : TinyOS est un système disponible, principalement développé par l'université américaine de Berkeley, qui l'offre en téléchargement sous la licence BSD et en assure le suivi.

- Préemptif :

Le TinyOS ne gère pas de 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 peut stopper l'exécution d'une tâche.

- Event-driven :

Le TinyOS est basé sur un fonctionnement évènementiel (event-driven) qui s'appuie sur la gestion des évènements qui se produisent.

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.

- La Consommation énergétique :

TinyOS a été conçu dans le but de réduire au maximum la consommation en énergie avec la procédure de mise en veille d'un capteur lorsqu'aucune tâche n'est active.

Nous allons à présent résumer ces propriétés dans un tableau général :

Propriétés	Valeurs pour TinyOS
Type	Event-driven
Disponibilité	Open-source
Langage	NesC
Préemptif	Non
Temps Réel	Non
Sources	Fournies

Tableau3 : Propriété de TinyOS

2.1.2. 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 puisqu'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 depuis 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 (Le mot dynamique est souvent employé désigner ou qualifier ce qui est relatif au mouvement. Il peut être employé comme :) de mémoire et pas de pointeurs de fonctions. Bien sur cela simplifie grandement l'implémentation (Le mot implantation peut avoir plusieurs significations :). 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.

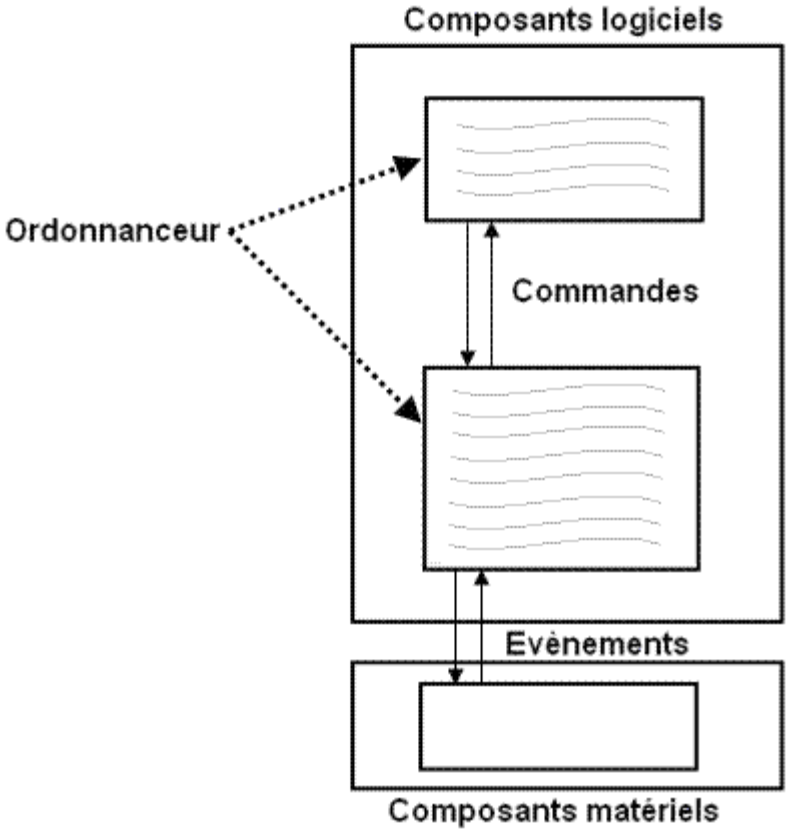

Figure 19: la gestion mémoire au niveau de TinyOS

2.1.3. La structure logicielle de Tinyos :

Le système d'exploitation TinyOS s'appuie sur le langage NesC qui 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. Nous allons détailler ceux-ci dans le tableau 4-4 suivant :

Action	Utilisation
Tâche	Travaux de longue durée
Commandes	Exécution d'une fonctionnalité précise dans un autre composant.
Evénement	Equivalent logiciel (Un logiciel ou une application est un ensemble de programmes, qui permet à un ordinateur ou à un système informatique d'assurer une tâche ou une fonction en particulier (exemple :...) à une interruption matérielle.

Tableau 4: Structure logicielle de Tinyos

2.1.4. Le Package TinyOS :

TinyOS fonctionne sur diverses plateformes, disponibles dès l'installation, à partir d'un environnement Windows (2000 et XP) ou bien GNU/Linux

La version utilisée dans notre projet est la version Tinyos-1.x: qui en court de test. Nous focalisons nos recherches sur cette version sous linux plus exactement sur Ubuntu 10.10. GNU/Linux : offre des packages RPM qui sont proposés au téléchargement, avec un guide qui explique la marche à suivre.

2.2. Le langage NesC (Network Embedded System C) :

Le langage NesC [41] est vu comme étant un dialecte du langage de programmation C qui été désigné pour faciliter l'implémentation des structures et des composants TinyOS.

Dans le concept de NesC il y a une séparation entre la construction et la composition.

Les programmes sont construits à l'intérieur des composants qui sont ensuite reliés entre eux pour former toute une application.

Les composants NesC utilisent et fournissent des interfaces bidirectionnelles qui représentent les seuls points d'accès pour les composants TinyOS [44].

2.2.4. Organisation des concepts dans NesC :

En effet, L'unité de code de base de NesC est le composant "component" qui exécute des commandes ou lance des événements. Une application en NesC comporte les éléments suivants :

- Le Module : est un composant qui implémente une ou plusieurs interfaces et peut utiliser une ou plusieurs interfaces
- La Configuration : est un composant qui relie d'autres composants ensemble.

- L'Interface : définie de manière abstraite les interactions entre deux composants

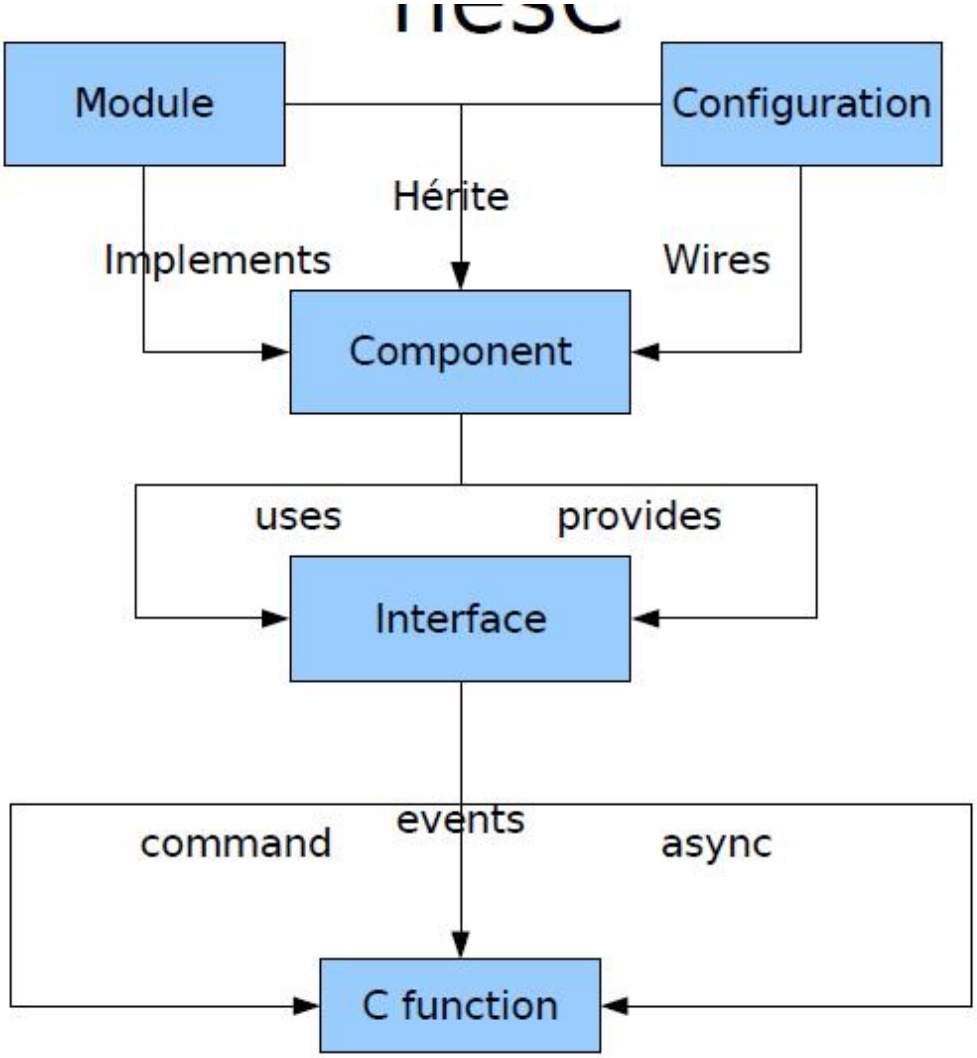

Figure 20: L'organisation de concepts dans NesC

2.2.5. Exemple d'un programme en NesC :

Celui-ci consiste à allumer la Led jaune chaque seconde.

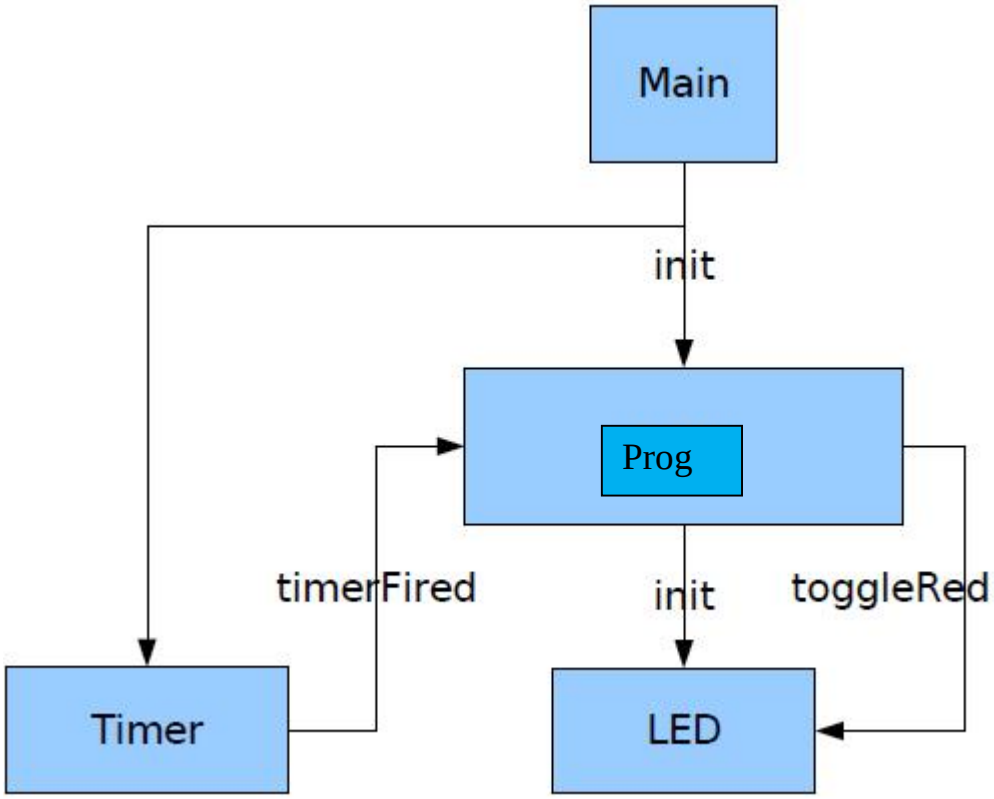

Figure 21: L'architecture du programme qui affiche la Led chaque seconde

Le code source du programme :

```
Prog.nc
configuration Prog {
}
implementation {
    components Main, ProgM, SingleTimer, LedsC;
    Main.StdControl -> ProgM.StdControl;
    Main.StdControl -> SingleTimer.StdControl;
    ProgM.Timer -> SingleTimer.Timer;
    ProgM.Leds -> LedsC;
}
```

L'interface StdControl:

```
StdControl.nc
interface StdControl {
    command result_t init();
    command result_t start();
    command result_t stop();
}
```

Le module ProgM.nc :

```
module ProgM {
provides {
    interface StdControl;
}
uses {
    interface Timer;
    interface Leds;
}
command result_t StdControl.init() {
call Leds.init();
return SUCCESS;
}
command result_t StdControl.start() {
return call Timer.start(TIMER_REPEAT,1000) ;
}
command result_t StdControl.stop() {
return call Timer.stop();
}
event result_t Timer.fired()
```

```
{

call Leds.redToggle();
return SUCCESS;
}
}
}
```

L'interface Timer:

```
Timer.nc
interface Timer {
command result_t start(char type, uint32_t interval);
command result_t stop();
event result_t fired();
}
```

2.3. Tossim : le simulateur de TinyOS :

TOSSIM [42] : le simulateur de TinyOS est basé sur la programmation événementielle qui a été conçue et désignée pour simuler les réseaux de capteurs qui utilisent la plateforme TinyOS.

Il permet de simuler le comportement d'un capteur (envoie/réception de messages via les ondes radios, traitement de l'information, ...) au sein d'un réseau de capteurs.

Son but principal est de créer une simulation très proche de ce qui se passe dans ces réseaux dans le monde réel.

En effet, TOSSIM fournit deux modèles de radios pour la communication dont le modèle par défaut est « le simple ».

Les paquets dans le réseau sont transmis sans erreur et sont reçus par chaque nœud. Il est possible pour ce modèle que deux nœuds différents puissent envoyer leurs paquets en même temps ce qui cause le chevauchement de signaux ainsi que la destruction de ces paquets.

Le deuxième modèle est le modèle « lossy » dans lequel les nœuds sont placés dans un graphe directe formé d'un couple (a, b) ce qui signifie qu'un paquet envoyé par le nœud a peut être reçu par le nœud b.

Pour suivre la dépense d'énergie on utilisera un autre simulateur communément appelé « PowerTossim ».

2.4. Le PowerTossim :

Cet outil permet de faire des simulations de la même manière que TOSSIM sauf que celui-ci prend en considération la consommation énergétique, ainsi le nœud qui ne possède plus d'énergie s'arrête de fonctionner, ce qui nous permet d'exécuter la simulation jusqu'à la mort du réseau.

2.5. L'interface TinyViz :

L'interface TinyViz est fournie avec le système d'exploitation TinyOS. Il s'agit d'une interface graphique programmée en java qui représente un réseau de capteurs émulés grâce à TOSSIM [43].
Si vous avez utilisé l'installeur Ubuntu, vous pouvez immédiatement le lancer par la commande : ./tinyviz qui affichera alors une interface non connectée, pour cela il faut accéder au dossier Blink dont le chemin est le suivant :

/opt/tinyos-1.x/apps/**Blink**

Puis tapez les commandes suivantes:

1) make pc
2) Export DBG= usr1
3) Build/pc/main.exe suivi du nombre de nœuds voulu
4) Tinyviz –run build/pc/main.exe suivi du nombre de nœuds

Et voila ce qui s'affiche à l'écran, une interface avec par exemple 10 nœuds :

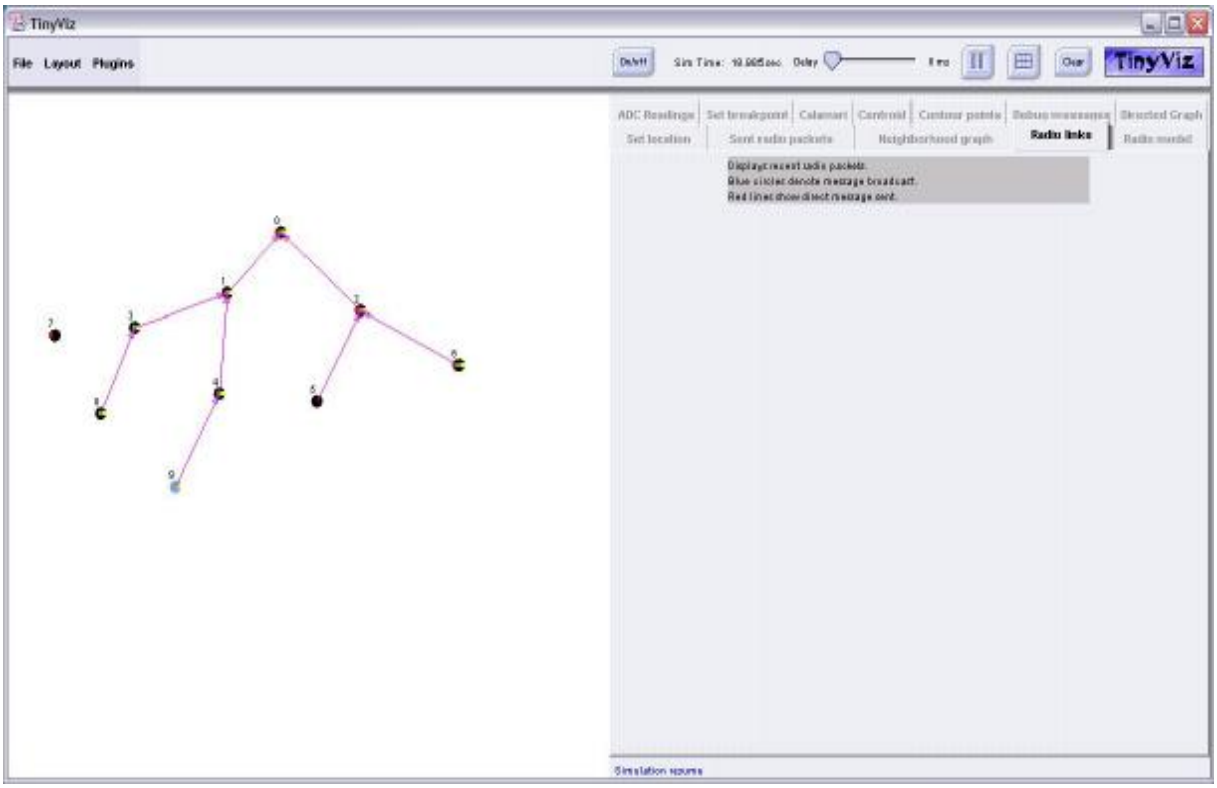

Figure 22 : L'interface TinyViz avec 10 nœuds

Remarque :

- La commande make pc est utilisée pour compiler l'application Blink qui se trouve dans le dossier apps au niveau de Tinyos.

- La commande export DBG=usr1 comporte deux mots clés expliqués comme suit:

Le mot DBG est une variable d'environnement dynamique utilisée par le système d'exploitation Tinyos. Elles servent à communiquer des informations entre programmes qui ne se trouvent pas sur la même ligne hiérarchique.

Il existe plusieurs variables d'environnement telles que power, led, clock, route, packet et notamment la variable usr1 qui représente la variable de sortie utilisée selon le besoin de l'application.

- La troisième commande : Build/pc/main.exe suivi du nombre de nœuds voulu qui

permet de montrer le résultat de la simulation de Tossim qui crée un exécutable main.exe se trouvant dans le dossier Build/pc.

- La commande Tinyviz –run build/pc/main.exe suivi du nombre de nœuds permet de

faire apparaître l'interface Tinyviz afin de montrer le comportement des nœuds de façon à avoir une simulation proche de la réalité.

3. Implémentations et déroulements

Afin d'implémenter notre approche dans le but de prolonger la durée de vie du réseau, notre travail s'est déroulé en trois phases :

- Ø Implémentation du protocole LEACH.
- Ø La présentation de l'approche proposée PW-LEACH.
- Ø La comparaison des relevés énergétiques des deux protocoles afin de montrer l'efficacité de notre approche.

3.1. Implémentation du protocole LEACH :

Dans cette section, nous décrivons les structures de données nécessaires pour l'implémentation du protocole LEACH. Le paquet dans TinyOS est envoyé dans une structure appelée TOS_Msg, qui est contenue dans un champ «uint8_t data[TOSH_DATA_LENGTH] ».

Les structures de données du paquet diffèrent selon le rang du nœud (Sink, CH ou membre).

A. Le nœud Sink :

```
Typedef struct PUITS
{
uint16_t ID ; // pour identifier le nœud Sink qui correspond à tos_local_address=0
uint8_t round ; // pour indiquer le round ou cycle courant
float probability ; //la probabilité que chaque nœud devienne Cluster-Head « CH »
uint8_t Depth ; // la puissance du signal d'un CH dans le réseau
}PUITS ;
```

B. Le nœud membre :

```
Typedef struct MEMBER
{
uint16_t ID_MEMBER ; // pour identifier chaque membre qui correspond à
tos_local_adress
uint16_t ID_CH ; // pour identifier le CH auquel appartiendra le nœud membre
uint8_t temp ;   // la température captée
}MEMBRE ;
```

3.1.1. Le déroulement du protocole LEACH :

Nous nous sommes focalisés dans cette section sur les phases de déroulement de l'algorithme LEACH via l'interface TinyViz spécifique à TinyOS. En effet, ce protocole se décompose en trois phases fondamentales :

- Le déclenchement du nouveau cycle « round », et l'annonce des CH
- La création de cluster et l'envoie de températures captées aux CH
- L'envoie de résultats de température au nœud Sink

A. Le déclenchement du nouveau cycle « round », et l'annonce des CH :

D'après l'illustration 4-20, nous constatons une transmission broadcast effectuée par les nœuds en termes de cercle bleu. Ainsi, pour l'annonce d'un nouveau round, le nœud Sink envoie un broadcast aux différents nœuds voisins qui eux même prennent le relais en envoyant à leur tour une autre transmission broadcast aux autres nœuds.

Fonctionnement du protocole LEACH :

La figure montre aussi que, le nœud dont la LED a été activée en rouge est élu Cluster_Head en fonction de son pourcentage énergétique et le nombre de fois dans lequel il occupait le statut de Cluster_Head dans les cycles précédents.

Le choix du Cluster_Head se fait selon un jeu aléatoire qui s'effectue de la manière suivante :

Chaque nœud choisit un nombre aléatoire compris entre 0 et 1, puis si le nombre choisi est inférieur à la valeur fixée par la formule Pi(t) alors il sera élu Cluster_Head sinon il gardera le statut de membre

Suite à l'élection du Cluster_Head, celui-ci envoie une information aux différents nœuds pour leurs annoncer son statut.

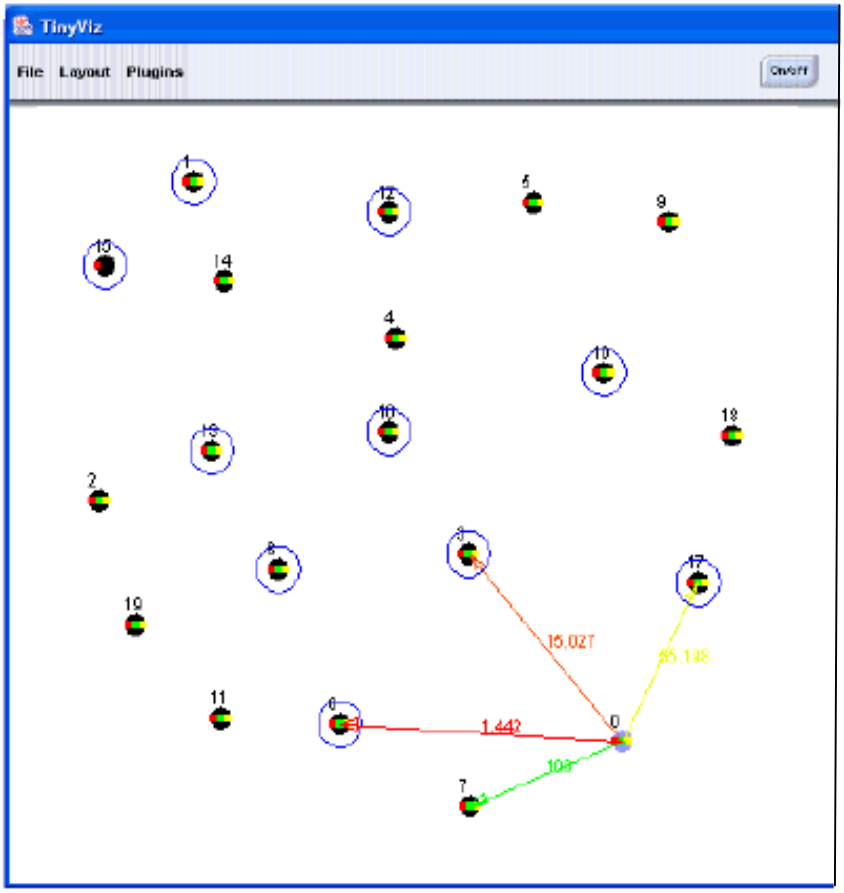

Figure 23: Déclenchement et relais et nouveau round, et l'annonce des CH

B. La création de cluster et l'envoie de températures captées aux CH :

Apres l'élection du Cluster_Head, une transmission unicast est représentée par une flèche pour permettant aux autres nœuds n'ayant pas le statut de chef de répondre à l'annonce du CH le plus proche. Les clusters se forment dans le réseau comportant un CH accompagné de membres.

Cette seconde étape permet aux membres de chaque cluster du réseau de capter la température puis chacun attend le début de son slot pour qu'il puisse l'envoyer aux différents Cluster_Head.

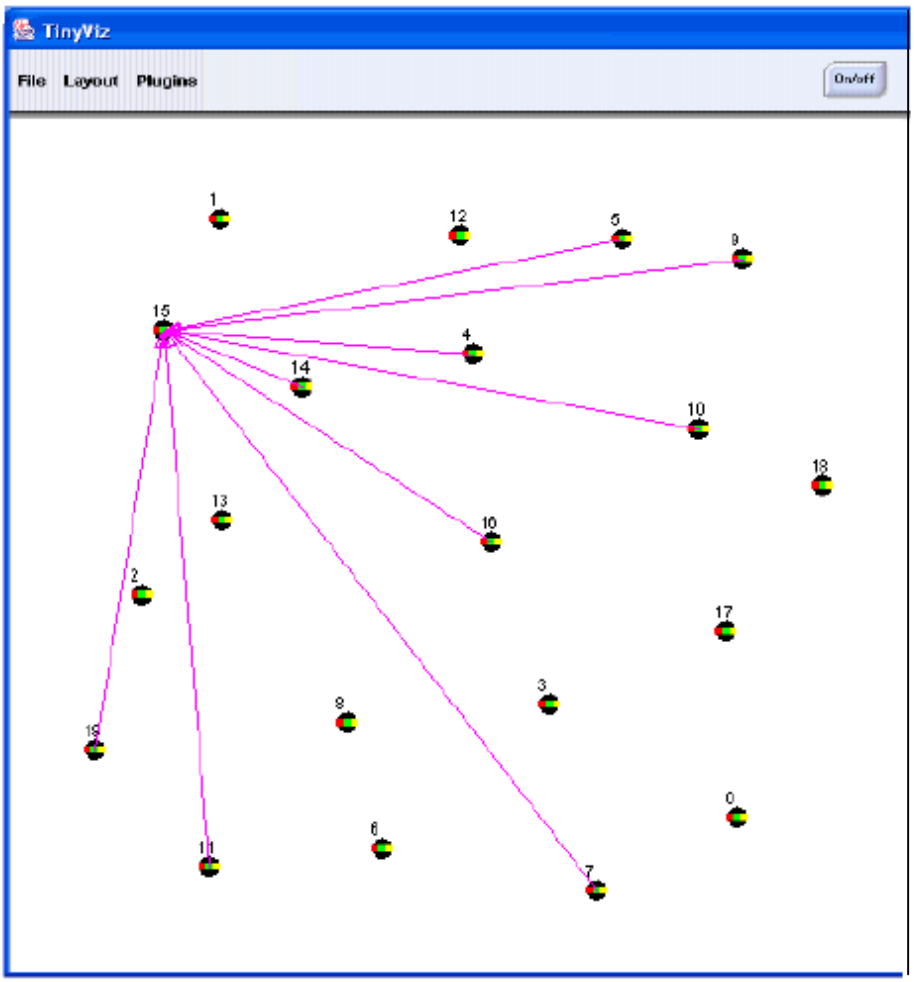

Figure 24 : La création de Cluster et l'envoie de température.

C. L'envoie de résultats de température au nœud Sink :

Comme le montre l'illustration juste en dessous, le Cluster_Head numéro 15 agrège les températures reçues et les envoient au nœud Sink qui par le biais d'internet ou satellite les diffuse à un ordinateur central.

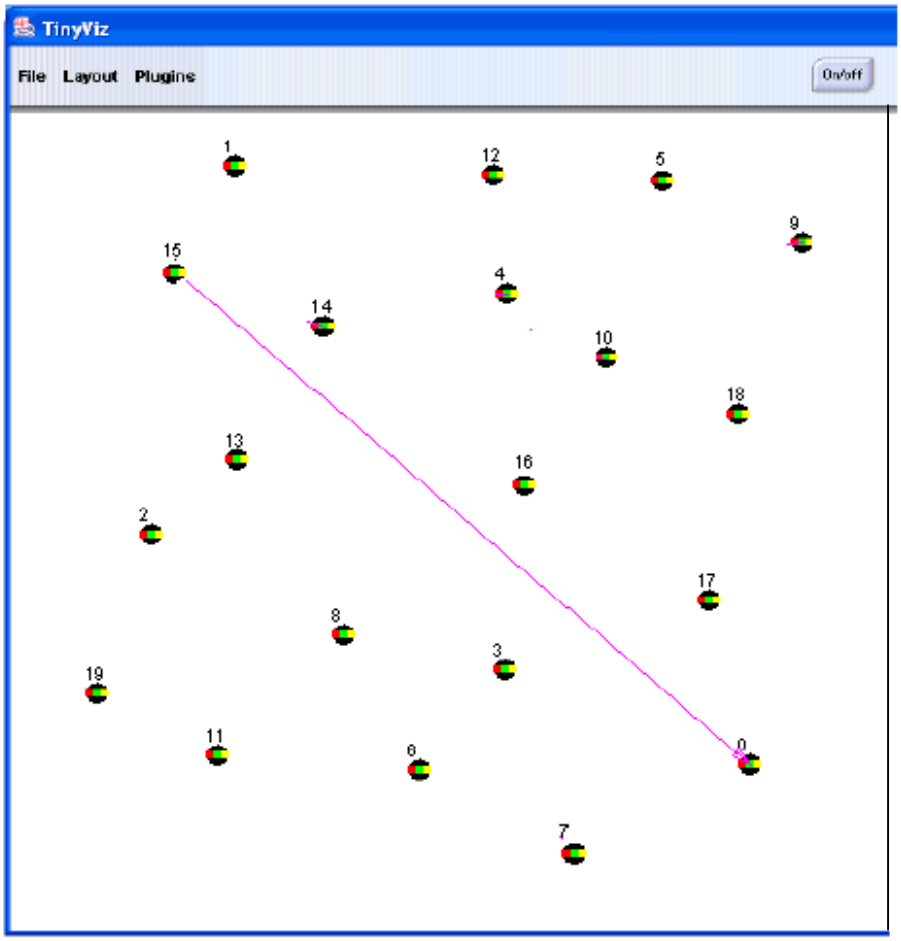

Figure 25 : L'envoie de résultats au nœud Sink

3.2. L'algorithme détaillé du protocole LEACH :

L'algorithme se déroule en « rounds dit aussi cycles » qui possèdent approximativement le même intervalle de temps déterminé au préalable. Chaque cycle est constitué d'une phase d'initialisation et d'une phase de transmission.

3.2.1. Phase d'initialisation

Comme l'indique la figure IV-4, la phase d'initialisation est composée de 3 sous-phases: d'annonce, d'organisation des groupes et enfin d'ordonnancement, et qui seront détaillée ci-dessous.

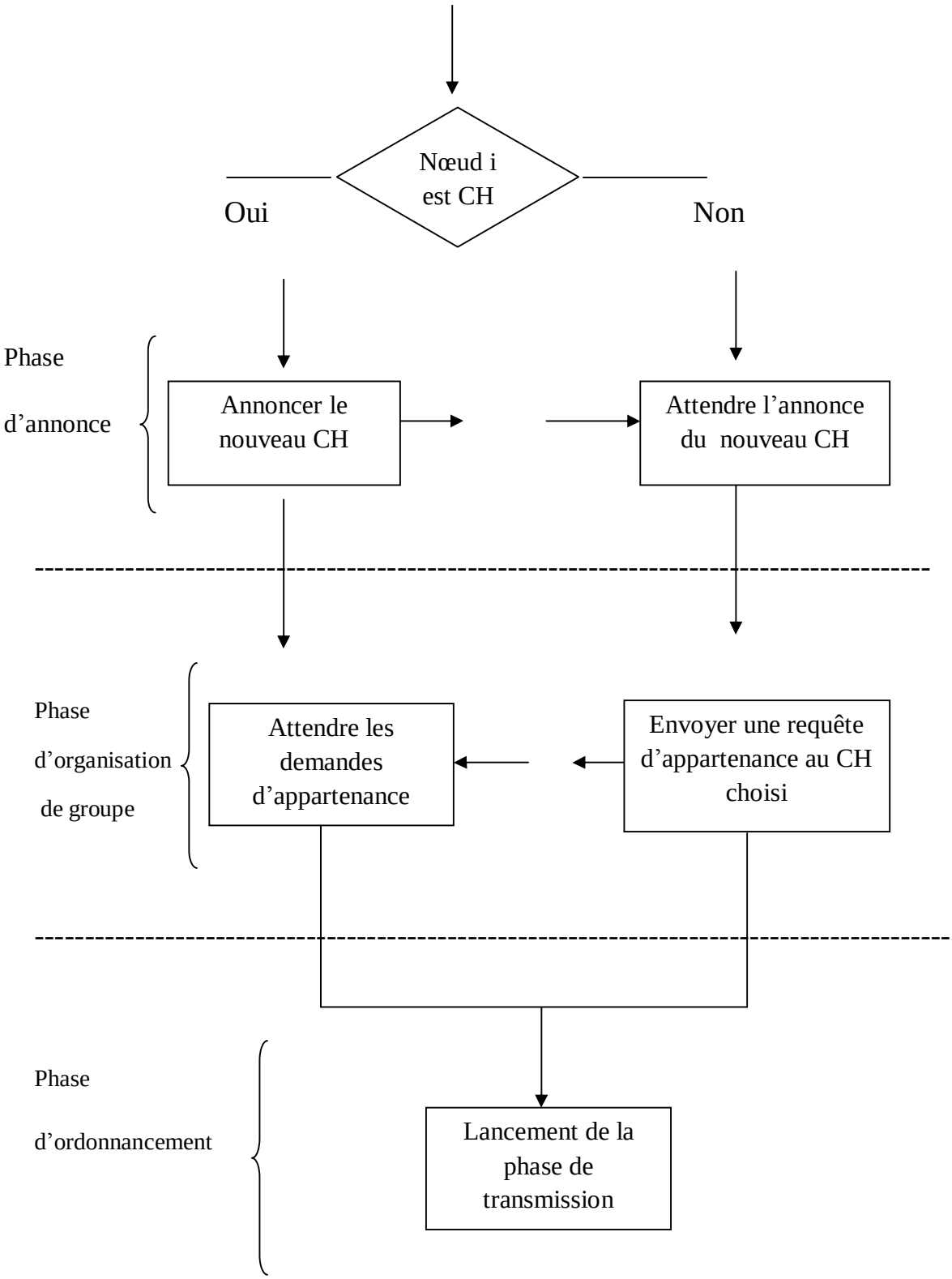

Figure 26: Opérations de l'étape d'initialisation de LEACH [54].

3.2.1.1. Phase d'annonce:

Avant de procéder à cette phase, nous désirons avoir un certain nombre de CH que l'on note par K, celui-ci est fixe et étant inchangé durant tous les rounds. Nous estimons que le pourcentage optimal du nombre de CH désiré devrait être de 5% à 15% du nombre total de nœuds [54].

Si ce pourcentage n'est pas respecté, cela nous mènera à une grande dissipation d'énergie dans le réseau.

- En effet, si le nombre de CH est très élevé, on aura un nombre important de nœuds

(CH) qui se consacrent aux tâches très couteuses en ressources énergétiques.

Ainsi, on aura une dissipation d'énergie considérable dans le réseau.

- De plus, si le nombre de CH est très petit, ces derniers vont gérer des groupes de

grandes tailles. Ainsi, ces CH s'épuiseront rapidement à cause du travail important qui leur est demandés.

Cette phase commence par l'annonce du nouveau cycle par le nœud puits généralement le nœud 0, et, par la prise de décision locale d'un nœud pour devenir CH avec une certaine probabilité Pi(t) [55].

En effet, le nombre de Cluster_Head s'effectue avec la formule suivante :

Nombre(CH)= $\sum_{i=1}^{N} Pi(t)$= K

Où N est le nombre total de nœuds dans le réseau. Si on a N nœuds et K CH, alors, il faudra

N/K rounds durant lesquels un nœud doit être élu seulement une seule fois autant que CH

avant que le round soit réinitialisé à 0. Donc la probabilité de devenir CH pour chaque nœud i est :

$$Pi(t) = \frac{Le\ nombre\ de\ Cluste\ Head\ désiré}{Le\ nombre\ de\ noeuds\ qui\ n'ontpas\ encore\ été\ élus\ Cluster\ Head\ durant\ les\ r\ round\ précédents}$$

$$Pi(t) = \begin{cases} \dfrac{K}{N - K * \left(r \bmod \frac{N}{K}\right)} & Ci(t) = 1 \\ 1 & Ci(t) = 0 \end{cases}$$

Où Ci(t) égal à 0 si le nœud i a déjà été CH durant l'un des (r mod N/K) rounds précédents, et, il est égal à 1 dans le cas contraire. Donc, seuls les nœuds qui n'ont pas encore été CH, ont vraisemblablement une énergie résiduelle suffisante que les autres et ils pourront être choisis.

3.2.1.1. Phase d'organisation de groupe :

Après qu'un nœud soit élu CH, il doit informer les autres nœuds membres de son nouveau rang dans le round courant. Pour cela, un message d'avertissement ADV contenant l'identificateur du CH est diffusé à tous les nœuds en utilisant le protocole MAC CSMA pour éviter les collisions entre les CH. La diffusion permet de s'assurer que tous les nœuds non-CH ont reçu le message. La décision est basée donc sur l'amplitude du signal reçu; le CH ayant le signal le plus fort (i.e. le plus proche) sera choisi. En cas d'égalité des signaux, les nœuds non-CH choisissent aléatoirement leur CH [55]. Chaque membre informe son CH de sa décision. Une fois que le CH ait reçu la demande, il lui envoie un message d'acquittement Join- REQ.

3.2.1.2. Phase d'ordonnancement

Après la création des groupes, chaque CH agit comme un centre de commande local pour coordonner les transmissions des données au sein de son groupe.

Par ailleurs, pour de éviter les interférences entre les transmissions dans des groupes adjacents, le CH choisit aléatoirement un code dans une liste de codes de propagation CDMA qui le transmet par la suite à ses membres afin de l'utiliser pour leurs transmissions [56].

3.2.1. Phase de transmission:

Cette phase permet de collecter les données captées qui seront ensuite agrégées par les CH qui les fusionnent et les compressent, et, envoient le résultat final au nœud puits. Après un certain temps prédéterminé, le réseau va passer à un nouveau round. Ce processus est répété jusqu'à ce que tous les nœuds du réseau soient élus CH, une seule fois, tout au long des rounds précédents. Dans ce cas, le round est réinitialisé à 0.

3.3. L'implémentation de l'approche proposée « PWLEACH »:

Dans ce qui suit, nous allons donner les structures de données ainsi que les principaux événements et commandes nécessaires pour l'implémentation de la solution.

3.3.1. Les structures de données :

En se basant sur les champs utilisés dans les structures de données mises en place dans le protocole LEACH, cette approche a besoins d'informations suivantes :

A. Le nœud Sink :

```
Typedef struct PUITS
{
uint16_t ID ; // pour identifier le nœud Sink qui correspond à tos_local_address=0
uint8_t round ; // pour indiquer le round ou cycle courant
float subprobability ; //la probabilité que chaque nœud devienne sous Cluster-Head
« sousCH »
uint8_t SubDepth ; // la puissance du signal d'un sous CH dans le réseau
}PUITS ;
```

B. Le nœud non cluster head :

```
Typedef struct MEMBRE
{
uint16_t ID_MEMBRE; // pour identifier chaque nœud qui correspond à
tos_local_adress
uint16_t ID_sousCH ; // pour identifier le sous CH auquel appartiendra le nœud membre
uint8_t temp ;   // la température captée
}MEMBRE ;
```

C. Le nœud SousCluster_Head :

```
Typedef struct sousCLUSTER_HEAD
{
uint16_t IDENT_sousCH ;   // pour identifier chaque sous CH qui correspond à
tos_local_address
uint16_t sousIDENT_NODE ;  //indique l'identification du membre qui appartiendra au
groupe
uint8_t data_ATT ;  //le slot attribué à chaque membre
uint16_t FREQ ;  // la fréquence nécessaire pour que le membre envoie sa donnée
}sousCLUSTER_HEAD ;
```

Le fonctionnement de la nouvelle version modifiée de LEACH :

Le PWLEACH est une nouvelle approche modifiée qui se base sur le même fonctionnement que le protocole « LEACH » et repose sur les trois phases fondamentales :

- Le déclenchement du nouveau Round et l'annonce du Cluster_Head et du sous

Cluster_Head selon la loi probabiliste.

- La création de Cluster et l'envoie d'informations captées aux Cluster_Head et aux

sous Cluster_Head qui eux même se mettent d'accord pour se répartir les tâches de manière équitable et permettre au Cluster_Head d'alléger sa charge grâce au sous Cluster_Head qui effectue presque la moitié du travail du Cluster_Head.

En effet, la charge du sous Cluster_Head s'effectue selon sa capacité énergétique, c'est-à-dire que si celle-ci est approximativement la même que celle du chef, les tâches se répartissent de manière équitable, sinon se serait en fonction de la capacité.

- L'envoie de résultat de température au nœud Sink.

Le choix de l'élection du Cluster_Head et sous Cluster_Head s'effectue comme suit :

Prenons l'exemple de deux nœuds A et B qui chacun prend un nombre aléatoire compris entre 0 et 1, si le nombre choisi par les deux nœuds A et B est inférieure à la valeur fixée par la nouvelle formule modifiée Pi(t), alors un autre test s'ajoute pour pouvoir désigner le chef et le sous chef.

Il faut par la suite effectuer une comparaison entre les deux nombres choisis par les nœuds A et B. Le plus petit nombre permet au nœud d'occuper le statut de Cluster_Head. Le plus grand nombre permet au nœud d'occuper le statut de sous Cluster_Head.

Le résultat de cette approche a bien révélé une réduction de la consommation énergétique par rapport à la première version, car la charge du Cluster_Head a été allégée en prenant en considération la notion de sous Cluster_Head qui fonctionne de manière parallèle avec le CH et évite le disfonctionnement de celui-ci rapidement.

4. Récupération de traces:

Après implémentation et lancement de simulation, une variété de traces peut être récupérée selon la caractéristique de l'information voulue :

Par exemple :

1. <u>en ce qui concerne la **led** : « mote leds » voila ce qui s'affiche à nous</u>

sarah@ubuntu:/opt/tinyos-1.x/apps/LEACH$ export DBG=led

sarah@ubuntu:/opt/tinyos-1.x/apps/LEACH$ build/pc/main.exe 5

SIM: Random seed is 809814

4: LEDS: Red on.

4: LEDS: Red off.

4: LEDS: Green on.

4: LEDS: Yellow on.

2: LEDS: Red on

3: LEDS: Red off.

Exiting on SIGINT at 0:4:17.28158275

2. <u>En ce qui concerne **am** : active message transmission / reception</u>

sarah@ubuntu:/opt/tinyos-1.x/apps/LEACH$ export DBG=**am**

sarah@ubuntu:/opt/tinyos-1.x/apps/LEACH$ build/pc/main.exe 5

SIM: Random seed is 981594

: Sending message: ffff, faff ff fa 7d 0c 00 00 00 00 cd cc cc 3d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

: AM_address = ffff, fa;

counter:1: Received message: ff ff fa 7d 0c 00 00 00 00 cd cc cc 3d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00 27 0d 00 00

: AM_type = 250

3. En ce qui concerne l'énergie **Power**

sarah@ubuntu:/opt/tinyos-1.x/apps/LEACH$

export DBG=power

sarah@ubuntu:/opt/tinyos-1.x/apps/LEACH$ build/pc/main.exe -p 5

SIM: Random seed is 163575

: POWER: Mote 4 RADIO_STATE ON at 4046290189

: POWER: Mote 1 RADIO_STATE ON at 1693360007

: POWER: Mote 0 RADIO_STATE ON at 334186717

: POWER: Mote 0 RADIO_STATE TX at 3344607156

: POWER: Mote 4 RADIO_STATE TX at 3457230907

: POWER: Mote 0 RADIO_STATE TX at 345723092

: POWER: Mote 2 RADIO_STATE RX at 3457309034

Remarque:

ON: signifie que la radio est allumée.

TX: signifie que la radio est en mode transmission.

RX: signifie que la radio est en mode réception.

5. Le paramétrage des résultats de simulation :

Pou de meilleurs résultats, le paramétrage a été choisi de la manière suivante :

Le nombre de nœuds du réseau	200 nœuds dans le réseau
Le délai de la simulation	500 secondes car les résultats n'étaient pas adéquats avec un délai < 300 secondes
Le nombre d'itération	5 : le nombre d'itération est en moyenne de 5 simulations pour un même scénario

Tableau 5: le paramétrage de la simulation

6. La comparaison des résultats énergétiques générés par ces deux protocoles :

Le fichier énergétique généré par ces deux protocoles est vue comme étant une trace qui se présente comme suit :

Le protocole LEACH	La nouvelle approche
SIM: Random seed is 7409082 : POWER: Mote 0 RADIO_STATE ON at 1453551218702 : POWER: Mote 2 RADIO_STATE RX at 1462600123412 : POWER: Mote 2 RADIO_STATE TX at 1509570344320 : POWER: Mote 0 RADIO_STATE RX at 1510351578732 : POWER: Mote 2 RADIO_STATE TX at 1510370789072 : POWER: Mote 2 RADIO_STATE TX at 1511170798760 : POWER: Mote 0 RADIO_STATE RX at 1511201798762 : POWER: Mote 2 RADIO_STATE TX at 1511970796660 : POWER: Mote 0 RADIO_STATE RX at 1512001905452 : POWER: Mote 2 RADIO_STATE TX at 1512770873250 : POWER: Mote 0 RADIO_STATE RX at 1512801654512 Simulation of 5 motes completed.	. SIM: Random seed is 9329039 : POWER: Mote 0 RADIO_STATE ON at 880254410 : POWER: Mote 0 RADIO_STATE RX at 88114493 : POWER: Mote 3 RADIO_STATE RX at 88115449 : POWER: Mote 9 RADIO_STATE RX at 88116319 : POWER: Mote 9 RADIO_STATE TX at 885856310 : POWER: Mote 10 RADIO_STATE TX at 88585633 : POWER: Mote 3 RADIO_STATE TX at 88585630 : POWER: Mote 0 RADIO_STATE RX at 88593449 : POWER: Mote 9 RADIO_STATE TX at 885936310 : POWER: Mote 10 RADIO_STATE TX at 88593633 : POWER: Mote 3 RADIO_STATE TX at 88593639 : POWER: Mote 9 RADIO_STATE TX at 886016310 : POWER: Mote 10 RADIO_STATE TX at 88601633 : POWER: Mote 3 RADIO_STATE TX at 88601630 : POWER: Mote 0 RADIO_STATE RX at 88601949 Exiting on SIGINT at 0:0:29.32114900.

Tableau 4 : La comparaison entre les deux protocoles en termes d'énergie

Dans lequel :

- ON : signifie que la radio est allumée
- TX : signifie l'état « transmission » de la radio.
- RX : signifie l'état « réception » de la radio.

En prenant en considération plusieurs critères tels que :

La réduction du nombre de nœuds mis en veille, l'ajout d'un nouveau statut qui est le sous chef, et en répartissant la charge des Cluster_Head, nous pouvons constater que le nouveau protocole PWLEACH a amélioré les résultats en termes d'énergie.

L'interprétation de résultat :

Ces résultats sont reproduits dans la figure 4-24 en termes de pourcentage en effectuant une illustration de 10 nœuds.

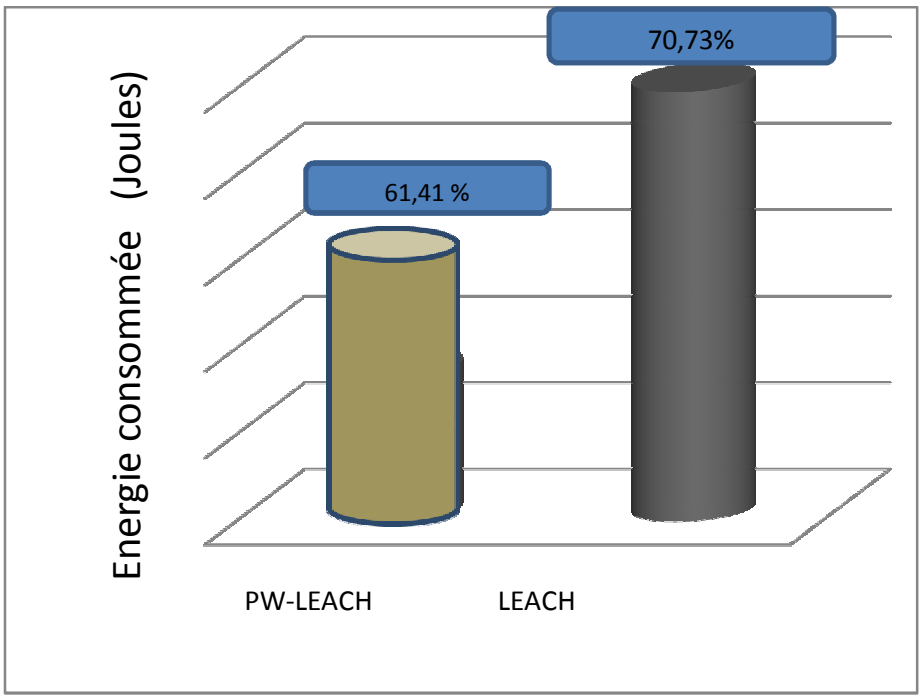

Figure 27: Le pourcentage de la consommation de l'énergie avec un essai de 10 nœuds

7. Conclusion :

Ce chapitre nous a donné la possibilité de se familiariser avec un système d'exploitation destiné aux réseaux de capteurs sans fil qui est le Tinyos accompagné d'un langage de programmation dont on ignorait l'existence.

L'objectif était d'éclaircir et de présenter l'approche proposée et implémentée en passant par une simulation faite sur Tinyos tout en se basant sur la notion de mise en veille afin de minimiser la consommation énergétique qui est l'un des services piliers sur lesquels se base le fonctionnement d'un RCSF.

Nous avons étudié en premier lieu le protocole LEACH et après une série de tests effectuée, nous nous sommes rendu compte que l'énergie consommée dans le réseau était très importante, ce qui n'aidait pas à la maximisation de la durée de vie du réseau.

Après réflexion, l'énergie consommée pouvait être réduite grâce à de petites améliorations via l'introduction de la notion de mise en veille dans les réseaux de capteurs sans fil.

Nous avons focalisé nos recherches sur la nouvelle approche PWLEACH qui prend en considération les inconvénients du protocole LEACH pour essayer de les surmonter, chose qui était faisable car les tests de performance effectués sur la consommation d'énergie ont montré que le PWLEACH répondait bien aux critères de performances souhaités.

Conclusion générale

Conclusion générale

Arrivée aux termes de ce travail, nous pouvons conclure que la problématique de la réduction de la consommation d'énergie et l'amélioration de la durée de vie des réseaux de capteurs sont des objectifs primordiaux ou essentiels.

Ce problème est causé par le fait que les réseaux de capteurs sont déployés dans des environnements hostiles inaccessibles par les êtres humains et que leurs batteries à faible puissance ne peuvent pas être rechargeables, ni remplaçables et épuisables à un certain moment.

Ce travail nous a donné l'opportunité de se familiariser avec les réseaux de capteurs sans fil et en particulier avec les techniques de mise en veille dans ce type de réseau.

En effet, au bout de ces quelques mois d'étude, notre expérience dans le domaine des réseaux de capteurs sans fil s'est enrichie et notre savoir s'est approfondi.

Nous avons appris à orienter le protocole LEACH vers le côté énergétique et proposé un autre protocole PW-LEACH qui permet de pallier aux problèmes de la grande consommation énergétique avec les techniques de mise en veille dans ce type de réseau.

En guise de perspective, nous envisageons d'étudier l'effet de l'existence de plusieurs chefs dits aussi Cluster_Head via une nouvelle approche de Clustering.

Liste des acronymes

A

ADC: Analog to Digital Converters.

ASLEEP: Adaptive Staggered sLEEP

C

CH: Cluster Head

CP: Communication Period

CSIP: Collaborative Signal and Information Processing

CSMA: Carrier Sense Multiple Access

CSMA/CA: Carrier Sense Multiple Access Collision Avoidance

CTS: Clear To Send

D

DVS: Dynamic Voltage Scaling

E

Eamp : Energie d'amplification

Eelec : Energie électronique

ER : Energie de Réception

ET: Energie de Transmission

F

FRTS: Futur Request To Send

G

GPRS: General Packet Radio Service

L

LEACH : Low-Energy Adaptive Clustering Hierarchy

M

MAC: Media Access Control

MANET : Mobile Ad Hoc Network

MHz: Mega Hertz

N

NesC: Network Embedded System C

NP-CSMA-PS: Non Persistant- Carrier Sense Multiple Access-PerSistant

O

OSI: Open Systems Interconnection

P

PDA: Personal Digital Assistant

PW-LEACH : PoWer-Low-Energy Adaptive Clustering Hierarchy

R

RAM: Random Access Memory

RCSF : Réseaux de Capteurs Sans Fil

ROM: Read Access Memory

RPM: Redhat Package Manager.

RTS: Request To Send

S

SI: Silence Interval

S-MAC: Sensor-MAC

T

TDMA: Time Division Multiple Access

TI: Talk Interval

Tinyos: Tiny microthreading Operating System

T-MAC: Timeout MAC

W

WSN: Wireless Sensor Network

Liste des références

[ASSC02b] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Eredal Cayirci. Wireless sensor networks : a survey. Computer Networks, 38(4) :393422, 2002. 5, 6, 7, 20

[ASC02] I. Akyildiz, W. Su, E. Cayirci, Y. Sankarasubramaniam. "A survey on sensor networks", *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, Georgia Institute of Technology, Atlanta, USA. Août 2002.

[ASSC02a] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Eredal Cayirci. A Survey

on Sensor Networks. IEEE Communications Magazine, 40(8) :102114, August 2002. 5,7

[CES04] David Culler, Deborah Estrin, and Mani Srivastava. Guest editors' introduction :

Overview of sensor networks. Computer, 37(8) :4149, August 2004. 5, 6

[HCB00] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, " Energy-Efficient Communication Protocol for Wireless Micro sensor Networks", *In proc of the Hawaii International Conférence on Systems Science*, vol. 8, pp. 8020, January 2000.

[HW05] K. Holger and Andreas Willig. Protocols and Architectures for Wireless Sensor Networks.Wiley, 2005. 5, 6, 7, 14, 20, 21

[MI05] M. Ilyas and I. Mahgoub. "Handbook of sensor networks Compact wireless and wired Sensing Systems", ISBN 08493196864. CRC PRESS LLS, USA, 2005.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. l. Cayirci. "A survey on sensor

networks". IEEE Communications Magazine, Vol. 40, No. 8, pp. 102-116, August 2002.

[5] Equipe de Get 2005 Capt'Ad-hoc. "Sensor networks: State of the art". Technical Report,

Telecom Paris, ENST Br, INT, INRIA, Mars 2006.

[6] Sophia Kaplantzis, « security models for Wireless Sensor Networks », rapport de recherche, Mars 2006.

[7] Yasser Romdhane, "évalusation des performances des protocols S-MAC et directed diffusion dans les réseaux de capteurs", rapport de projet de fin d'étude, école supérieure des communication de Tunis, 2007

[5] Bouabdellah Kechar, « problématique de la consommation de l'énergie dans les réseaux de capteur sans fil », LIUPA, université d'Oran, Octobre 2007.

[5'] Bouabdellah Kechar, « problématique de la consommation de l'énergie dans les réseaux de capteurs sans fil », LIUA, université d'Oran, Octobre 2007.

[7'] Ajay Kalambur, « Secure Routing in Wireless Sensor Networks : A study on Directed Diffusion », San Jose State University, Dept, of EE

Annexe

Les étapes d'installation de TinyOS-1.x :

Au début de notre implémentation, nous avons installé en premier lieu le TinyOS-2.1.1 mais nous nous sommes avérés que cette version n'était pas vraiment adéquate par manque d'interface TinyViz qui permet de simuler le comportement du réseau dans la réalité.

Par la suite, nous nous sommes tournés vers la version TinyOS-1.x dont le guide d'installation se présente comme suit :

L'étape1 :

1. L'installation d'Ubuntu version 10.10.

L'étape 2 :

2. Le téléchargement de la JDK1.5 qui elle-même va être placée dans le dossier /opt en enlevant les droits de permission avec la commande:

sudo chmod 777 /opt

puis l'installer avec la commande :

cd /opt sudo sh jdk-1_5_5_22-linux-i586.bin

L'étape 3 :

3. L'ouverture du terminal de commande : "Applications > accessoires >terminal", puis l'ouverture du fichier « sources.list » avec la commande :

sudo gedit /etc/apt/ sources.list

L'étape 4 :

4. une fois le fichier ouvert copier et coller en fin de fichier les lignes juste en dessous puis sauvegardez.

#tinyOS

deb http://tinyos.stanford.edu/tinyos/dists/ubuntu edgy main

deb http://tinyos.stanford.edu/tinyos/dists/ubuntu feisty main

deb http://tinyos.stanford.edu/tinyos/dists/ubuntu edgy main

deb http://us.archive.ubuntu.com/ubuntu/ feisty main restricted

deb-src http://us.archive.ubuntu.com/ubuntu/ feisty main restricted

deb http://us.archive.ubuntu.com/ubuntu/ feisty-updates main restricted

deb-src http://us.archive.ubuntu.com/ubuntu/ feisty-updates main restricted

deb http://us.archive.ubuntu.com/ubuntu/ feisty universe

deb-src http://us.archive.ubuntu.com/ubuntu/ feisty universe

deb http://us.archive.ubuntu.com/ubuntu/ feisty multiverse

deb-src http://us.archive.ubuntu.com/ubuntu/ feisty multiverse

deb http://security.ubuntu.com/ubuntu feisty-security main restricted

deb-src http://security.ubuntu.com/ubuntu feisty-security main restricted

deb http://security.ubuntu.com/ubuntu feisty-security universe

deb-src http://security.ubuntu.com/ubuntu feisty-security universe

deb http://security.ubuntu.com/ubuntu feisty-security multiverse

deb-src http://security.ubuntu.com/ubuntu feisty-security multiverse

deb http://tinyos.stanford.edu/tinyos/dists/ubuntu feisty main

L'étape 5:

5. Nous aurons besoin de certains packages dont l'installation s'effectue comme suit:

   - sudo apt-get install cvs subversion autoconf automake1.9 python-dev
   - sudo apt-get install g++ g++-3.4 gperf swig sun-java5-jdk graphviz alien fakeroot
   - sudo apt-get install tinyos-msp430 tinyos-avr

En effet, **apt-get** est un outil à utiliser en ligne de commande pour l'installation, la désinstallation de paquets provenant de dépôt APT.

Pour une installation simple :

sudo apt-get install <paquet(s)>

Pour l'installation d'une version présente dans le dépôt c'est avec la commande :

sudo apt-get install <paquet>=<version> -V

Pour la désintallation de paquet :

sudo apt-get remove <paquets(s)>

L'étape 6 :

6. Pour les variables d'environnement il faut ouvrir le fichier baschrc dans le home (il est caché donc le faire apparaître avec ctrl h) puis ajouter en fin de ligne ce qui suit:

Copyright (c) 2006 Chad Metcalf <chad@5secondfuse.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

#The software is provided "AS IS", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and no infringement, in no event shall the author or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise arising from out of or in connection with the software or the use or other dealings in the software

Java

export JDKROOT=/usr/lib/jvm/java-1.6.0-sun

export JAVAXROOT=$JDKROOT

if ["$LD_LIBRARY_PATH" == ""]; then

So Java can find libtoscomm.so and libgetenv.so

export LD_LIBRARY_PATH=$JDKROOT/jre/lib/i386

fi

Set your default version of TinyOS here.

if ["$TINYOS_VER" == "2"]; then

export TINYOS_VER=2

else

export TINYOS_VER=1

fi

Preserve any non Tinyos related class paths that have been set prior to this file being sourced. We check the for an empty OLD_CLASSPATH to ensure that OLD_CLASSPATH only gets set once.

if ["$OLD_CLASSPATH" == ""]; then

```
if [ "$CLASSPATH" == "" ]; then
export OLD_CLASSPATH="empty"
else
export OLD_CLASSPATH=$CLASSPATH
fi
fi
# Conditional environmental setup for 3 versions of TinyOS
if [ "$TINYOS_VER" == "1" ]; then
echo "Setting up for TinyOS 1.x"
export TOSROOT=/opt/tinyos-1.x
export TOSDIR=$TOSROOT/tos
export MAKERULES=$TOSROOT/tools/make/Makerules
unset CLASSPATH
# Restore the old non tinyos classpath if its not empty
if [ "$OLD_CLASSPATH" != "empty" ]; then
export CLASSPATH=$OLD_CLASSPATH
fi
export CLASSPATH=$CLASSPATH\:`$TOSROOT/tools/java/javapath`
# These aliases only apply to a 1.x environment
alias tinyviz="$TOSROOT/tools/java/net/tinyos/sim/tinyviz"
alias Deluge="java net.tinyos.tools.Deluge"
else
echo "Setting up for TinyOS 2.x"
```

```
export TOSROOT=/opt/tinyos-2.x

export TOSDIR=$TOSROOT/tos

unset CLASSPATH

# Restore the old non tinyos classpath if its not empty

if [ "$OLD_CLASSPATH" != "empty" ]; then

export CLASSPATH=$OLD_CLASSPATH

fi

export
CLASSPATH=$CLASSPATH\:$TOSROOT/support/sdk/java/tinyos.jar\:$TOSROOT/suppor
t/sdk/java/

export PYTHONPATH=:$TOSROOT/support/sdk/python/

export MAKERULES=$TOSROOT/support/make/Makerules

unset TOSMAKE_PATH

fi

# Finally set the path for the new MSPGCCROOT

export MSPGCCROOT=/opt/msp430

export PATH="$MSPGCCROOT/bin:$PATH"

# Helpful aliases

alias sf="java net.tinyos.sf.SerialForwarder"

alias listen="java net.tinyos.tools.Listen"

alias apps="cd $TOSROOT/apps"

alias tos="cd $TOSROOT/tos"
```

```
# Environment changing functions
# Change this shell's environment to support Tinyos 2.x development
function tos2 () {
export TINYOS_VER=2
. ~/.bash_tinyos
}
# Change this shell's environment to support Tinyos 1.x development
function tos1 () {
export TINYOS_VER=1
. ~/.bash_tinyos
}
# Set the MOTECOM variable to the first mote in motelist
function setMoteCom () {
MOTE=`motelist -c| cut -d, -f2`
if [ "$BOOMERANG" == "1" ]; then
TYPE="tmote"
else
TYPE="telosb"
fi
export MOTECOM="serial@$MOTE:$TYPE"
}
# Change ownership of the TinyOS directories to the TinyOS group.
```

This was originally written for the LiveCD xubuntos but it's handy to have around.

function tos-fix-permissions () {

echo "sudo may prompt you for your password."

sudo chown -R root:tinyos /opt/tinyos-*

sudo chmod -R g+w /opt/tinyos-*

sudo find /opt/tinyos-* -type d -exec chmod 2775 {} \;

}

Add this to your .bashrc

if [-f ~/.bash_tinyos]; then

. ~/.bash_tinyos

fi

L'étape 7 :

7. ensuite pour Installer TinyOS 1.x il faut taper la commande:

cvs -d:pserver:anonymous@tinyos.cvs.sourceforge.net:/cvsroot/tinyos login password

root

cvs -z3 -d:pserver:anonymous@tinyos.

cvs.sourceforge.net:/cvsroot/tinyos co tinyos-1.x

sudo mv tinyos-1.x /opt

pour la vérification taper su root

password root ou ainsi la phrase setting up for tinyos-1.x apparaît au début de la console.

Le mot clé CVS:

Le CVS représente un système de contrôle de version qui nous permet de garder d'anciennes versions de fichiers habituellement du code source.

Il n'opère pas seulement sur un unique fichier ou répertoire à la fois mais sur des ensembles hiérarchiques de répertoires contenant des fichiers dont les versions sont contrôlées.

Remarque:

Certaines classes peuvent manquer alors il faut la vérification de dossier en les comparant avec ceux du site officiel de TinyOS :

http://www.tinyos.net/tinyos-1.x/