REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI OUZOU

FACULTE DE GENIE ELECTRIQUE ET DE L'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

Mémoire de fin d'études

En vue de l'obtention du diplôme de Master en informatique

Thème :

Optimisation de l'énergie dans un réseau de capteur sans fil

Application dans le protocole Leach

Proposé et dirigé par :

M^r^ LAGHROUCHE MOURAD

Réalisé par :

M^lle^ SALMI KENZA

Promotion : 2010-2011

REMERCIEMENTS

Mes remerciements vont en premier lieu à Monsieur LAGHROUCHE Mourad d’avoir accepté de diriger ce travail et pour ses qualités d’encadrement.

Je tiens également à remercier les membres du jury pour avoir accepté de juger ce travail.

Enfin tous ceux qui m’ont encouragé, aidé et contribué à la réalisation de ce projet.

Dédicaces

Je dédie ce modeste travail à :

Mes parents pour leurs soutiens et encouragements
Mon frère et ma sœur
Mes tantes, oncles, cousines et cousins
Tous mes amis (es).

KENZA

Table des matières

Chapitre II **« Consommation et conservation d'énergie dans les RCSF »**

Chapitre III **« protocole de routage Leach »**

Introduction générale

Les avancées technologiques et techniques opérées dans le domaine des réseaux sans fil, de la micro fabrication et de l'intégration des microprocesseurs ont fait naître une nouvelle génération de réseaux de capteurs à grande échelle adaptes à une gamme d'applications très variée. Cette nouvelle technologie promet de révolutionner notre façon de vivre, de travailler et d'interagir avec l'environnement physique qui nous entoure. Des capteurs communicants sans fil et dotés de capacités de calcul facilitent une série d'applications irréalisables ou trop chères il y a quelques années.

Aujourd'hui, des capteurs minuscules et bon marché peuvent être littéralement éparpillés sur des routes, des structures, des murs ou des machines capable de détecter une variété de phénomènes physiques. De nombreux domaines d'application sont alors envisagés tels que la détection et la surveillance des désastres, le contrôle de l'environnement et la cartographie de la biodiversité, le bâtiment intelligent, l'agriculture de précision, la surveillance et la maintenance préventive des machines, la médecine et la santé, la logistique et les transports intelligents.

Les réseaux de capteurs sans fil sont souvent caractérisés par un déploiement dense et à grande échelle dans des environnements limités en terme de ressources. Les limites imposées sont la limitation des capacités de traitement, de stockage et surtout d'énergie car ils sont généralement alimentés par des piles. Recharger les batteries dans un réseau de capteurs est parfois impossible en raison de l'emplacement des nœuds, mais le plus souvent pour la simple raison que cette opération est pratiquement ou économiquement infaisable. Il est donc largement reconnu que la limitation énergétique est une question incontournable dans la conception des réseaux de capteurs sans fil en raison des contraintes strictes qu'elle impose sur l'exploitation du réseau. En fait, la consommation d'énergie des capteurs joue un rôle important dans la durée de vie du réseau qui est devenue le critère de performance prédominant dans ce domaine. Si nous voulons que le réseau fonctionne de manière satisfaisante aussi longtemps que possible, ces contraintes d'énergie nous obligent à faire des compromis entre différentes activités aussi bien au niveau du nœud qu'au niveau du réseau.

Plusieurs travaux de recherche sont apparus avec un objectif : optimiser la consommation énergétique des nœuds à travers l'utilisation de techniques de conservation innovantes afin d'améliorer les performances du réseau, notamment la maximisation de sa durée de vie. De façon générale, économiser l'énergie revient finalement à trouver le meilleur compromis entre les différentes activités consommatrices en énergie.

La communauté des chercheurs a proposé un grand nombre de protocoles à tous les niveaux, de la couche physique jusqu'à la couche application, ils se classent naturellement en deux catégories. La première classe de protocoles émane du domaine des réseaux ad hoc, mais l'application de ces protocoles aux réseaux de capteurs sans fil engendre, sans surprise, une complexité et des couts énergétiques notoires. La deuxième classe de protocoles est propre aux réseaux de capteurs, ils sont souvent dirigés et élagués pour une application précise. Par conséquent, ils ne peuvent pas être appliqués et déployés de manière appropriée à tout contexte applicatif et nombre d'entre eux ont été largement déployés en raison des inconvénients liés aux problématiques de durabilité du réseau. Afin de prolonger la durée de vie du réseau, des protocoles tenant compte de l'efficacité énergétique sont nécessaires. En revanche, la liberté laissée à l'implantation est forte et impose de concevoir complètement l'infrastructure, les mécanismes et les protocoles en fonction de l'application visée puisque c'est la durabilité de cette dernière qui est en jeu.

Notre projet consiste à optimiser l'énergie dans les réseaux de capteurs sans fil on utilisant la stratégie des arbres de routage optimal qui est appliquée au protocole de routage LEACH, conçu pour les topologies des RCSF hiérarchiques.

Pour mieux cerner les enjeux de notre étude, nous introduirons dans un premier chapitre les RCSF, leurs caractéristiques, leurs domaines d'applications ainsi que leurs architectures. Dans le deuxième chapitre, nous présenterons la consommation et la conservation d'énergie dans les RCSF.

Dans le troisième chapitre, nous présenterons le protocole de routage Leach et son fonctionnement. Le quatrième chapitre quant à lui nous permettra d'exposer les résultats d'implémentation et de tests de simulation de notre solution.

Enfin, nous clôturons par une conclusion et des perspectives.

CHAPITRE I :

Généralités sur les réseaux de capteurs sans fil

1. Introduction

Avec l'essor de l'informatique mobile et les progrès réalisés en microélectronique, de nouveaux environnements ont vu le jour : les réseaux de capteurs sans fil.

Formés d'un ensemble de dispositifs miniatures capables de capter, traiter, mémoriser et transmettre des données, ces réseaux suscitent de plus en plus d'intérêt.

Ce nouveau type de réseaux, basé sur un travail collaboratif, permet d'observer et de contrôler certains phénomènes physiques tels que la pression atmosphérique, la température et le degré de pollution de l'air.

2. Environnements sans fil

Les réseaux sans fil suscitent de plus en plus d'intérêt, notamment dans les situations où la mise en place d'un réseau câblé est confrontée à de nombreux obstacles. Les réseaux sans fil représentent donc une réelle alternative.

Dans ce qui suit, nous donnons une description globale des réseaux sans fil et de leur fonctionnement.

2.1.Définition et caractéristiques

Un réseau sans fil, comme son nom l'indique, est un réseau composé de terminaux pouvant être fixes ou mobiles et communiquant sans liaison filaire. Contrairement à un réseau classique, les nœuds d'un réseau sans fil utilisent les ondes radiofréquences ou infrarouges pour communiquer. Les réseaux sans fil offrent une grande flexibilité d'emploi ; en particulier, ils permettent la mise en réseau de sites dont le câblage serait trop onéreux à réaliser.

Cependant, ils présentent certains inconvénients parmi lesquels : une puissance d'émission faible, un niveau de sécurité inférieur à celui offert par un réseau classique ainsi qu'une intégrité de données compromise par les interférences dues à l'environnement.

2.2. Classification des réseaux sans fil

Les réseaux sans fil peuvent être classés selon la zone de couverture (le périmètre géographique offrant une connectivité) ou selon l'existence d'une infrastructure de communication.

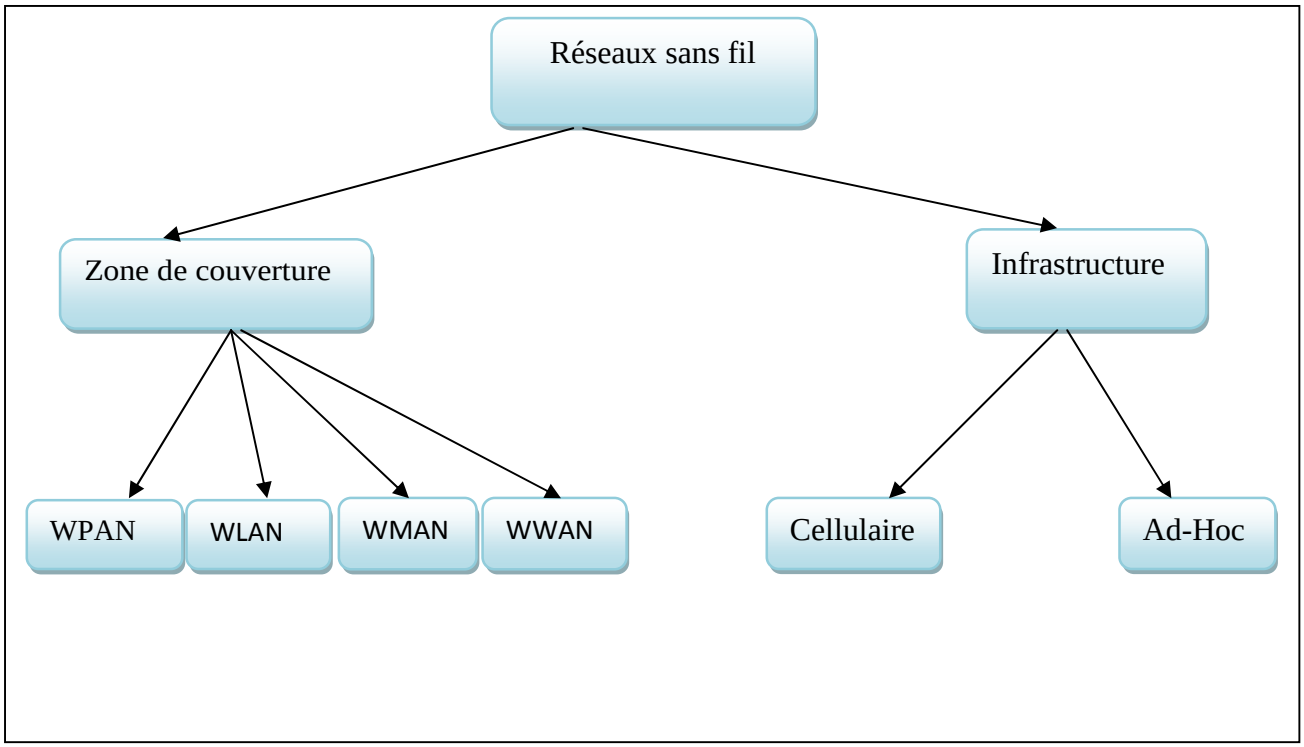


Fig. I.1. Classification des réseaux sans fil [HAM 07]

2.2.1. Classification selon la zone de couverture

La classification selon la zone de couverture permet de distinguer quatre catégories de réseaux: les réseaux personnels sans fil (WPAN : Wireless Personal Area Network), les réseaux locaux sans fil (WLAN : Wireless Local Area Network), les réseaux métropolitains sans fil (WMAN : Wireless Métropolitain Area Network) et les réseaux étendus sans fil (WWAN : Wireless Wide Area Network) **(Tableau 1)**

Type	Couverture Géographique	Performances	Normes
WPAN	Faible	Modérée	Bluetooth (2 Mbps). IrDa (4 Mbps)

WLAN	Quelques centaines de mètres	Hautes	Wifi (11-54 Mbps) HiperLAN(54 Mbps)
WMAN	Dans les limites d'une ville	Hautes	WiMAX (1Mbps-10 Mbps)
WWAN	Mondiale	Faibles	GSM (9,6Kbps),GPRS(115Kbps),UMT(2Mbps)

Tableau 1. Comparaison entre les différents types de réseaux sans fil [GEI 04]

2.2.2. Classification selon l'infrastructure de communication

Selon l'infrastructure de communication, les réseaux sans fil peuvent être classés en réseau avec infrastructure (réseaux cellulaires) et réseaux sans infrastructure (réseaux ad-hoc).

a. Réseau avec infrastructure

Dans les réseaux sans fil avec infrastructure, les communications entre les nœuds se font via un point d'accès qui peut être relié à un réseau fixe. Les nœuds du réseau ne peuvent pas communiquer entre eux, ils doivent obligatoirement passer par un point d'accès. Chaque point d'accès définit une région appelée cellule. La cellule correspond à la zone de couverture à partir de laquelle les unités mobiles peuvent émettre et recevoir des messages provenant d'autres nœuds [HAM 07]. Notons qu'il est possible de relier plusieurs points d'accès entre eux par des liens filaires ou un réseau sans fil.

b. Réseau sans infrastructure (Réseaux Ad-Hoc)

Dans le mode sans infrastructure, appelé aussi réseau **Ad-Hoc**, les nœuds du réseau communiquent entre eux en utilisant leur interface de communication sans fil. Dans ces réseaux, les unités se connectent les unes aux autres afin de constituer un réseau point à point, c'est-à-dire un réseau dans lequel chaque unité joue en même temps le rôle de client et celui de point d'accès. Ainsi, chaque nœud communique directement avec les nœuds se trouvant à sa portée. Quant aux nœuds se trouvant hors de sa portée, la communication se fait en passant par des nœuds intermédiaires qui se chargent de trouver un chemin vers le nœud destinataire. On parlera ainsi de routage multi-saut.

3. Réseaux de capteurs sans fil (RCSFs)

Dans cette section, nous présentons différents concepts liés aux RCSFs tels que l'architecture du réseau, la communication au sein du réseau ainsi que quelques domaines d'application.

3.1.Définitions

a. Un capteur

Un capteur est un composant électronique autonome à faible coût, servant à la surveillance et au contrôle d'un phénomène donné. Ce dispositif est capable de récolter des données, de les traiter et de les communiquer à ses voisins. L'alimentation électrique de chaque capteur est assurée par une batterie à faible capacité.

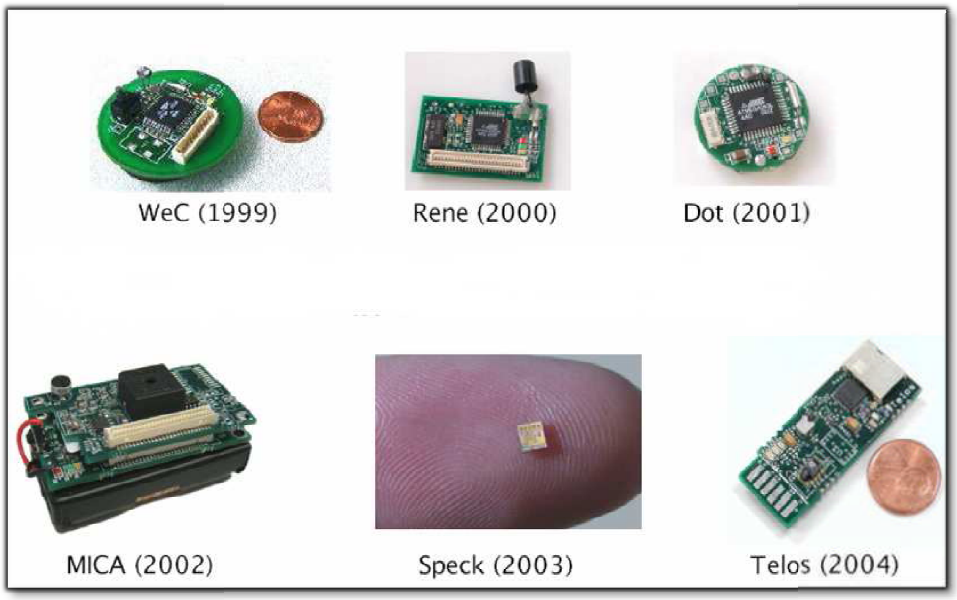


Fig. II.2 Ensembles des capteurs

b. Un réseau de capteurs

Un réseau de capteurs sans fil (RCSF ou WSN : Wireless Sensor Network) est un ensemble de nœuds capteurs dont le nombre varie de quelques dizaines d'éléments à plusieurs milliers. Les capteurs sont placés d'une manière plus ou moins aléatoire dans des environnements pouvant être dangereux. Toute intervention humaine après le déploiement des nœuds capteurs est la plupart du temps exclue. Le réseau doit donc s'autogérer.

3.2.Description physique d'un capteur

Un capteur est composé de quatre composantes principales : une unité de détection (Sensing Unit), une unité de calcul (Processing Unit), une unité de transmission (Transceiver unit), et une unité de contrôle d'énergie (Power Unit).

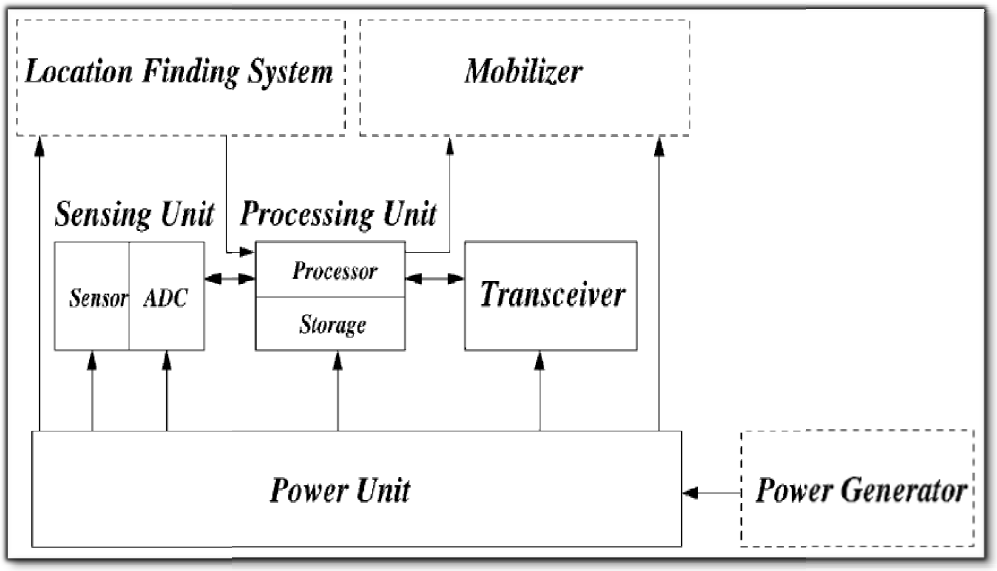


Fig. I.3. Description physique d'un capteur [AKY02]

Il se peut cependant que le capteur contienne des composantes additionnelles, et ce, selon son domaine d'application.

Dans ce qui suit, on détaillera les composants de base d'un capteur :

- **L'unité de détection (Sensing Unit)**

Comme le montre la (figure I.3) ci-dessus, l'unité de détection est généralement composée de deux modules : le capteur, et un convertisseur analogique numérique.

Le capteur est censé communiquer l'information recueillie depuis son environnement sous forme d'un signal analogique au convertisseur ADC (Analog to Digital Converter).

Le convertisseur analogique numérique transforme quant à lui les signaux provenant du capteur en signaux numériques en vue de les transmettre à l'unité de calcul.

- **L'unité de calcul (Processing Unit)**

L'unité de traitement comprend un processeur associé généralement à une petite unité de stockage et fonctionne à l'aide d'un système d'exploitation spécialement conçu pour les micros capteurs [DOU06]. Cette unité gère également les procédures et protocoles permettant la communication entre les nœuds.

- **L'unité de transmission (Transceiver)**

Cette unité gère toutes les émissions/réceptions au sein du capteur. Ainsi, l'unité de transmission travaille essentiellement à connecter le nœud au réseau. Typiquement, le module radio d'un nœud capteur a quatre modes opérationnels [HAC 03] :

- ✓ **Idle:** le module radio est allumé et il n'y a ni émission ni réception (écoute).
- ✓ **Transmit (Tx) :** le module radio est en train de transmettre un paquet.
- ✓ **Receive (Rx) :** le module radio est en train de recevoir un paquet.
- ✓ **Sleep :** le module radio est éteint.

- **L'unité de contrôle d'énergie**

La gestion de l'énergie est un point primordial dans les réseaux de capteurs. Tout capteur est muni d'une ressource énergétique à durée de vie limitée, et ce, à cause de la taille réduite de la batterie. Il est à noter que la gestion de l'énergie est un réel problème pour les réseaux de capteurs, et divers travaux tentent d'y remédier.

De ce fait, l'unité de contrôle d'énergie est sans doute le composant le plus important du capteur. Elle s'occupe de la « répartition » de l'énergie au sein du capteur entre les divers modules, elle permet également quelques fois de réduire la consommation d'énergie en agissant sur les modules inactifs. Le niveau de batterie est un critère primordial pour les décisions de participation au routage et de transmission, un nœud ayant un niveau de batterie critique ne participe pas au routage. On peut aussi conditionner la transmission lorsque le niveau de batterie devient critique, c'est-à-dire une fois le niveau critique atteint, le nœud ne transmet que si l'information captée est elle aussi critique.

3.3.Les états d'un nœud capteur

Un capteur est initialisé avant toute utilisation, c'est l'état Initiale. Puis il fonctionne selon trois modes : le mode Actif où il exécute une fonction ou transmet un message, le mode écoute (Idle) où il est seulement à l'écoute d'éventuels messages à recevoir et le mode veille (Sleep) où il est en veille. Enfin il se met dans l'état Stop lorsque la batterie est vide ou qu'il est mis hors tension (bouton OFF).

Lorsqu'il est actif, le capteur peut envoyer et recevoir des messages (et seulement recevoir s'il est dans l'état (Idle). La transmission des messages connaît trois états. Avant toute transmission, le module de communication du capteur (la radio) doit être dans l'état

Idle, c'est-à-dire qu'aucune transmission n'est en cours sur le réseau. Le réseau doit être vide de message. Après quoi le capteur peut soit envoyer soit recevoir un message.

Un capteur ne peut pas envoyer et recevoir des messages simultanément. Il fonctionne en half duplex, c'est-à-dire qu'il ne fait qu'une transmission à la fois.

La figure. I.4 présente les différents états d'un capteur.

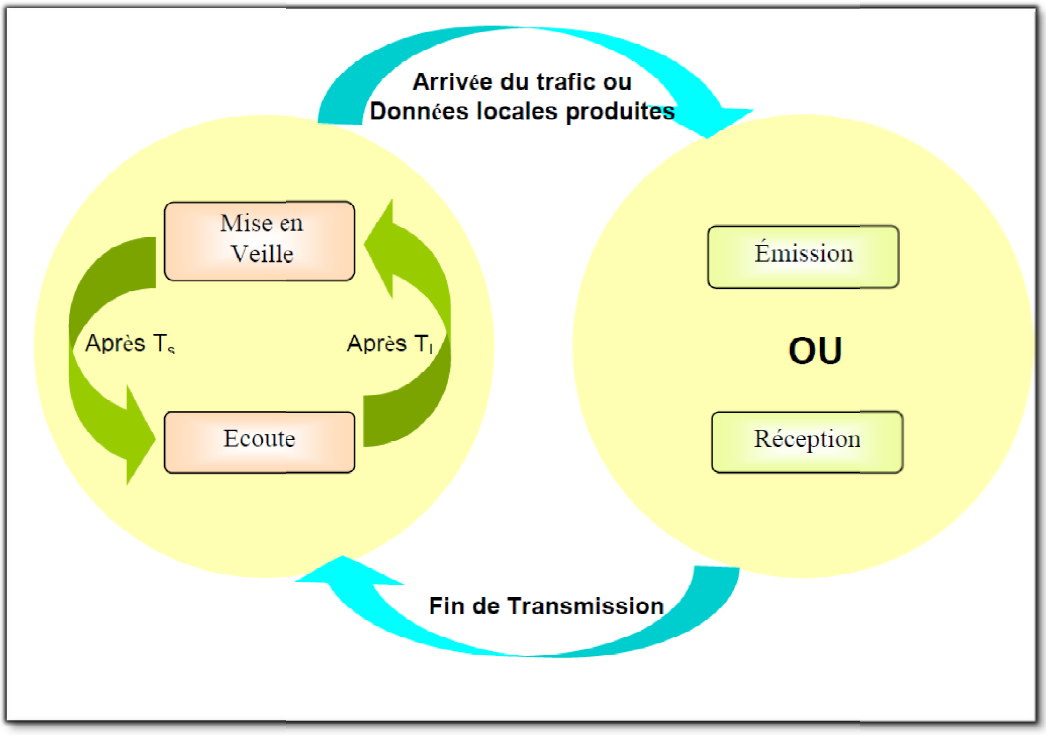


Fig. I.4.Diagramme de transition d'état du modèle du nœud capteur

3.4. Architecture d'un réseau de capteurs sans fil

Un RCSF est typiquement constitué d'un champ de captage, plusieurs nœuds capteurs, une station de base et un centre de traitement des données :

- **Le champ de captage (espace de collecte)**

  Il est considéré comme étant la zone d'intérêt pour le phénomène capté, donc les nœuds capteurs y sont placés.

- **Les nœuds capteurs** :

  Ce sont le cœur du réseau, leur rôle est de collecter les données et de les router vers la station de base. Leur énergie est souvent limitée puisqu'ils sont alimentés par des batteries.

- **Le sink (station de base, puits)**

  C'est un nœud particulier chargé d'accueillir, stocker et traiter les données en provenance des autres nœuds et de diffuser les différentes requêtes au réseau. Sa source d'énergie doit être illimitée puisqu'il faut qu'il reste toujours actif pour recevoir les données.

- **Centre de traitement des données (gestionnaire de tâches)**

  Il reçoit les données collectées par le puits, son rôle consiste à les regrouper et les traiter pour en extraire les informations utiles.

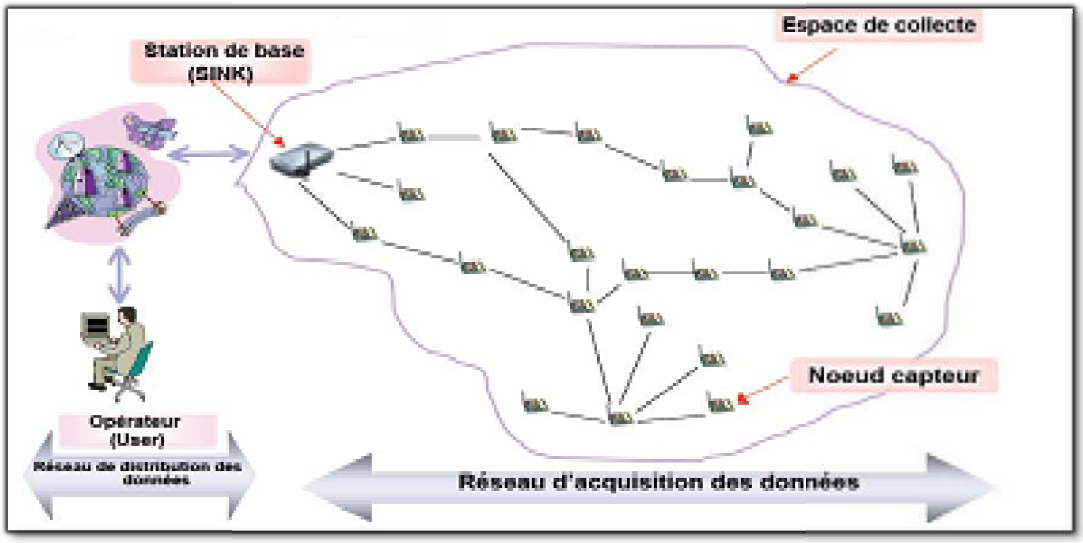


Fig. I.5. Architecture d'un réseau de capteur

Comme le montre la (Fig. 5), un RCSF est composé d'un grand nombre de nœuds capteurs éparpillés sur le champ de captage. Quand le puits diffuse une requête, les nœuds collaborent entre eux pour lui envoyer les informations captées à travers une architecture multi-saut.

Le puits transmet ensuite ces données par internet ou par satellite au gestionnaire de tâches pour les analyser et prendre des décisions. A un niveau plus élevé un WSN peut être vu comme étant une combinaison de deux entités de réseau :

- Le réseau d'acquisition des données : c'est l'union des nœuds capteurs du puits. Son rôle consiste à collecter les données à partir de l'environnement de les rassembler au puits.
- Le réseau de distribution des données : son rôle est de connecter le réseau d'acquisition des données à un utilisateur.

3.4.1. Classification des architectures

3.4.1.1.Architecture plate

Un réseau de capteurs sans fil plat est un réseau homogène, où tous les nœuds disposent des mêmes capacités et fonctionnalités concernant le captage, la communication et la complexité du matériel, seul le puits échappe à cette règle puisqu'il joue le rôle d'une passerelle chargée de la collecte des données issues des différents nœuds capteurs pour les transmettre à l'utilisateur.

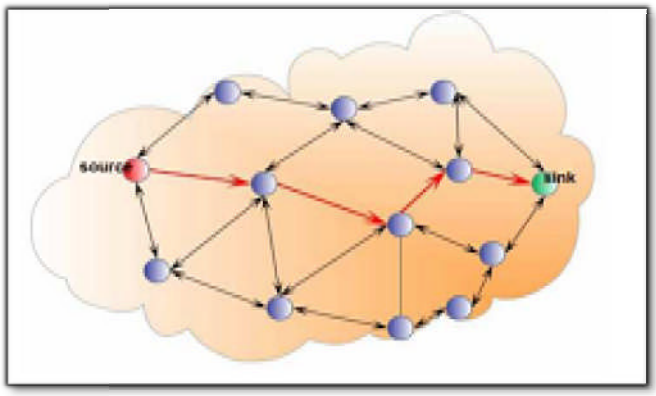


Fig. I.6. Architecture plate

3.4.1.2.Architecture hiérarchique

Un réseau de capteurs hiérarchique est un réseau hétérogène où les nœuds ont des capacités différentes, par exemple certains nœuds peuvent disposer d'une source d'énergie plus importante, une plus longue portée de communication et/ou une plus grande puissance de calcul. Ceci permet de décharger la majorité des nœuds simples à faible coût de plusieurs fonctions du réseau.

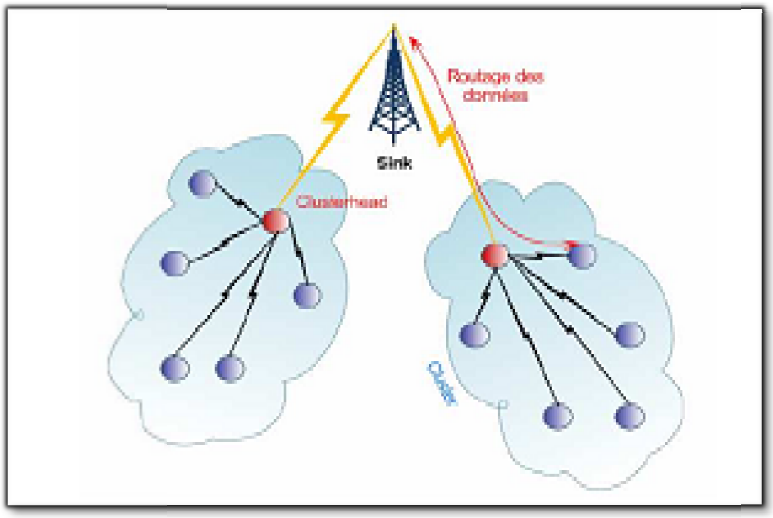


Fig. I.7. Architecture hiérarchique [YAS07]

3.5.Caractéristiques d'un réseau de capteurs sans fil :

- **Durée de vie limitée :**

  Les nœuds capteurs sont très limités par la contrainte d'énergie, ils fonctionnent habituellement sans surveillance dans des régions géographiques éloignées. Par conséquent, recharger ou remplacer leurs batteries devient impossible.

- **Ressources limitées :**

  Habituellement les nœuds capteurs ont une taille très petite, ce facteur de forme limite la quantité de ressources qui peuvent être mises dans ces nœuds. Par conséquent, la capacité de traitement et de mémoire devient très limitée.

- **Topologie dynamique :**

La topologie des réseaux de capteurs change d'une manière fréquente et rapide car :

- ✓ Les nœuds capteurs peuvent être déployés dans des environnements hostiles (par exemple un champ de bataille), la défaillance d'un nœud capteur est, donc très probable.
- ✓ Les nœuds capteurs et les nœuds puits peuvent être mobiles.

- **Scalabilité :**

Le nombre de nœuds capteurs peut être de l'ordre de centaines ou de milliers (selon l'application).

- **Déploiement aléatoire et dense :**

Dans un RCSF, les capteurs sont généralement déployés d'une façon dense et plus ou moins aléatoire. La forte densité est souvent liée à des raisons de fiabilité.

- **Auto-organisation :**

Pour remédier au problème de changement non prédictible de topologie, une auto organisation du réseau s'avère nécessaire. C'est-à-dire que les nœuds doivent savoir localiser leurs voisins et établir des routes pour que l'information puisse circuler à travers le réseau.

- **Sécurité limitée :** Les contraintes et limitations physiques font que le contrôle des données transférées doit être minimisé.
  De plus, les réseaux de capteurs sans fils sont plus sensibles aux attaques qui menacent les données transmises en raison de l'absence d'infrastructure.

3.6.Communication dans les réseaux de capteurs sans fil :

a. La pile protocolaire :

La pile de protocole employer par le puits et les autres nœuds de capteurs est donné par la figure.8. Cette pile de protocole combine la gestion de la puissance et de routage, intégrer les données avec les protocoles de gestion de réseau, communique la puissance efficacement a travers le média sans fil, et favorise les efforts coopératifs des nœuds de capteurs. La pile de protocole comprend la couche application, couche transport, couche réseau, couche liaison des données, la couche physique, le plan de gestion de puissance, le plan de gestion de mobilité, et le plan de gestion des tâches.

•Couche application :

Elle assure l'interface avec les applications. Il s'agit donc de la couche la plus proche des utilisateurs, gérée directement par les logiciels.

•**Couche transport :**

Elle vérifie le bon acheminement des données et la qualité de la transmission. Dans les RCSF, la fiabilité de transmission n'est pas majeure. Ainsi, les erreurs et les pertes sont tolérées.

•**Couche réseau :**

Elle s'occupe du routage de données fournies par la couche transport.

Elle établit les routes entre les nœuds capteurs et le nœud puits et sélectionne le meilleur chemin en termes d'énergie, délai de transmission, débit, etc.

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

•**Couche physique :** Elle permet de moduler les données et les acheminer dans le media physique tout en choisissant les bonnes fréquences.

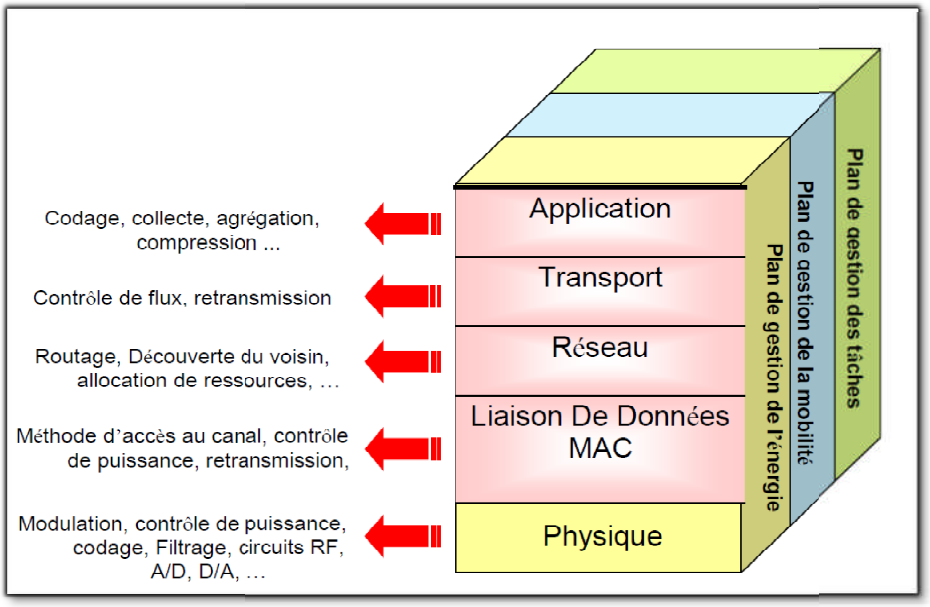


Fig. I.8. La pile protocolaire des RCSF

b. Types de communications dans les réseaux de capteurs

Un réseau de capteurs sans fil doit assurer trois types de schémas :

- **Many-to-one :** Il est généralement utilisé pour acheminer les rapports de données vers le puits.
- **One-to-many :** Ce schéma est utilisé par le puits pour communiquer avec les capteurs, afin de propager des requêtes ou contrôler le fonctionnement des nœuds.
- **Communications locales :** Ce sont les communications à un saut entre des capteurs voisins.

3.7. Domaines d'applications des réseaux de capteurs sans fil

a) Applications militaires

Les premières applications des réseaux de capteurs ont été introduites dans le domaine militaire, elles consistent en des applications de surveillance : interception des mouvements de l'ennemi, étude des caractéristiques du champ de bataille, détection des attaques nucléaires et chimiques, etc...

Fig. I.9. Un service militaire utilisant les RCSF

b) Applications environnementales

Cette technologie consiste en la mesure des températures, des pressions, d'humidités dans un endroit.

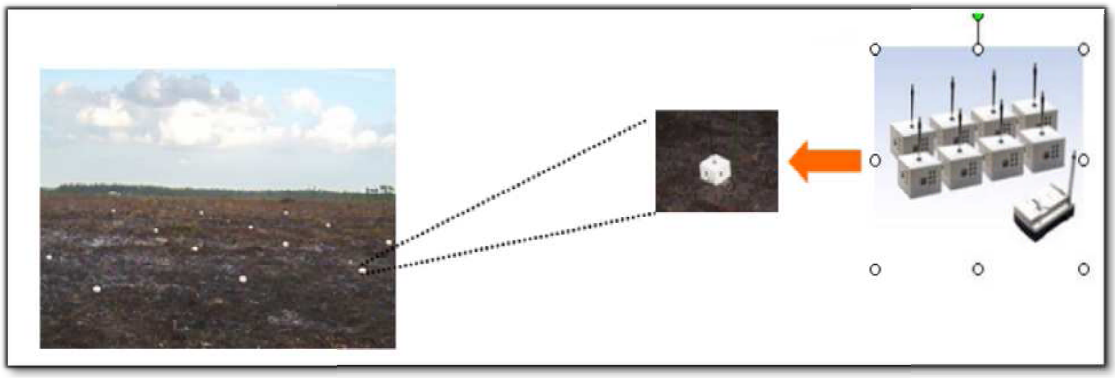


Fig. I.10. Domaine de contrôle de l'environnement en utilisant RCSF

✓ **Système de surveillance sous l'eau par un réseau de capteurs sans fil.**

Les nœuds capteurs situés sur la surface permettent la surveillance sous l'eau

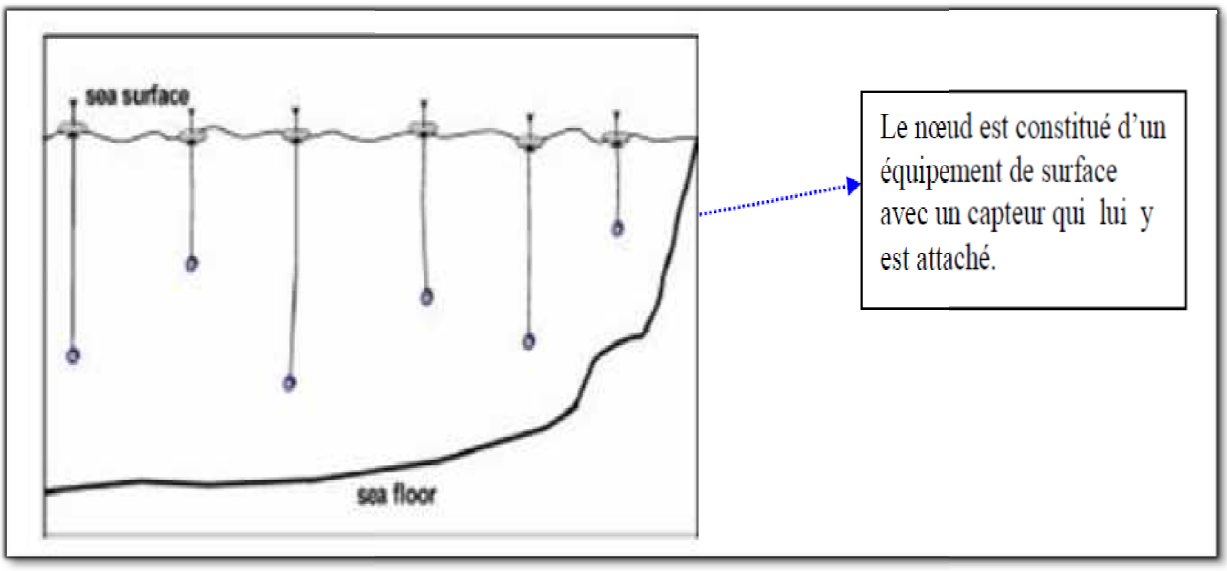


Fig. I.11. Surveillance sous l'eau par RCSF

✓ **Etude de l'activité volcanique et sismique**

Un réseau de capteurs étalé autour d'un volcan ou sur une grande surface terrestre permet un suivi continu des activités volcaniques ou sismiques.

Fig. I.12. Etude de l'activité volcanique

c) **Applications industrielles :** pour gérer la circulation, la direction d'un objet, la lumière, etc.

d) **Applications médicales** : cette technologie consiste en la surveillance de la température, la fréquence cardiaque, l'oxygénation du sang et le pouls du patient,…etc.

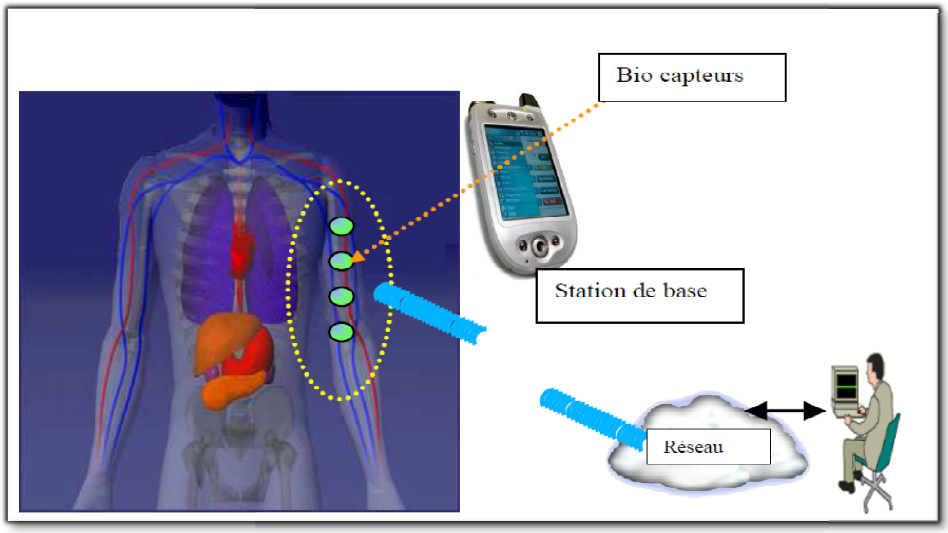


Fig. I.13. Application des RCSF dans le domaine de la médecine.

4. Conclusion

Bien qu'ils apportent de nombreux avantages, les réseaux de capteurs posent un certain nombre de défis scientifiques. En effet, la miniaturisation des batteries a entrainé un problème de limitation d'énergie. La consommation d'énergie représente ainsi un facteur majeur lors de la conception des réseaux de capteurs. De ce fait, les travaux doivent se porter sur les techniques minimisant la consommation énergétique.

CHAPITRE II :

Consommation et conservation d'énergie dans les RCSF

1. Introduction

La durée de vie est sans doute la métrique la plus importante dans l'évaluation des performances d'un réseau de capteurs. En effet, dans un environnement contraint, toute ressource limitée doit être prise en compte. Toutefois, la durée de vie du réseau, comme mesure de la consommation d'énergie, occupe une place exceptionnelle puisqu'elle constitue la borne supérieure de l'utilité de ce réseau.

La durée de vie est également considérée comme un paramètre fondamental dans un contexte de disponibilité et de sécurité dans les réseaux de capteurs sans fil**.**

Maximiser la durée de vie du réseau revient à réduire la consommation énergétique des nœuds. Malgré les progrès qui ont été faits, la durée de vie de ces dispositifs à piles continue d'être un défi majeur et un facteur clé, exigeant davantage de recherches sur l'efficacité énergétique des plates-formes et des protocoles de communication.

2. Consommation d'énergie dans les réseaux de capteurs sans fil

Les capteurs sont généralement munis d'une ressource énergétique à durée de vie limitée et sont, la plupart du temps, déployés dans des endroits inaccessibles, donc toute intervention pour changer ou recharger les batteries est pratiquement impossible. Ceci fait que la durée de vie du réseau dépende de la consommation d'énergie.

En effet, les différentes tâches effectuées par les nœuds capteurs contribuent à l'épuisement des ressources énergétiques des capteurs. Avec l'épuisement de leurs ressources énergétiques les nœuds ne pourront plus participer au routage des données afin de les acheminer vers la station de base. Ainsi, la topologie du réseau risque de changer.

C'est pourquoi, l'optimisation en consommation d'énergie est primordiale. Cette dernière est fortement liée aux trois principales tâches accomplies par un nœud et qui sont :

2.1.Energie de Capture

Cette phase inclut l'échantillonnage des signaux physiques, le traitement du signal et la conversion analogique/numérique [ARA 08]. L'énergie consommée pendant la phase de capture varie selon la nature de l'application. En effet, une capture à intervalles réguliers consomme moins d'énergie qu'une surveillance continue.

2.2.Energie de traitement

L'énergie de traitement se divise en deux parties : l'énergie de commutation et l'énergie de fuite. L'énergie de commutation est déterminée par la tension d'alimentation et la capacité totale commutée au niveau logiciel (en exécutant un logiciel). Par contre l'énergie de fuite correspond à l'énergie consommée lorsque l'unité de calcul n'effectue aucun traitement.

2.3.Energie de communication

Les communications (Connectivité) consomment beaucoup plus d'énergie que les autres tâches. Elles couvrent les communications en émission et en réception.

2.3.1. Modèle de consommation d'énergie

Heinzelman et al. [HCB00] proposent un modèle radio de consommation d'énergie (voir figure 1). Ainsi, les énergies nécessaires pour émettre ETx(s, d) et recevoir ERx(s) des messages sont données par :

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

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

$$ETx(s, d) = (Eelec * s) + (Eamp * s * d2)$$

• Pour recevoir un message de s bit, 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

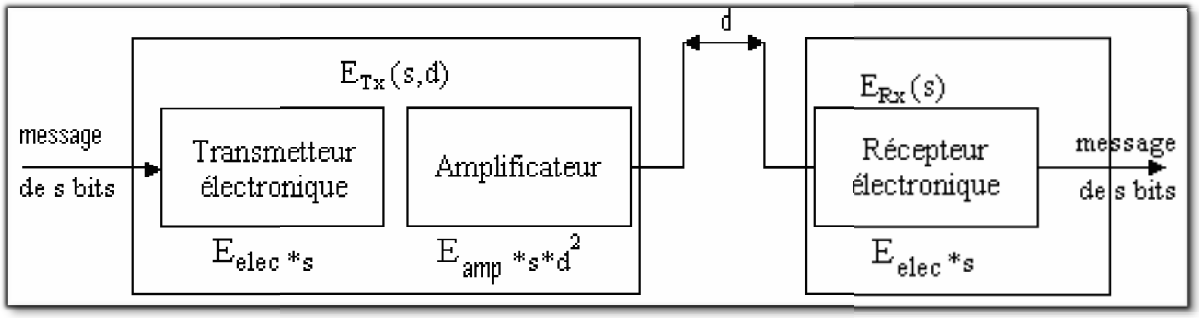


Fig. II.1. Modèle de consommation d'énergie pour la communication

La consommation de l'énergie dans un réseau de capteurs est définie par l'ordre décroissant suivant [POM 04] :

• La communication (réception, émission).

• Traitement des données.

• Capture des données.

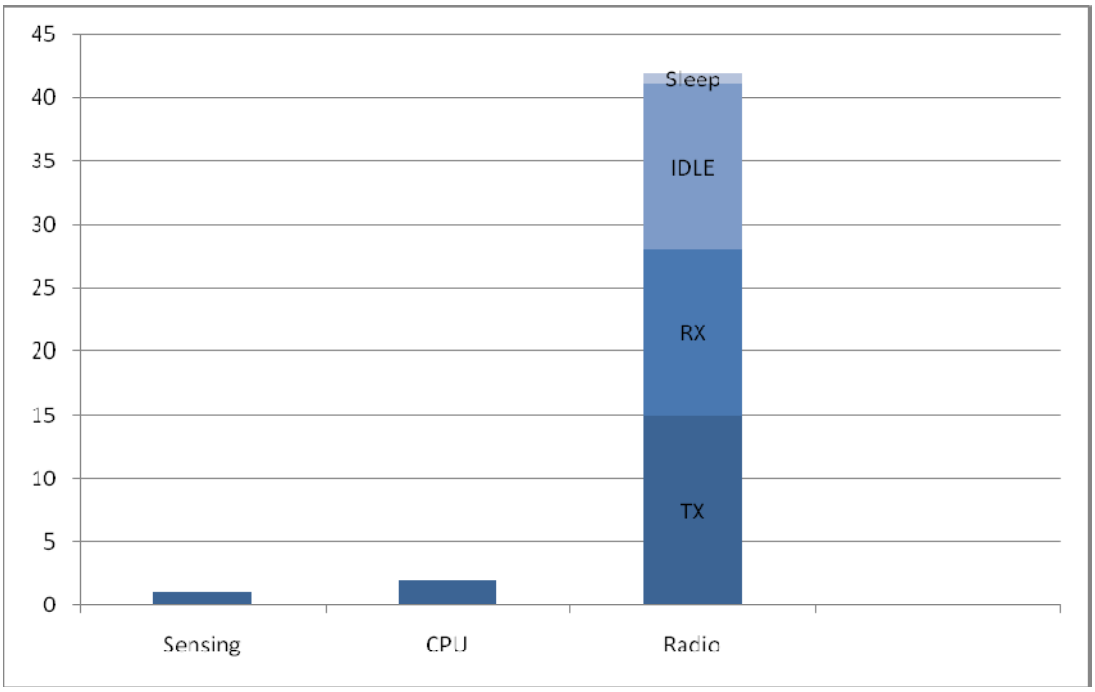


Fig. II.2.Energie consommée par les sous-systèmes d'un capteur [POM 04]

3. Facteurs intervenants dans la consommation d'énergie

La consommation d'énergie dépend de plusieurs facteurs qui sont expliqués ci-dessous.

3.1. Etat du module radio

Le module radio est le composant du nœud capteur qui consomme le plus d'énergie, puisque c'est lui qui assure la communication entre les nœuds. On distingue quatre états des composants radio : actif, réception, transmission et sommeil. [YN04]

• Etat actif

La radio est allumée, mais elle n'est pas employée. En d'autres termes, le nœud capteur n'est ni en train de recevoir ni de transmettre.

Cet état provoque une perte de l'énergie suite à l'écoute inutile du canal de transmission. Pour éviter cette perte d'énergie, un capteur doit s'activer qu'en cas de nécessitée, et le reste du temps il doit se mettre dans l'état sommeil.

• **Etat sommeil**

La radio est mise hors tension.

• **Etat transmission**

La radio transmet un paquet.

• **Etat réception**

La radio reçoit un paquet.

3.2.Accès au medium de transmission

La sous couche MAC assure l'accès au support de transmission, la fiabilité transmission, le contrôle de flux, la détection d'erreur et la retransmission des paquets.

Puisque les nœuds partagent le même médium de transmission, la sous-couche MAC joue rôle important pour la coordination entre les nœuds et la minimisation de la consommation d'énergie. En effet, minimiser les collisions entre les nœuds permet de réduire la perte d'énergie. Dans ce qui suit, nous analyserons les principales causes de consommation d'énergie au niveau de la couche MAC. [MI05]

3.2.1. La retransmission

Les nœuds capteurs possèdent en général une seule antenne radio et partagent-le même canal de transmission. Par ailleurs, la transmission simultanée des données provenant plusieurs capteurs peut produire des collisions et ainsi une perte de l'information transmise.

La retransmission des paquets perdus peut engendrer une perte significative de l'énergie.

3.2.2. 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.

3.2.3. La surémission

Le phénomène de surémission (overemitting) se produit quand un nœud capteur envoie les données à un destinataire qui n'est pas prêt à les recevoir. En effet, les messages envoyés sont considérés inutiles et consomment une énergie additionnelle.

3.2.4. La taille des paquets

La taille des messages échangés dans le réseau a un effet sur la consommation d'énergie des nœuds émetteurs et récepteurs. Ainsi, la taille des paquets ne doit être ni trop élevée ni trop faible. En effet, si elle est petite, le nombre de paquets de contrôle (acquittement) généré augmente l'overhead. Dans le cas contraire, une grande puissance de transmission est nécessaire pour des paquets de grande taille.

3.3.Modèle de propagation radio

Le modèle de propagation représente une estimation de la puissance moyenne reçue du signal radio à une distance donnée d'un émetteur. La propagation du signal radio est généralement soumise à différents phénomènes : la réflexion, la diffraction et la dispersion par divers objets. Généralement, la puissance du signal reçue est de l'ordre de 1/dn , où d est la distance entre l'émetteur et le récepteur, n un exposant de perte d'un chemin (Exemple : n=2 dans le vide, de 4 à 6 dans un immeuble) [LNN05].

3.4. Routage des données

Le routage dans les réseaux de capteurs est un routage multi-sauts. L'acheminement des paquets d'une source donnée à une destination se fait à travers plusieurs nœuds intermédiaires. Ainsi, un nœud consomme de l'énergie soit pour transmettre ces données ou pour relayer les données des autres nœuds. Dans ce contexte, une mauvaise politique de routage peut avoir des conséquences graves sur la durée de vie du réseau.

4. Techniques de minimisation de la consommation d'énergie

Après la description des principales causes de consommation d'énergie dans les RCSF, nous présentons dans ce qui suit les différentes techniques utilisées pour minimiser cette consommation.

Ces techniques sont appliquées soit au niveau de la couche liaison (sous couche MAC) ou au niveau de la couche réseau. Le schéma suivant (figure II.3) donne un aperçu global de ces mécanismes.

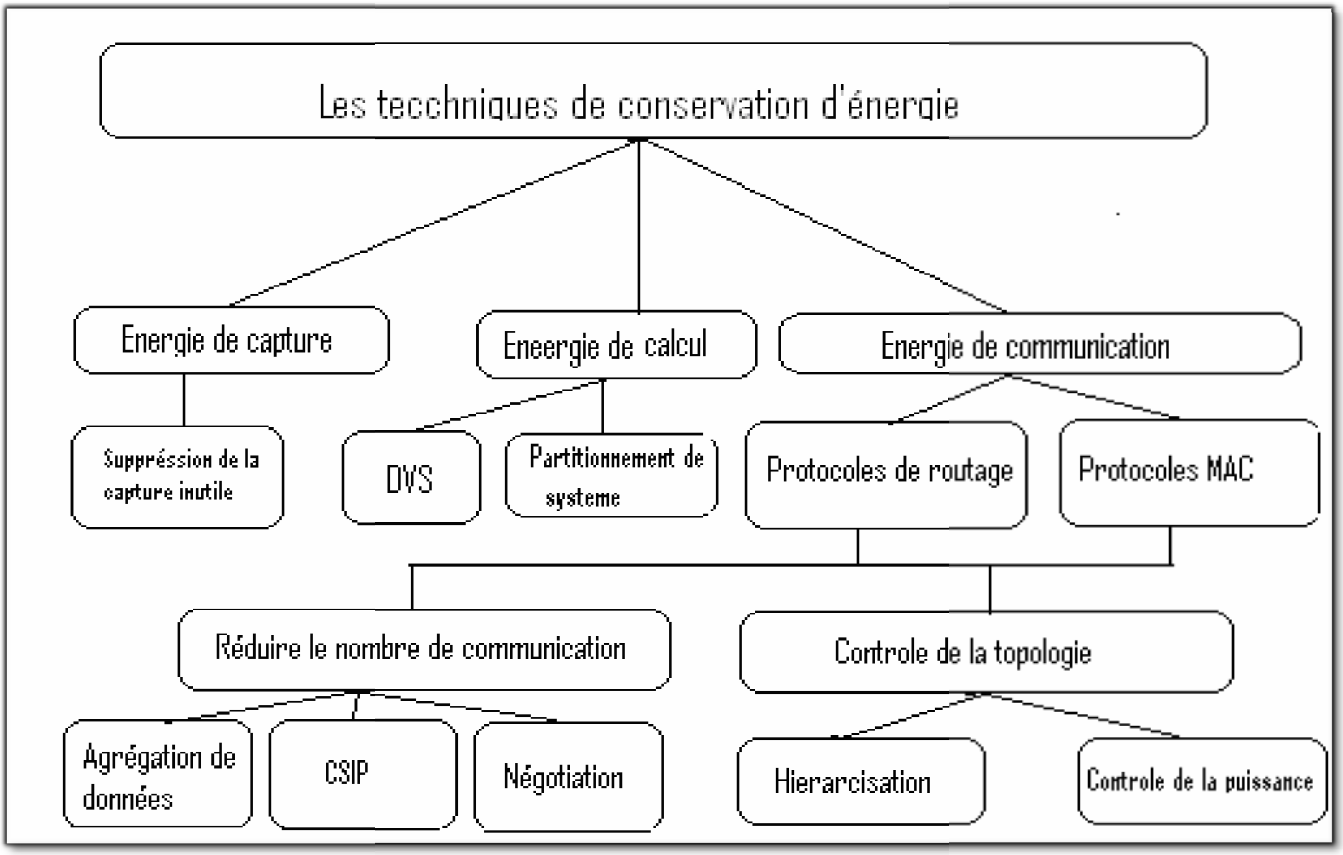


Figure II.3. Les techniques de conservation d'énergie

L'énergie de capteur peut être économisée soit au (a) niveau de capture, (b) niveau de traitement ou au (c) niveau de communication.

a- La seule solution apportée pour la minimisation de la consommation d'énergie au niveau de la capture consiste à réduire les durées de captures.

b- L'énergie de calcul peut être optimisée en utilisant deux techniques :

- ✓ L'approche DVS (Dynamique Voltage Scaling) [ZM05], consiste à ajuster de manière adaptative la tension d'alimentation et la fréquence de microprocesseur pour économiser la puissance de calcul sans dégradation des performances.

- ✓ L'approche de partitionnement de système, consiste à transférer un calcul prohibitif en temps de calcul vers une station de base qui n'a pas de contraintes énergétiques et qui possède une grande capacité de calcul. [SAN05]

c- La minimisation de la consommation d'énergie pendant la communication est étroitement liée aux protocoles développés pour la couche réseau et la sous couche MAC. Ces protocoles se basent sur plusieurs techniques : agrégation de données, négociation et CSIP (Collaborative Signal and Information Processing). Cette dernière technique est une discipline qui combine entre plusieurs domaines [KSZ02] : la communication et le calcul à basse puissance, traitement de signal, algorithmes distribués et tolérance aux fautes, systèmes adaptatifs et théorie de fusion des capteurs et des décisions. Ces techniques ont le but de réduire le nombre d'émission/ réception des messages.

Par contre, le contrôle de la topologie permet l'ajustement de la puissance de transmission et le regroupement des nœuds capteurs (hiérarchisation). [SAN05]

Le contrôle de la puissance de transmission n'a pas seulement un effet sur la durée de vie de la batterie d'un nœud capteur, mais aussi sur la capacité de charge du trafic qui est caractérisée par le nombre de paquets transmis avec succès vers une destination. En outre, il influe sur la connectivité et la gestion de la densité (le nombre de nœuds voisins).

Le module de contrôle de la puissance est souvent intégré dans des protocoles de la couche réseau ou MAC. [NKSK06]

La hiérarchisation consiste à organiser le réseau en structure à plusieurs niveaux. C'est le cas, par exemple, des algorithmes de groupement (clustering), qui organisent le réseau en groupes (clusters) avec des leaders de groupe (clusterheads) et des nœuds membres. [LNN05]

5. Les principaux algorithmes de conservation d'énergie

De nombreux algorithmes de routage ont été spécifiquement conçus pour les réseaux de capteurs où la consommation d'énergie est un facteur essentiel. Ce facteur a posé de nombreux défis à la conception et à la gestion des réseaux de capteurs. Ces défis nécessitent une gestion efficace de l'énergie pour toutes les couches de la pile de protocole réseau. Les problèmes liés aux couches liaison et physique sont généralement communs pour les différents types d'applications de capteurs. Pour la couche réseau, l'objectif principal est de trouver des moyens pour une mise en œuvre efficace de l'énergie et pour une diffusion fiable

des données de la source vers la destination de sorte que la durée de vie du réseau soit maximisée.

Quelques algorithmes ont été développés pour prendre soin du contrôle de la capacité.

Certains se concentrent sur l'utilisation effective du temps d'activation des nœuds des capteurs Plusieurs paramètres tels que la distance de transmission, le nombre de sauts et le retard ont été pris en compte pour les économies d'énergie dans les réseaux de capteurs.

La recherche dans le domaine des protocoles d'énergie efficaces dans les réseaux de capteurs sans fil est relativement nouvelle.

L'objectif principal de tous ces protocoles et algorithmes est de trouver les routes qui sont économes en énergie et donc de maximiser la durée de vie du réseau. Pour atteindre cet objectif, de nombreux protocoles ont été développés. Ces protocoles utilisent des stratégies différentes pour obtenir leurs routes.

Certains de ces protocoles utilisent le clustering, des fonctions, des équations, des arbres de routage optimal, des recherches multi-paths ou un mécanisme efficace pour détourner les trous.

Les algorithmes d'énergie efficace les plus connues sont:

5.1.L'algorithme de routage « EARLEAHSN »

L'algorithme Energy Aware Routing for Low Energy Ad Hoc Sensor Networks (EARLEAHSN) utilise un ensemble de sous-chemins optimaux afin d'augmenter la durée de vie du réseau. Ces chemins sont choisis au moyen d'une fonction de probabilité qui dépend de la consommation énergétique de chaque route. La survie du réseau est la mesure principale de cette approche qui propose d'éviter l'utilisation permanente de la route la plus économe en énergie car cela épuise l'énergie des nœuds sur cette route. Au lieu de cela, l'un des trajets multiples est utilisé avec une certaine probabilité de sorte que la vie entière du réseau se prolonge. Le protocole suppose que chaque nœud est adressable par le biais d'une classe d'adresse qui comprend l'emplacement et les types de nœuds.

5.2.L'algorithme de routage «EARCBSN »

L'algorithme Energy-Aware Routing in Cluster-Based Sensor Networks (EARCBSN) propose un algorithme de routage hiérarchique basé sur une architecture à trois niveaux.

Les capteurs sont regroupés en clusters avant l'exploitation du réseau. L'algorithme emploie les clusters-head comme des passerelles et ces clusters-head possèdent de l'énergie plus que les autres capteurs et connaissent l'emplacement des tous les capteurs.

Le routage nécessite une maintenance d'un cluster-head qui inclut tous les paramètres qui influent sur la décision de routage. Dans cet algorithme, ces paramètres sont l'état du capteur, sa localisation, l'énergie restante et le trafic des messages. Il y a une certaine imprécision dans le modèle d'énergie des passerelles due à la surcharge, la perte des paquets et au retard de propagation des messages. Le nœud passerelle agit comme un gestionnaire de cluster de réseau centralisé qui achemine les routes pour les données des capteurs, qui contrôle la latence dans tout le cluster et qui arbitre les accès entre les capteurs.

Le nœud passerelle trace l'utilisation d'énergie de chaque nœud des capteurs et il contrôle aussi les changements dans l'environnement. De plus, il permet de configurer les capteurs et le réseau efficacement afin de prolonger la vie du réseau.

5.3. L'algorithme de routage « GBR »

L'algorithme Gradient-Based Routing (GBR) ou le routage par gradient est une version légèrement modifiée de la diffusion dirigée . L'idée de ce protocole est de maintenir le nombre de sauts lorsque le paquet est diffusé à travers le réseau. Ainsi chaque nœud peut découvrir le nombre minimal de sauts jusqu'à la destination. Ce nombre est appelé hauteur du nœud. La différence entre la hauteur d'un nœud et celui de son voisin est considérée comme le gradient sur ce lien. Un paquet est transmis sur un lien avec le gradient le plus grand. Le routage par gradient vise à utiliser certaines techniques auxiliaires telles que l'agrégation des données afin d'équilibrer le trafic de manière uniforme sur le réseau. Trois techniques différentes pour gérer les données ont été présentées:

· **Stochastic Scheme** : quand il y a deux sauts ou plus avec le même gradient, le nœud choisit l'un d'eux au hasard.

· **Energy-based scheme :** lorsque l'énergie d'un nœud tombe en dessous d'un certain seuil, il augmente sa hauteur afin que les autres capteurs soient découragés d'envoyer des données à ce nœud.

· **Stream-based scheme** : l'idée est de détourner les nouveaux flux à partir de nœuds qui font actuellement parti de la trajectoire des autres filières Les données s'efforcent alors de parvenir à une répartition égale de la circulation à travers l'ensemble du réseau, ce qui contribue à

équilibrer la charge sur les nœuds des capteurs et augmente la durée de vie du réseau. Les techniques employées pour équilibrer la charge de trafic et la fusion de données sont également applicables aux autres protocoles de routage pour des performances améliorées.

6. Conclusion

La durée de vie d'un réseau de capteurs est étroitement liée à la vie nodale. Cette dernière, quant à elle, dépend essentiellement de la consommation d'énergie du nœud. Nous avons présenté dans ce chapitre quelques approches de conservation d'énergie dans les réseaux de capteurs sans fil.

Il existe bien évidemment beaucoup d'autres techniques de conservation d'énergie. La prochaine phase de notre travail consiste justement à présenter une solution protocolaire de niveau applicatif qui permet de minimiser l’énergie.

CHAPITRE III:

Protocole de routage Leach (Low-energy Adaptive Clustering Hierarchy)

1. Introduction

Les protocoles de routage hiérarchiques sont considérés comme étant des protocoles très favorables en termes d'efficacité énergétique. Deux grandes approches sont dérivées de ce type de protocoles: l'approche basée sur les chaînes (chaine-based approach) dont l'idée de formation de chaînes a été proposée pour la première fois dans l'algorithme PEGASIS, et, l'approche basée sur les groupes (cluster-based approach).

LEACH est considéré comme étant le premier protocole de routage hiérarchique basé sur la seconde approche.. Il combine l'efficacité en consommation d'énergie et la qualité de l'accès au média, et ce en se basant sur le découpage en groupes, en vu de permettre l'utilisation du concept de l'agrégation de données pour une meilleure performance en termes de durée de vie.

Dans ce chapitre, nous expliquerons l'architecture, l'algorithme et les caractéristiques du protocole LEACH.

2. Le routage hiérarchique

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

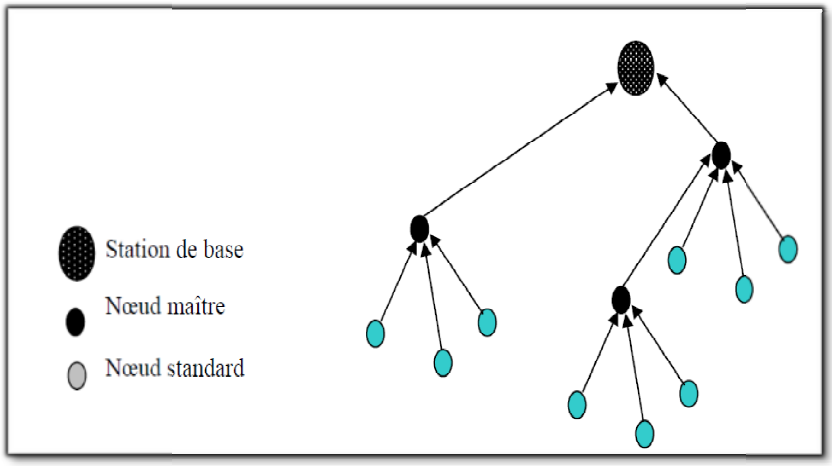


Fig.III.1. Le routage hiérarchique

3. Protocoles MAC utilisés par LEACH

Pendant son fonctionnement, le protocole LEACH appelle certains schémas des protocoles MAC qui seront détaillés dans cette section pour mieux comprendre son déroulement. Les nœuds doivent avoir une certaine capacité de calcul pour supporter différents protocoles MAC. Deux versions des protocoles MAC pour l'accès au media sont alors proposées pour les RCSF :

- ✓ l'accès aléatoire
- ✓ l'allocation fixe. [LNA04]

3.1.Accès aléatoire

Dans ces schémas, les nœuds qui possèdent des données à transmettre doivent essayer d'obtenir l'autorisation pour l'accès au media tout en réduisant les collisions avec les transmissions des données des autres nœuds.

Le schéma d'accès multiple avec surveillance de porteuse CSMA (Carrier Sense Multiple Access) sur lequel se base le protocole LEACH est l'un des schémas d'accès aléatoire. [ISA07b]

Lorsqu'un nœud veut transmettre un message, il examine le média pour vérifier s'il est libre ou occupé par un autre nœud. Dans le cas où le media est libre, ce nœud pourra émettre son message afin d'éviter les collisions. Cela dit, des nœuds peuvent émettre des données en même temps, ce qui mène à des collisions. Il est nécessaire donc que celles-ci soient détectées et que la récupération de données soit effectuée et que ces données soient retransmises.

Si les retransmissions se passent encore en même temps, d'autres collisions vont se produire. Une solution à ce problème consiste à introduire que chaque nœud attende un délai aléatoire avant de retransmettre ses données, ce qui réduit la probabilité d'une autre collision. [STE04]

3.2.Allocation fixe

Les schémas à allocation fixe permettent d'allouer pour chaque nœud le media de transmission suivant des intervalles de temps (schéma TDMA) ou un schéma de codage particulier (schéma CDMA).

Étant donné que chaque nœud est attribué en exclusivité à un intervalle, il n'y a pas de collisions entre les données. Toutefois, les schémas à allocation fixe s'avèrent inefficaces lorsque tous les nœuds n'ont pas de données à transmettre. En effet, ces intervalles sont affectés à des nœuds qui n'ont pas besoin de les utiliser. [PRE05]

3.2.1. TDMA

Le schéma d'accès multiple à répartition de temps ou TDMA (Time Division Multiple Access) permet de diviser le temps en intervalles (time-slot) attribués à chaque nœud (voir figure III-1). Ainsi, un seul nœud a le droit d'accès au canal (il utilise toute la plage de la bande passante du canal), mais doit émettre ses données pendant les intervalles de temps qui lui sont accordés. [STE04]

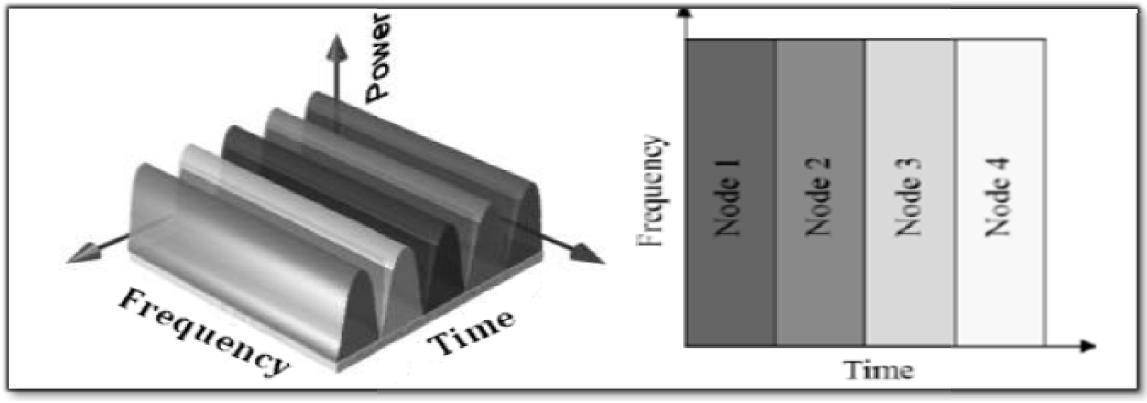


Fig. III.2. Diagrammes représentant le protocole MAC TDMA.

3.2.2. CDMA :

Le schéma d'accès multiple par répartition en code ou CDMA (Code Division Multiple Access) permet de côtoyer plusieurs nœuds simultanément (voir figure III-2). En effet, il ne divise ni la plage de fréquences ni l'intervalle de temps. Ainsi, des nœuds peuvent émettre leurs données continuellement et selon une large plage de fréquence. Le protocole CDMA utilise des techniques d'étalement de spectre afin d'éviter les collisions entre les transmissions simultanées des nœuds. [ERI01]

La figure suivante est un diagramme qui représente le protocole MAC CDMA :

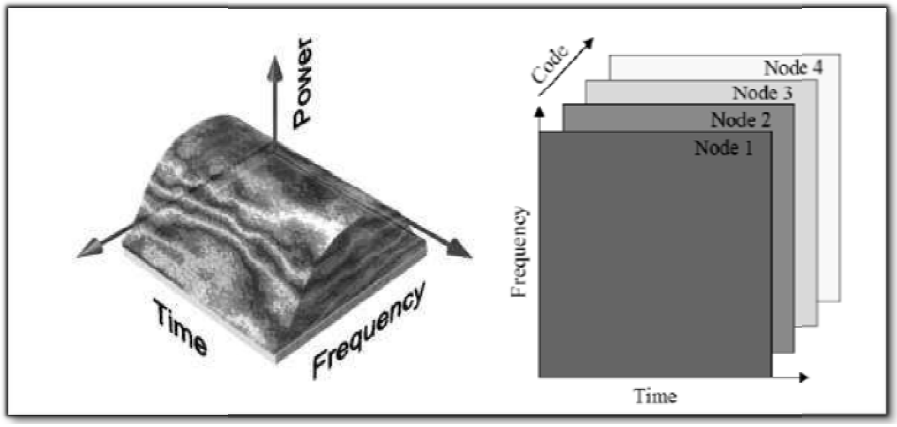


Fig. III.3. Diagrammes représentant le protocole MAC CDMA. [ERI01]

4. Architecture de communication de LEACH

L'architecture de communication de LEACH consiste, de façon similaire aux réseaux cellulaires, à former des cellules basées sur l'amplitude du signal, et utiliser les têtes de cellules comme routeurs vers le nœud puits. Ces cellules sont appelées groupes (clusters), quant aux têtes : chefs de groupes (cluster-heads CH). Les chefs de groupes sont choisis de façon aléatoire selon un algorithme spécifique d'élection basé sur une fonction de probabilité qui prend en compte différents critères comme l'énergie disponible des nœuds.
Comme la figure III-3 l'indique, les nœuds sont chargés de collecter des données, les envoyer à leurs CH qui les agrègent et transmettent, à leur tour, les résultats d'agrégation au nœud puits selon une communication unicast (à un seul saut).

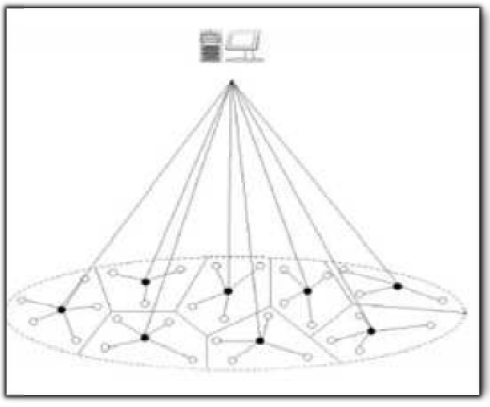


Fig. III.4. Architecture de communication du protocole LEACH.

Les CH ont pour mission d'assurer les fonctions les plus coûteuses en énergie, à savoir la communication avec le nœud puits qui est supposé éloigné, ainsi que tous les traitements de données (agrégation, fusion et transmission de données) afin de réduire la quantité des données transmises. Ce dispositif permet d'économiser l'énergie puisque les transmissions sont uniquement assurées par les CH plutôt que par tous les nœuds du réseau. Par conséquent, LEACH réalise une réduction significative de la dissipation d'énergie. [SAM06]

5. Algorithme détaillé de LEACH

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

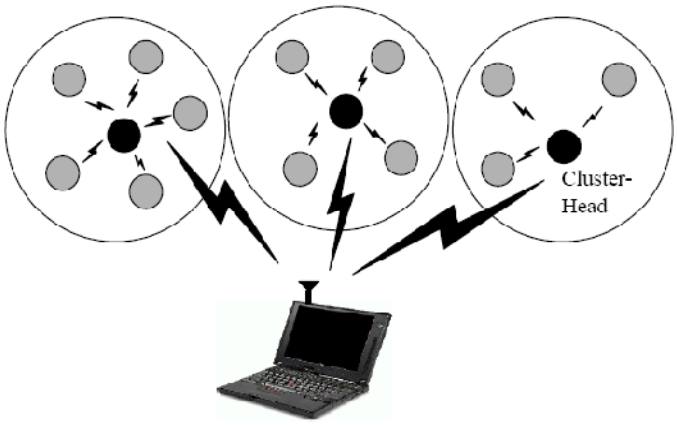


Fig.III.5. Algorithme de routage LEACH

5.1. Phase d'initialisation

Comme l'indique la figure III.6, la phase d'initialisation est composée de 3 sous phases:

- ✓ Phase d'annonce
- ✓ Phase d'organisation des groupes
- ✓ Phase d'ordonnancement.

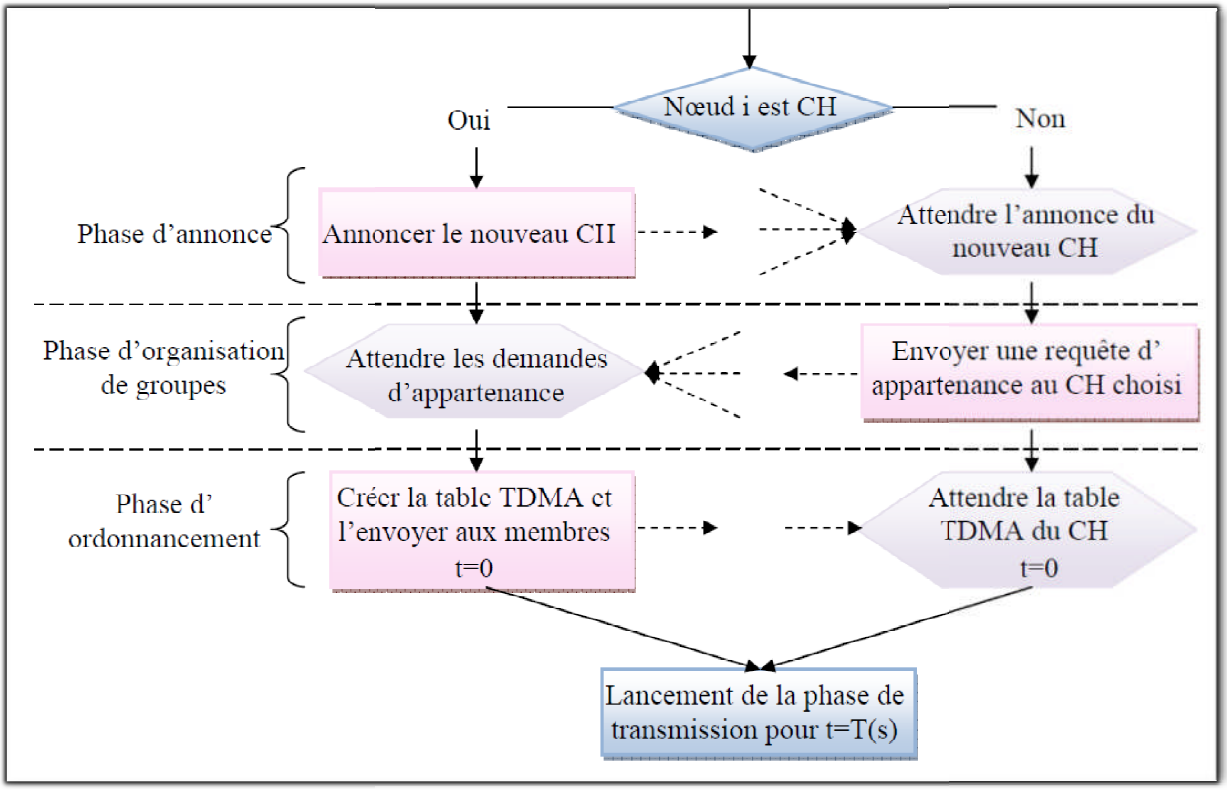


Fig. III.6. Opérations de l'étape d'initialisation de LEACH [DJA08].

5.1.1. Phase d'annonce

Avant de lancer cette phase, on désire avoir un certain nombre de CH. Ce nombre, que l'on note K, est fixe et il est inchangé durant tous les rounds. On estime que le pourcentage optimal du nombre de CH désirés devrait être de 5% à 15% du nombre total de nœuds. [DJA08] Si ce pourcentage n'est pas respecté, cela 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 de travail important qui leur est demandés.

Cette phase commence par l'annonce du nouveau round par le nœud puits, et, par la prise de décision locale d'un nœud pour devenir CH avec une certaine probabilité Pi(t) au début du round r+1 qui commence à l'instant t. Chaque nœud i génère un nombre aléatoire entre 0 et 1.

Si ce nombre est inférieur à Pi(t), le nœud deviendra CH durant le round r+1. Pi(t) est calculé en fonction de K et de round r [WEN00]:

$$\text{Nombre(CH)}=\Sigma^{N}_{i}\ Pi(t)= K$$

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 :

$$P_i(t) = \frac{\text{le nombre de CH désirés}}{\text{Le nombre de nœuds qui n'ont pas encore été élus CH durant les r rounds précédents}}$$

$$P_i(t) = \begin{cases} \dfrac{K}{N-k*(r \bmod N/k)} & : Ci(t)=1 \\ 1 & : Ci(t)=0 \end{cases} \quad \dots (1)$$

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 vrai semblablement une énergie résiduelle suffisante que les autres et ils pourront être choisis. Le terme $\sum^{N}_{i=1}$ Ci(t) représente le nombre total des nœuds éligibles d'être CH à l'instant t. Il est égal à : Σ Ci(t) = N- K*(r **mod** N/K) … (2)

5.1.2. Phase d'organisation de groupes

Après qu'un nœud soit élu CH, il doit informer les autres nœuds non-CH 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 non-CH 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. Par ailleurs, elle permet de garantir que les nœuds appartiennent au CH qui requière le minimum d'énergie pour la communication. 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. [WEN00] Chaque membre informe son CH de sa décision. Une fois que le CH ait reçu la demande, il lui envoie un message d'acquittement.

5.1.3. Phase d'ordonnancement

Après la formation des groupes, chaque CH agit comme un centre de commande local pour coordonner les transmissions des données au sein de son groupe. Il crée un ordonnanceur (schedule) TDMA et assigne à chaque nœud membre un slot de temps durant lequel il peut transmettre ses données. L'ensemble des slots assignés aux nœuds d'un groupe est appelé frame. La durée de chaque frame diffère selon le nombre de membres du groupe.
Par ailleurs, afin de minimiser les interférences entre les transmissions dans des groupes adjacents, chaque CH choisit aléatoirement un code dans une liste de codes de propagation CDMA. Il le transmet par la suite à ses membres afin de l'utiliser pour leurs transmissions. [SAC04]

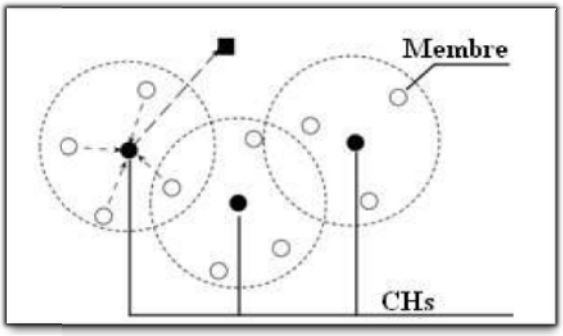


Fig. III.7. Interférence lors d'une communication dans LEACH [ABD06].

5.2.Phase de transmission

Cette phase est plus longue que la phase précédente, et permet la collecte de données captées. En utilisant l'ordonnanceur TDMA, les membres émettent leurs données captées pendant leurs propres slots. Cela leur permet d'éteindre leurs interfaces de communication en dehors de leurs slots afin d'économiser leur énergie. Ces données sont 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 seront élus CH, une seule fois, tout au long des rounds précédents. Dans ce cas, le round est réinitialisé à 0.

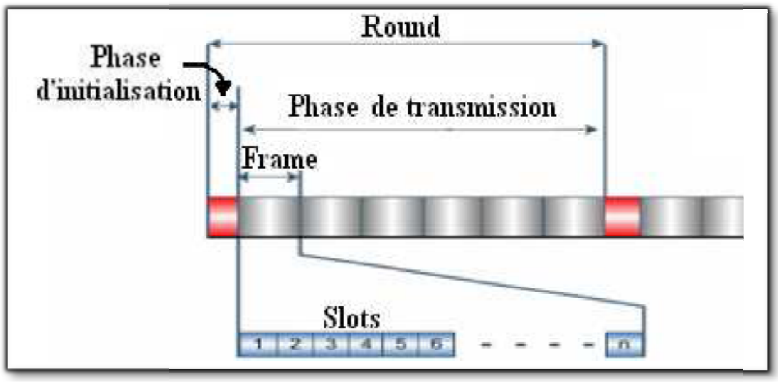


Fig. III.8. Répartition du temps et différentes phases pour chaque round. [YAS07]

Dans [39] Parveen &al. citent certaines remarques mettant en cause l'efficacité du protocole LEACH en consommation d'énergie, nous listons ces remarque dans ce qui suit :

- Les auteurs présument que ce protocole peut étendre la durée de vie du réseau par un facteur de 8, comparé aux autres protocoles basés sur le routage multi-sauts ou a regroupement statique.
- Les auteurs n'ont pas considéré, durant l'étape de simulation décrite dans le papier, la possibilité de destruction des nœuds par des actions hostiles, il est très bien connu, cependant, qu'un nœud capteur doit prévoir la possibilité de déploiement dans les environnements hostiles.
- La dissipation d'énergie est supposée égale à 50 nJ/bit, tandis que la meilleure technologie disponible offre une dissipation de 115nj/bit. Bien que les auteurs ont mentionné ce fait dans leur papier, ils n'ont pas fournit d'arguments suffisants pour supporter cette contrainte.
- Il a été proposé dans LEACH qu'une certaine fraction prédéterminé de nœud est sélectionné comme chefs de groupe, une méthode qui détermine la valeur optimale de cette fraction a été également donnée, mais le problème qui traite la distribution géographique uniforme des chef de groupes à travers le réseau en entier n'a pas été adressée. Par conséquent, il est possible d'avoir une grande concentration de chef de groupe dans une partie du réseau, alors que la partie majeure des nœuds capteurs se trouve sans aucun chef de groupe dans leur région.
- Dans l'étude liée à la simulation, les auteurs ont considéré des réseaux comportant 100 nœuds capteur, ce nombre est généralement plus grand et peut facilement

atteindre plusieurs milliers de nœuds. Ceci, peut avoir un impact considérable sur les résultats de simulation finaux liés à l'énergie consommée et la durée de vie du réseau.

6. Avantages et inconvénients de LEACH

Le protocole LEACH engendre beaucoup d'avantages en ce qu'il offre comme bonne manipulation de ressources du réseau en respectant plusieurs contraintes telle que la consommation d'énergie.

Bien que LEACH économise la consommation d'énergie par un facteur de 8 [LNA04] comparé à la transmission directe, grâce à l'agrégation de données et la réutilisation de largeur de bande, un nombre d'inconvénients restent plus ou moins apparents.

Dans ce qui suit, on cite quelques avantages et inconvénients du protocole LEACH.

6.1. Avantages

• Protocole auto-organisateur basé sur le groupement adaptatif

LEACH est complètement distribué, autrement dit, les nœuds prennent leurs décisions de façon autonome et agissent de manière locale et n'ont pas besoin d'une information globale ni d'un système de localisation pour opérer de façon efficace. De plus, la collection de données est faite périodiquement (l'utilisateur n'a pas besoin de toutes les données immédiatement). Pour exploiter cette caractéristique, ce protocole introduit un groupement adaptatif, c'est-à-dire, il réorganise les groupes après un intervalle de temps aléatoire, en utilisant des contraintes énergétiques afin d'avoir une dissipation d'énergie uniforme à travers tout le réseau. [SRA03]

• Faible énergie pour l'accès au média

Le mécanisme de groupes permet aux nœuds d'effectuer des communications sur des petites distances avec leurs CH afin d'optimiser l'utilisation du média de communication en la faisant gérer localement par un CH pour minimiser les interférences et les collisions.

• Compression locale (agrégation)

Les CH compressent les données arrivant de leurs membres, et envoient un paquet d'agrégation au nœud puits afin de réduire la quantité d'informations qui doit lui être transmise. Cela permet de réduire la complexité des algorithmes de routage, de simplifier la gestion du réseau, d'optimiser les dépenses d'énergie et enfin de rendre le réseau plus évolutif (scalable).

6.2. Inconvénient

- On pourra ne pas avoir des CH durant un round si les nombres aléatoires générés par tous les nœuds du réseau sont supérieurs à la probabilité Pi(t).
- Les nœuds les plus éloignés du CH meurent rapidement par rapport aux plus proches.
- Le protocole LEACH ne peut pas être appliqué à des applications temps-réel du fait qu'il résulte en une longue latence.
- Il n'est pas évident que les CH soient uniformément distribués. Donc, il est possible que les CH puissent être concentrés dans une partie du réseau. Par conséquent, certains nœuds n'auront pas des CH dans leurs voisinages.
- Le protocole LEACH n'est pas sécurisé. Aucun mécanisme de sécurité n'est intégré dans ce protocole. Ainsi, il est très vulnérable même aux simples attaques. Donc, un attaquant peut facilement monopoliser le réseau et induit à son disfonctionnement.

7. Conclusion

Nous avons essayé lors de ce chapitre de présenter en générale le fonctionnement du protocole Leach de niveau MAC et routage de réseaux de capteurs sans fils et leurs différentes solutions proposées pour résoudre le problème de conservation d'énergie. Dans le chapitre suivant, nous allons essayer de se concentrer sur l'implémentation du ce protocole

CHAPITRE IV :

Simulation et Mise en œuvre

1. Introduction

Le déploiement d'un réseau de capteurs nécessite une phase de tests avant sa mise en place afin de s'assurer du bon fonctionnement du réseau. L'expérimentation réelle s'avère quelques fois très coûteuse. Néanmoins, une solution moins coûteuse existe : la simulation à évènements discrets.

La simulation des réseaux de capteurs consiste principalement en la reproduction du comportement des nœuds capteurs et des interactions entre eux. C'est une étape incontournable pour l'évaluation des modèles d'application ou des protocoles de communication. De plus, la simulation offre un gain considérable en temps, une flexibilité en permettant la variation des paramètres et une meilleure visualisation des résultats.

Ce chapitre débute par la présentation du simulateur Tossim. Nous détaillons par la suite les étapes d'implémentation de notre protocole sur TinyOS 1.x. Le chapitre s'achève par l'interprétation des résultats obtenus à l'issue des tests effectués sur LEACH.

2. Environnement de simulation

Dans ce qui suit, nous présentons l'environnement de simulation utilisé.

2.1.TOSSIM

TOSSIM est le simulateur de 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.

2.2.Choix du simulateur PowerTossim

Le simulateur TOSSIM n'a pas la capacité de vérifier le taux d'énergie dissipée pendant l'exécution des applications. Cependant, le besoin de vérifier la consommation énergétique dans un RCSF a un intérêt primordial. L'université de Harvard a conçu le simulateur PowerTOSSIM qui surmonte ce problème. Ce nouveau simulateur est intégré dans
TOSSIM. Il permet de calculer le total d'énergie consommée par chaque composant constituant l'architecture de TOSSIM (LED, radio, CPU, etc.).

Pour simuler ces composants (voir figure IV-1), on fait appel au module PowerState. Ce dernier engendre des messages de transition d'états d'énergie (power state transition messages) pour chaque composant. Ces messages peuvent être combinés avec un modèle

d'énergie pour générer en détail les consommations d'énergie. Pour se faire, un fichier programmé en langage python, intitulé « postprocess.py » est utilisé.

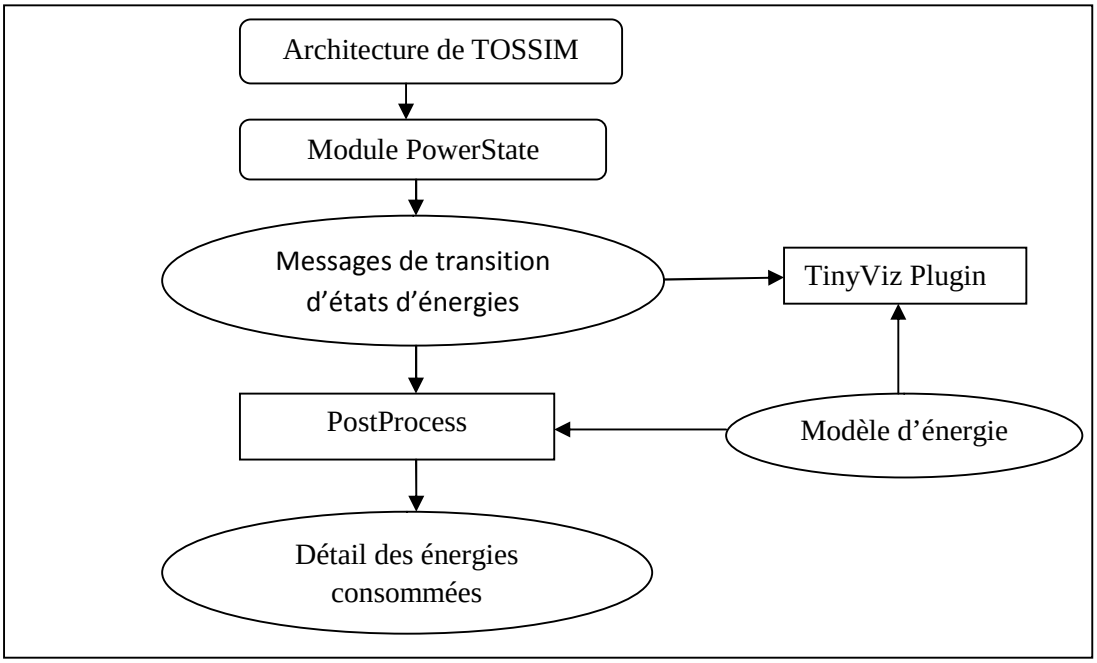


Fig. IV.1. Architecture de PowerTOSSIM

Pour une compréhension moins complexe de l'activité d'un réseau, PowerTOSSIM peut être utilisé avec une interface graphique, TinyViz, permettant de visualiser de manière intuitive le comportement de chaque capteur au sein du réseau.

- ✓ **Description :**

TinyViz est une application graphique qui donne un aperçu de notre réseau de capteurs à tout instant, ainsi que des divers messages qu'ils émettent. Il permet de déterminer un délai entre chaque itération des capteurs afin de permettre une analyse pas à pas du bon déroulement des actions.

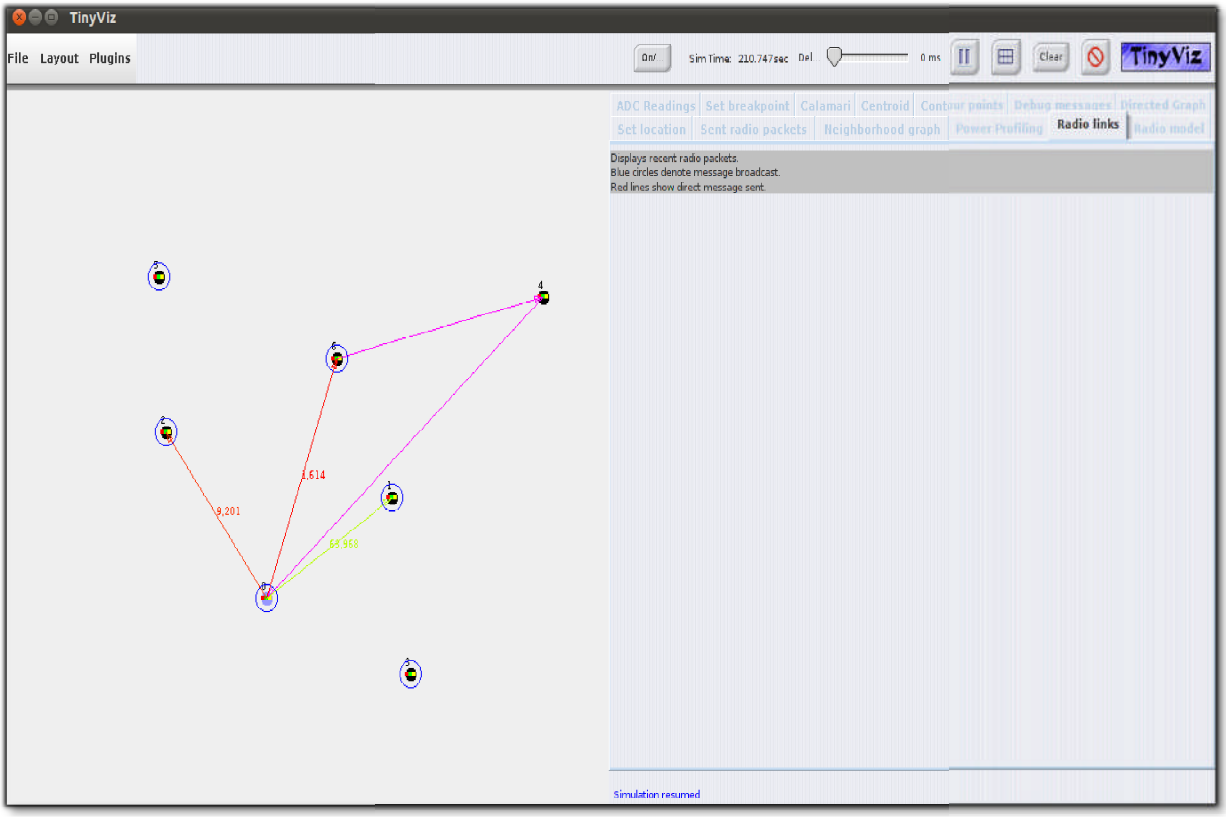


Fig. IV.2. Interface graphique TinyViz

2.3.Implémentation du protocole LEACH

Dans cette section, nous décrivons les structures de données ainsi que les principaux commandes et événements nécessaires pour l'implémentation du protocole LEACH.

2.3.1. Structures de données

Le paquet dans TinyOS est envoyé dans une structure appelée TOS_Msg, qui est contenue dans un champ « int8_t data[TOSH_DATA_LENGTH] ». Les structures de données du paquet diffèrent selon le rang du nœud (puits, CH ou membre).

a) Le nœud puits

```
typedef struct PUITS
{uint16_t ID; //l'identificateur du puits qui correspond à tos_local_address=0
uint8_t round; //le round courant
float probability; //la probabilité que chaque nœud devienne CH
uint8_t Depth; //la puissance du signal d'un CH dans le réseau
        }PUITS;
```

b) Le nœud CH

```
typedef struct CLUSTER_HEAD
                                        {
uint16_t ID_CH; //l'identificateur de chaque CH qui correspond à tos_local_address
uint16_t ID_MEMBRE; //l'identificateur du membre qui appartiendra à ce CH
uint8_t data_agre; //la donnée agrégée à envoyer au noeud puits
uint16_t SLOT_ATT; //le slot attribué à chaque membre
uint16_t FREQ; //la fréquence avec laquelle un membre envoie sa donnée
}CLUSTER_HEAD;
```

c) Le nœud membre

```
typedef struct MEMBRE
{
uint16_t ID_MEMBRE;//l'identificateur de chaque membre qui correspond à
tos_local_adress
uint16_t ID_CH; //l'identificateur du CH auquel appartiendra le nœud membre
                                uint8_t energ; //l'énergie captée
```

2.3.2. Evénements et commandes

Nous citons ici les principaux événements utilisés pour l'implémentation de LEACH.

Evénement	Sortie	Fonction
LEACH_ReceiveMsg.receive(TOS_MsgPtr pmsg)	TOS_MsgPtr	Reception du round
ANNONCE_ReceiveMsg.receive(TOS_MsgPtrr pmsg)	TOS_MsgPtr	Annonce du CH
ORGANISATION_ ReceiveMsg.receive(TOS_MsgPtrr pmsg)	TOS_MsgPtr	Formation de groupes
SLOT_ ReceiveMsg.receive(TOS_MsgPtrr pmsg)	TOS_MsgPtr	Reception des slots
AGGREGATION_ ReceiveMsg.receive(TOS_MsgPtrr pmsg)	TOS_MsgPtr	Réception du puits des résultats d'agrégation
ReqRelayTimer.fired ()	Result_t	Relai des annonces du round
RoundTimer.fired ()	Result_t	Envoi du nouveau round par le nœud puits

2.3.3. Déroulement

Dans cette partie, nous expliquons et déroulons les phases de l'algorithme LEACH en faisant appel à TinyViz. Un fichier de configuration est créé et permet à TinyViz de se lancer avec des paramètres spécifiés. Ces derniers représentent : le nombre et l'emplacement des capteurs, la durée de la simulation et les plugins que nous souhaitons activer dès le début de la simulation comme Debug Messages. A propos des captures d'écran de TinyViz, nous nous limitons, dans cette étape, sur la partie où l'on peut visualiser les capteurs.

1. Déclenchement et relai du nouveau round, et, annonce des CH

La figure IV-3 représente les transmissions broadcast qui se passent durant différentes étapes de l'algorithme LEACH. Une transmission broadcast est repérée par un cercle bleu.

Le nœud puits envoie un broadcast aux nœuds voisins pour l'annonce du round. Ses voisins prennent le relai en envoyant à leur tour selon une transmission broadcast. De plus, nous pouvons voir que le nœud 4 est élu CH. Cet évènement est marqué par l'activation des LED rouges des CH. Ensuite, le CH 4 diffuse une annonce pour signaler son statut

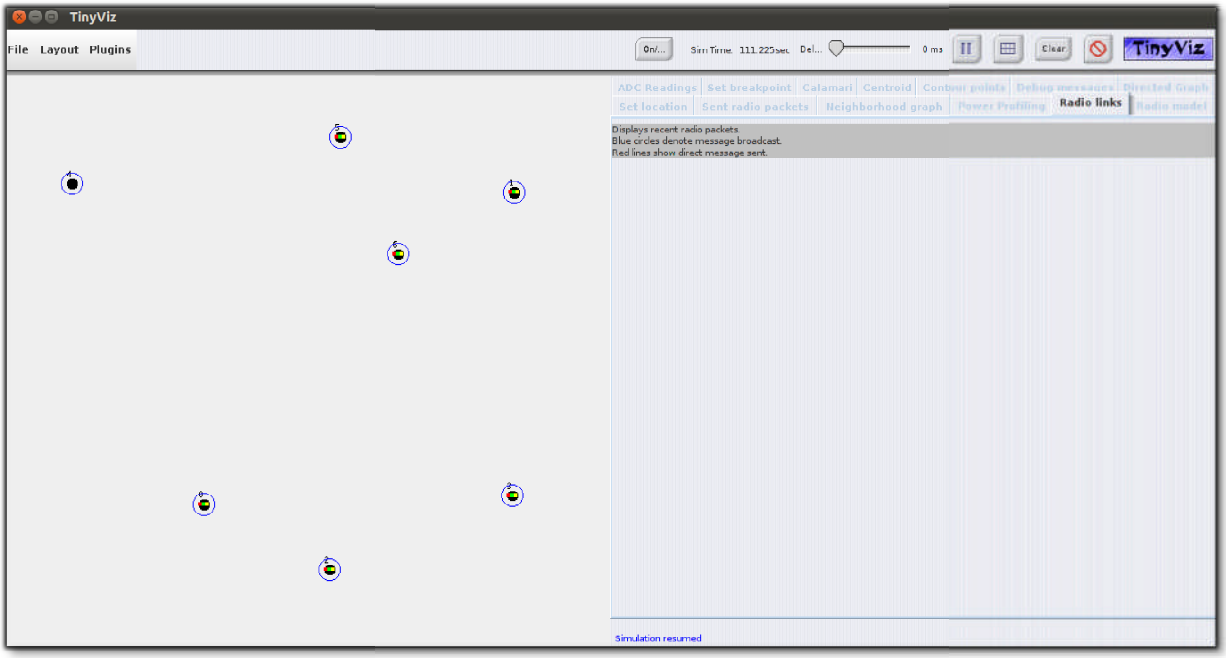


Fig. IV.3. Déclenchement et relai du nouveau round, annonce du CH 4

2. Formation de groupes et envoi des températures

La figure IV-4 représente quelques transmissions unicast qui se passent durant différentes étapes de l'algorithme LEACH. Une transmission unicast est repérée par une flèche. Durant la première étape, les nœuds non-CH répondent à l'annonce du CH le plus proche. La figure IV-2 illustre la formation du cluster du CH 4**.** Quant à la seconde étape, chaque membre capte la température et attend le début de son slot pour qu'il puisse l'envoyer à son
CH. Nous avons utilisé une application qui retourne la température sur une zone donnée afin de pouvoir valider l'implémentation du protocole LEACH.

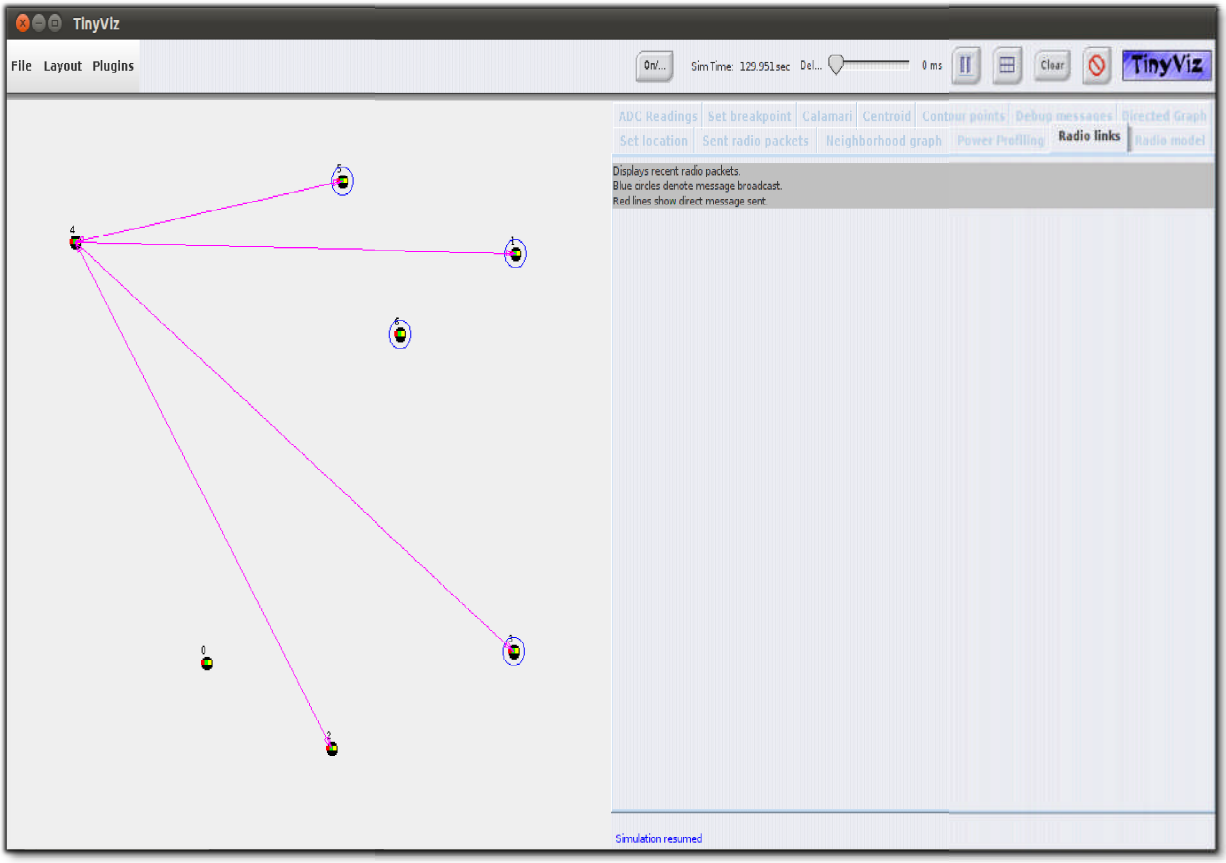


Fig. IV.4. Formation de groupes et envoi des températures

3. Envoi des résultats d'agrégation des températures au nœud puits

Dans la figure (IV.5) le CH (4) agrège les températures reçues et envoie son résultat d'agrégation au nœud puits.

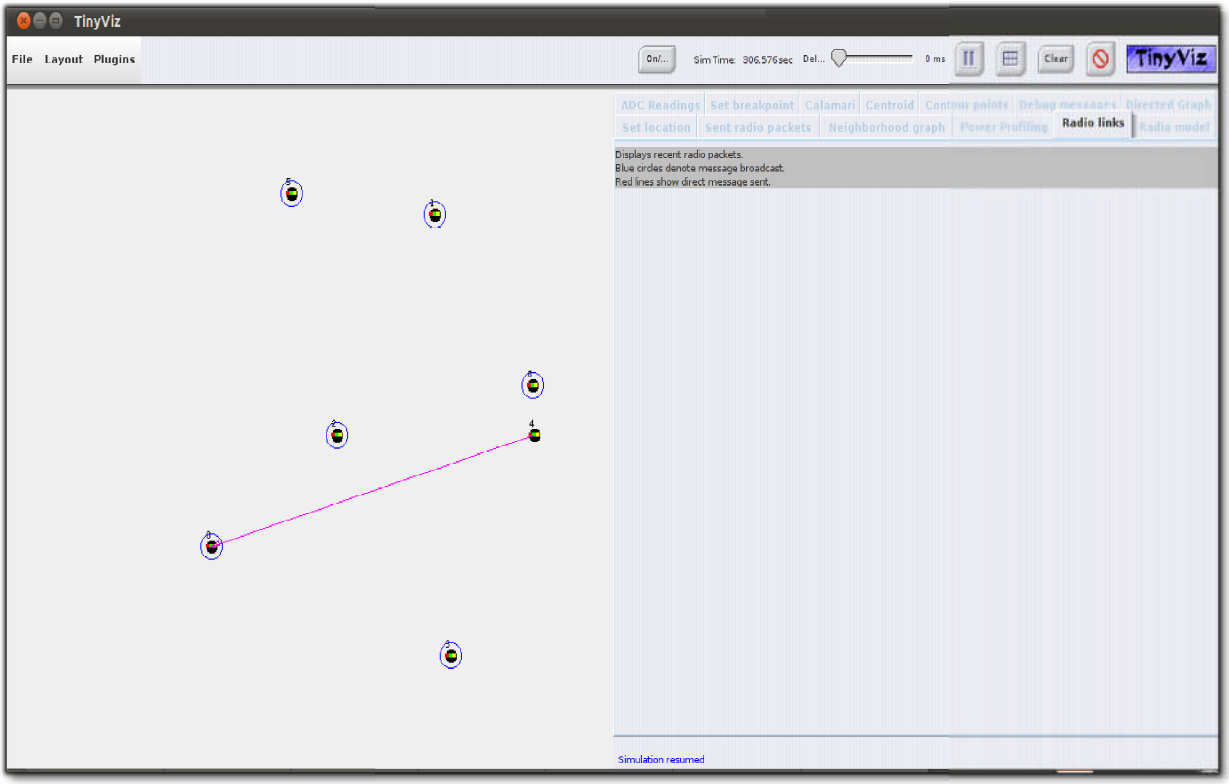


Fig. IV.5. Envoi du résultat d'agrégation du CH 4 au nœud puits

3. Proposition de la stratégie des arbres de routages optimal

La stratégie des arbres de routage optimal consiste à améliorer la technique de transmission de données entre les nœuds membres du cluster avec le nœud maitre, c'est-à-dire lors de la transmission de données des membres vers le cluster head ils empreintront le chemin le plus court afin d'optimiser l'énergie.

Dans ce qui suit on a illustré la stratégie utilisé.

3.1. Routes à consommation d'énergie minimale :

Au moment du routage, le choix de la route à consommation d'énergie minimale peut suivre quatre approches, a fin de décrire chacune d'elles nous utilisons la figure (IV.6) où le nœud T sera considéré comme nœud capteur source qui surveille le phénomène. Chaque nœud intermédiaire est caractérisé par la valeur (PA) qui désigne l'énergie disponible au niveau du nœud, de la même manière, chaque liaison entre deux nœuds est caractérisée par l'énergie (αi) nécessaire pour transmettre un paquet de données entre les deux extrémités.

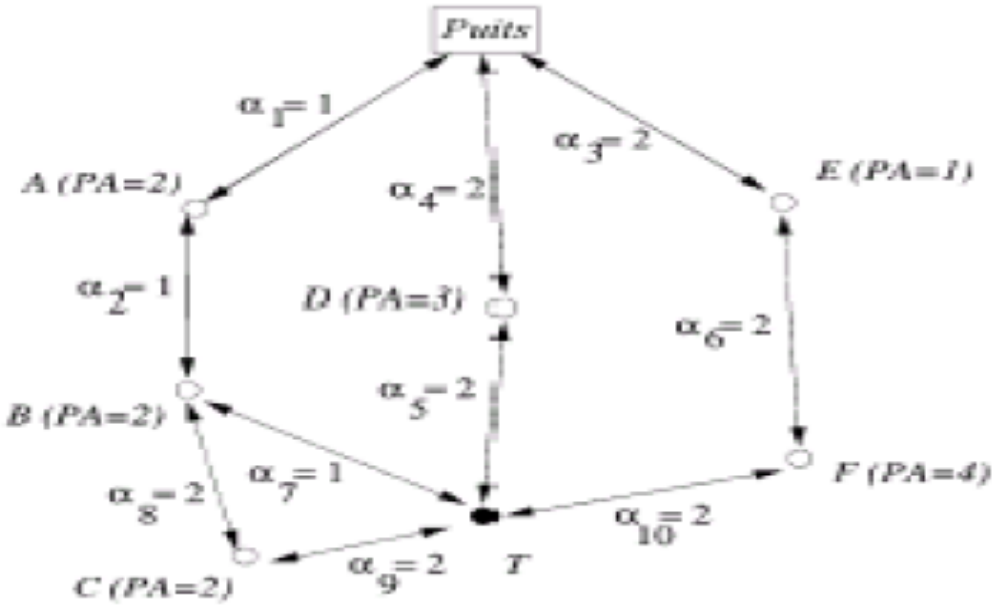


Fig.IV.6. Routes efficaces en consommation d'énergie

Pour transmettre un paquet du nœud T au nœud puits, 4 routes sont possibles :

• Route 1 : Puits-A-B-T, avec l'énergie totale disponible PA = 4 et l'énergie totale nécessaire α=3,

• Route 2 : Puits-A-B-C-T, avec l'énergie totale disponible PA = 6 et l'énergie totale nécessaire α=6

• Route 3 : Puits-D-T, avec l'énergie totale disponible *PA* = 3 et l'énergie totale nécessaire α=4

• Route 4 : Puits-E-F-T, avec l'énergie totale disponible *PA* = 5 et l'énergie totale nécessaire α=6.

Voici les différentes routes empreintes pour atteindre le chemin le plus court:

1. Route à énergie disponible maximum

La première approche pour le choix des routes efficaces en consommation d'énergie consiste à prendre celle qui contient les nœuds possédant, ensemble, un maximum d'énergie totale disponible. Cette quantité est égale à la somme des énergies (PA) de chaque nœud appartenant à cette route.

En se basant sur cette approche, la route 2 est sélectionnée dans l'exemple de la figure (IV.6), mais nous remarquons, tous de suite, que cette route contient les nœuds de la route 1 en plus du nœud C, ceci dit, cette route ne peut être la plus efficace en consommation d'énergie malgré qu'elle possède le maximum d'énergie totale disponible.

Par conséquent, il est important, dans un algorithme de routage, de ne jamais considérer comme alternatives les routes qui peuvent être dérivées par extension de celles allant du nœud capteur vers le nœud puits. L'élimination de la route 2 mène alors à choisir la route 4 comme route efficace en consommation d'énergie suivant la première approche.

2. Route à énergie de transmission minimum

Cette approche consiste à choisir la route qui consomme le minimum d'énergie pour transmettre un paquet entre le nœud capteur et le nœud puits. Suivant l'exemple, la route 1 est celle qui consomme le minimum d'énergie.

3. Route à nombre de sauts minimum

La route sélectionnée est celle qui traverse un nombre minimum de nœuds intermédiaires pour atteindre le nœud puits. Dans le cas de l'exemple de la figure 8, la route 4 est celle qui est la plus efficace suivant ce schéma de sélection.

Il est à noter que, les deux approches de « routes à sauts minimum » et « route à énergie de transmission minimum » sélectionnent les mêmes chemins quand les énergies de transmission αi sont identiques.

4.* Route à nœud ayant le maximum des minimums des énergies disponibles *PA

Cette approche favorise la route dans la quelle l'énergie disponible minimum traversée est plus grande que toutes les autres énergies disponible minimums des autres routes. Dans la figure 8, la route 3 est celle qui est favorable pour cette approche, la route 1 vient après.

La stratégie de ce schéma de sélection évite l'utilisation du nœud qui a une faible énergie disponible pour lui assurer une durée de vie plus longue.

Pour notre approche consiste à choisir la deuxième approche.

4. Simulation et évaluation de performances

Pour évaluer les performances du protocole OR-LEACH (optimal route Leach), nous avons procédé à le comparer au protocole de routage LEACH.

4.1.Métriques à évaluer

Pour pouvoir comparer les performances d'OR-LEACH avec celles de LEACH, il est commode de mesurer la métrique suivante :

- **Consommation énergétique**

Nous nous sommes intéressés essentiellement à la consommation d'énergie des nœuds puisqu'elle constitue un paramètre primordial pour la détermination de la durée de vie d'un RCSF.

4.2.Paramétrage de la simulation

Avant de lancer les simulations, nous devons ajuster certains paramètres qui sont présentés par le tableau

Paramètres du contexte de la simulation	
Nombre de nœuds du réseau	7
Délai de la simulation	400 secondes : les résultats de la simulation adéquats durant l'état transitoire
Nombre d'itération (simulations)	5 : les résultats que nous allons présenter sont une moyenne de 5 simulations pour un même scénario
Taille de paquet de données	29 octets : c'est le paquet de transmission de TinyOS

TAB . Paramètres du contexte de la simulation.

4.3.Résultats de la simulation

1. **Consommation énergétiques dans Leach :** on constate que la consommation énergétique dans le protocole Leach est plus élevée que le protocole OR-Leach par exemple la consommation énergétique dans le nœud 2 est un peu élevé à celle du nœud implémenté dans le protocole OR-Leach

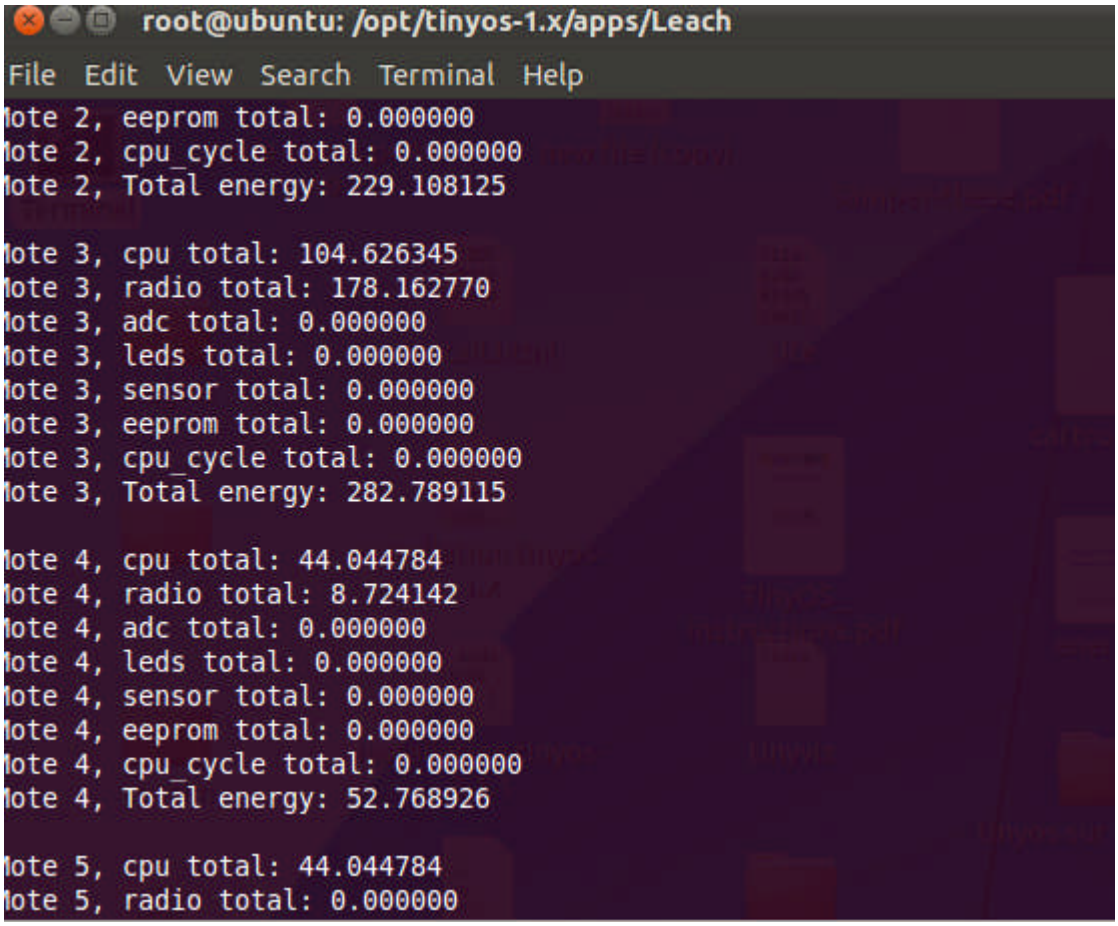


FIG.IV. 7. Consommation d'énergie dans le protocole Leach

2. Consommation énergétiques dans OR-Leach :

Pa contre la consommation énergétique dans le protocole OR-Leach est optimisée par exemple dans le nœud la consommation énergétique dans le nœud 2 est un peu élevé à celle du nœud implémenté dans le protocole Leach

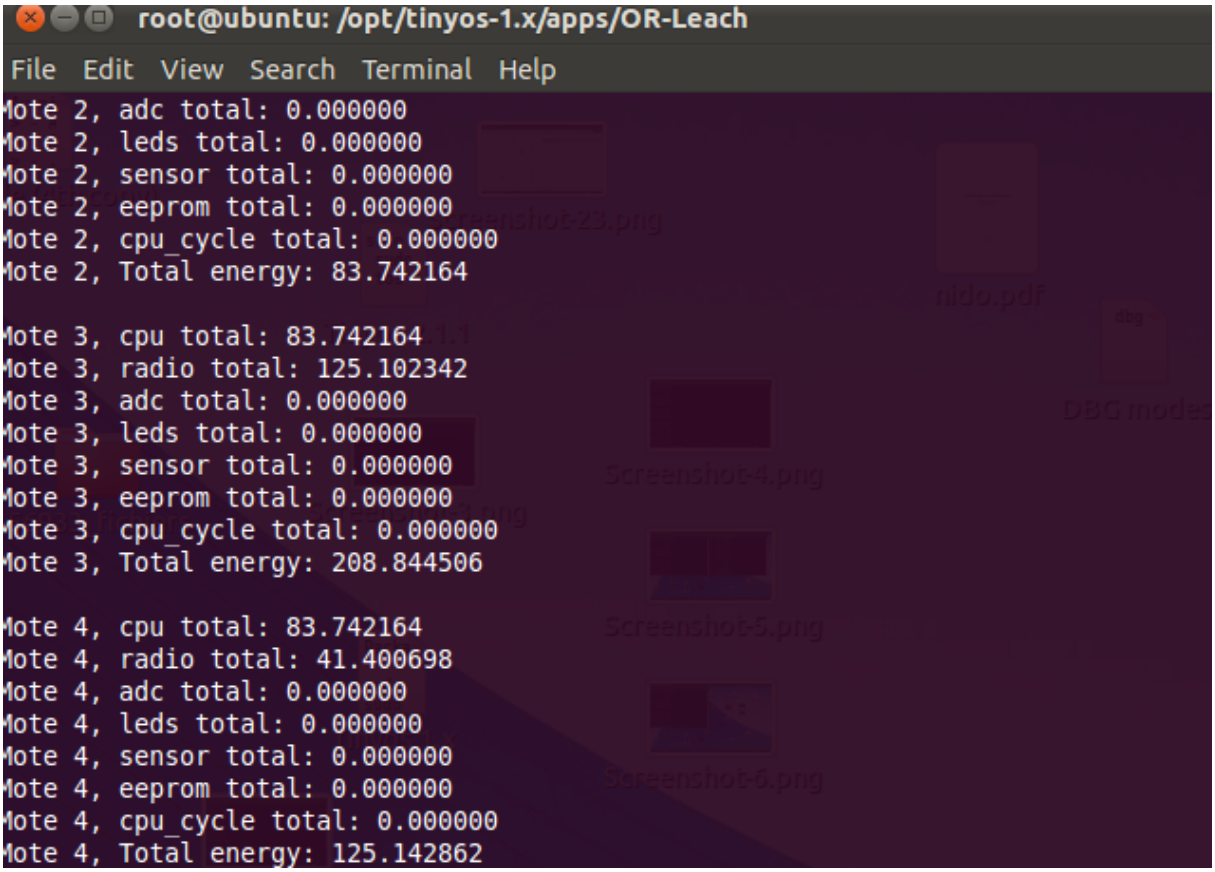


FIG.IV. 8. Consommation d'énergie dans le protocole OR-Leach

4.4.Interprétations des résultats

Dans cette partie, nous évaluons la métrique citée el la comparons pour les deux protocoles Leach et le protocole que nous avons amélioré qui sera appelé OR-Leach (Optimal Route-Leach). Par la suite nous simulons ces deux protocoles

4.4.1. Consommation énergétique

Pour avoir les tests en consommation d'énergie, nous avons eu recours au fichier trace généré par le simulateur PowerTOSSIM.

A. Consommation d'énergie par nœud sur un échantillon de 7 nœuds

Dans ce test, nous avons mesuré le taux de consommation d'énergie des CH par rapport aux nœuds membres pour les deux protocoles LEACH et OR-Leach

	Leach	OR-Leach
Consommation énergétique des MBR	15,50	13,22

(Joules)		
Consommation énergétique des CH (Joules)	19,54	15, 23
Energie additionnelle des CH par rapport aux membres	21,23	15,43

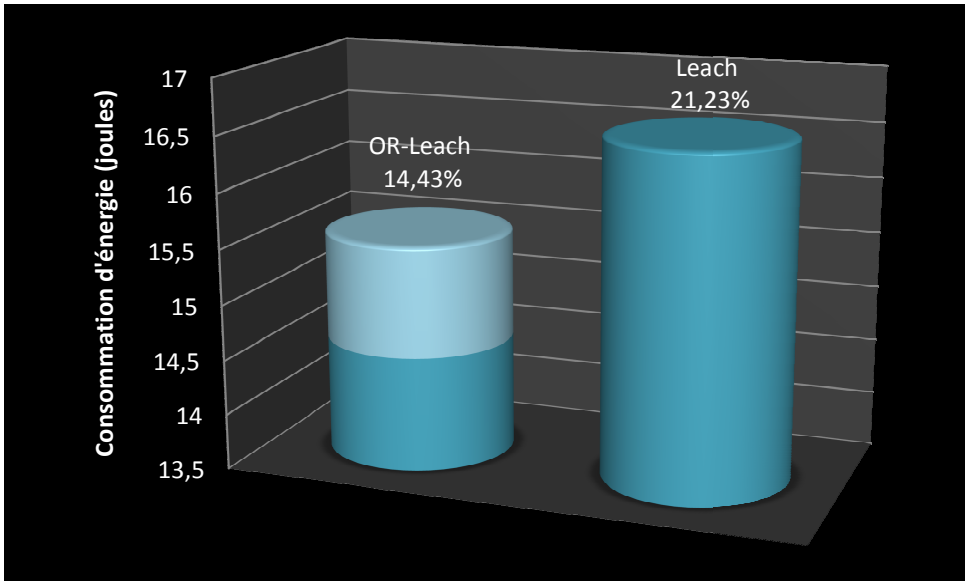


FIG.IV. 9. Consommation d'énergie des CH par rapport aux membres dans le protocole Leach et OR-Leach

5. Conclusion

Dans ce chapitre, nous avons présenté l'implémentation ainsi que l'évaluation des deux protocoles LEACH et OR-Leach. Le système d'exploitation TinyOS est utilisé

Il consiste une programmation entière en langage NesC et une simulation avec PowerTOSSIM.

Par ailleurs, nous n'avons constaté que les tests de performances effectués sur la consommation d'énergie, ont montré que le protocole OR-LEACH répond bien aux critères de performances souhaités

Conclusion générale

Conclusion générale

Les réseaux de capteurs suscitent un grand intérêt dans le domaine de la recherche. En effet plusieurs travaux ont été proposés afin d'améliorer leurs performances. L'objectif de notre travail consistait en la mise en œuvre d'un protocole de routage qui puisse garantir une bonne gestion de la consommation d'énergie et une longue durée de vie du réseau.

Les réseaux de capteurs sans fil ont connu au cours de ces dernières années un formidable essor aussi bien dans l'industrie que dans le milieu universitaire. Cela est principalement attribuable à l'ampleur sans précédent des possibilités qu'offre cette technologie. Toutefois, les réseaux de capteurs sans fil doivent aussi faire face à d'importants défis de conception en raison de leurs capacités de calcul et de stockage limitées et surtout de leur dépendance à l'égard d'une énergie limitée fournie par une batterie. L'énergie est une ressource critique et constitue souvent un obstacle majeur au déploiement des réseaux de capteurs qui prétendent à l'omniprésence dans le monde de demain. Cette thèse a porté sur l'efficacité énergétique dans les réseaux de capteurs sans fil, avec un accent particulier sur la conservation d'énergie au niveau application.

La stratégie des arbres de routage optimal nous permet d'optimiser l'énergie d'un pourcentage de 7%.

Comme perspectives envisageables pour améliorer les performances de notre stratégie nous proposons de :

- Gérer la panne des nœuds membres par l'introduction d'un mécanisme permettant la détection de la panne des membres par leur cluster-heads.
- Introduire le paramètre de la distance entre les nœuds dans la fonction d'élection des CHs en utilisant les coordonnées géographiques des nœuds. Ceci peut se faire en dotant les nœuds d'un système de localisation GPS. Cette méthode permet d'avoir la position exacte des capteurs mais présente néanmoins un coût élevé. D'autres méthodes moins coûteuses peuvent être envisagées telles que les méthodes de triangulation et la méthode des centroïdes qui donnent une estimation de la position des nœuds.

Bibliographie

[ABD06] : Abdelraouf Ouadjaout, « La Sécurité et la Fiabilité du Routage dans les Réseaux de Capteurs Sans Fils», Mémoire de magistère, USTHB, Algérie, 2006.

[AKY 01] : I.F. AKYILDIZ, W.SU, Y.SANKARASUBRAMANIAM and E.CAYIRCI, « Wireless sensor networks: A Survey », Georgia Institute of Technology, 2001.

[AKY 02]: I.F. AKYILDIZ, W.SU, Y.SANKARASUBRAMANIAM and E.CAYIRCI, « A Survey on Sensor Networks », Georgia Institute of Technology, 2002.

[ARA 08] :Salem ARAB, « réseau de capteurs sans fils et un satellite LEO » Projet de fin d'étude pour l'obtention du diplôme de magistère, Université A.Mira, Bejaia, 2008.

[BEC 09] : W. BECHKIT, « Un nouveau protocole de routage avec conservation d'énergie dans les réseaux de capteurs sans fil », Mémoire pour l'obtention du diplôme d'ingénieur d'état en informatique, École nationale Supérieure d'Informatique, 2009

[DJA08] : Djallel Eddine Boubiche, «Protocole de routage pour les réseaux de capteurs sansfil», Mémoire de magistère, Université de l'Hadj Lakhdar, Batna, Algérie, 2008.

[DOU 06]: N.DOUFENE et H.HADJAMMAR, « Routage dans les réseaux de capteurs: Optimisation du protocole Directed Diffusion », Projet de fin d'étude pour l'obtention du diplôme d'ingénieur d'état, Institut National de formation en Informatique, 2006

[ERI01]: Eric Lawrey, «The suitability of OFDM as a modulation technique for wireless telecommunications, with a CDMA comparison», Projet d'ingéniorat, Université James Cook, Australie, 2001.

[HAM 07] : A. HAMZI, « Plateforme pour l'aide à la conception et à la simulation des réseaux de capteurs sans fil », Mémoire pour l'obtention du grade de magistère en informatique, Institut National de formation en Informatique, 2007.

[HAC 03]: A. HAC, « Wireless Sensor Network Designs », Edition: John Wiley & Sons Ltd, 2003

[HEI 00]: W.R. HEINZELMAN, A. CHANDRAKASAN, and H.BALAKRISHNAN, « Energy-Efficient Communication Protocol for Wireless Microsensor Networks », Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.

[ILY 04] : M. ILYAS and I. MAHGOUB, « Handbook of sensor networks: compact wireless and wired sensing systems », CRC Press, 2004.

[ISA07b] : Isabelle Guérin Lassous, « Autonomic Computing : Accès au médium radio », Cours M2 Recherche RTS, RTS5, Page(s) : 43-95, Université de Lyon, 15 Septembre 2007.

[KHE 04] : L. KHELLADI et N. BADACHE, « », Rapport de recherche, Laboratoire LSI USTHB, 2004.

[GEI 04] : J.GEIER, « Wireless networks first step », Cisco Press, Août 2004.

[KHE 04] : L. KHELLADI et N. BADACHE, « », Rapport de recherche, Laboratoire LSI USTHB, 2004.

[LNA04] :Lyes Khelladi, Nadjib Badache « Les réseaux de capteurs: état de l'art », Rapport de recherche, Algérie, Février 2004.

[POT 00]: G.J.POTTIE and W.J.KAISER, « Wireless integrated network sensors », Communications of the ACM, May 2000.

[POM 04]: C. POMALAZA-REZ, « Wireless Ad Hoc & Sensor Networks », University of Oulu, Finland, 2004

[PRE05]: Preetha Radhakrishnan, "Enhanced routing protocol for graceful degradation in wireless sensor networks during attacks", Thèse d'ingéniorat, Université de Madras, Chennai, Décembre 2005.

[SAM06] : Samra Boulfekhar, «Approches de minimisation d'énergie dans les réseaux de capteurs», Mémoire de Magistère, Université Abderahmane Mira de Bejaïa, 2006.

[SAC04] : Sachin Mujumdar, « Prioritized Geographical Routing In Sensor Networks », Thèse, Université Vanderbilt, Mai 2004.

[STE04] :Sébastien Tixeuil, Ted Herman, « Un algorithme TDMA réparti pour les réseaux de capteurs », INRIA Projet Grand Large, Universités Iowa et Paris-Sud XI, 2004.

[WEN00] Wendi Beth Heinzelman, «Application-Specific Protocol Architectures for Wireless Network », IEEE Transactions on Wireless Communications, Massachusetts Institute of Technology, June 2000.

[YAS07] Yasser Romdhane, « Evaluation des performances des protocoles S-MAC et Directed Diffusion dans les réseaux de capteurs », Projet de fin d'études, Ecole Supérieure des Communications de Tunis (Sup'Com), 2006 / 2007.

TinyOS, 2007 [En Ligne]. Adresse URL : http://www.tinyos.net/.
NesC, 2007 [En Ligne]. Adresse URL : http://nescc.sourceforge.net/

ANNEXE A :

Le système d’exploitation TinyOS

1. Introduction

Suite aux différents problèmes vécut par les réseaux de capteurs (problème énergétiques et de mémoire), l'université de Berkeley a développé alors un système d'exploitation minime destiné pour ces réseaux : TinyOS**.** Il est orienté "composants" afin de faciliter l'implémentation de ces réseaux, tout en minimisant la taille du code afin de respecter les contraintes de mémoire des composants matériels

TinyOS, comme les applications tournant dessus, a été écrit en NesC**.** Ce langage a été inventé pour répondre aux attentes des systèmes embarqués. Il possède une syntaxe proche de C, supporte le système multitâche de TinyOS et défini des mécanismes pour architecturer et "linker" des composants logiciels en un système embarqué robuste.

Dans ce chapitre, je vais introduire et développé le mode de fonctionnement de la plateforme TinyOS, ainsi que le langage NesC et d'autre logiciels de simulation.

2. TinyOS (Tiny Microthreading Operating System)

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

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

Fig. A.1. Cigle du système d'exploitation TinyOS.

3. Propriétés de la plateforme TinyOS

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

• **Disponibilité et sources**

TinyOS est un système principalement développé et soutenu par l'université américaine de Berkeley, qui le propose en téléchargement sous la licence BSD et en assure le suivi;

• **Event-driven**

Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des évènements qui se produisent instantanément. Ainsi, l'activation de tâches, leur interruption ou encore la mise en veille du capteur s'effectue à l'apparition d'évènements, ceux-ci ayant la plus forte priorité. Ce fonctionnement basé sur les évènements (Event-driven) s'oppose au fonctionnement dit temporel (time-driven) où les actions du système sont gérées par une horloge donnée ;

• **Langage**

TinyOS a été programmé en langage NesC que nous allons détailler plus tard ;

• **Préemptif**

Le caractère préemptif d'un système d'exploitation précise si celui-ci permet l'interruption d'une tâche en cours. TinyOS ne gère pas ce mécanisme de préemption entre les tâches mais donne la priorité aux interruptions matérielles. Ainsi, les tâches entre-elles ne s'interrompent pas mais une interruption (sous forme d'un évènement) peut stopper l'exécution d'une tâche ;

• **Temps réel**

Lorsqu'un système est dit « temps réel » celui-ci gère des niveaux de priorité dans ses tâches permettant de respecter des échéances données par son environnement.

• **Consommation**

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

4. Gestion et 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 de mémoire et pas de pointeurs de fonctions. Bien sur cela simplifie grandement l'implémentation. Par ailleurs, il n'existe pas de mécanisme de protection de la mémoire sous TinyOS ce qui rend le système particulièrement vulnérable aux crash et corruptions de la mémoire.

Cette description de mémoire nous a permis de concevoir un contrôleur pour la gestion et l'organisation des différents types d'actions dans une application TinyOS

5. Modèle d'exécution de TinyOS

Pour maintenir une grande efficacité requise par les réseaux de capteurs, TinyOS utilise une programmation par évènement. Ce modèle permet un très haut niveau de concurrence pour un espace très réduit de mémoire.

L'implémentation des composants de TinyOS s'effectue en déclarant des tâches, des commandes ou des évènements

1. Programmation par évènement

Dans un système basé sur la programmation par évènement, chaque exécution est partagée entre les différentes taches de traitement. Dans TinyOS, chaque module est désigné pour fonctionner en attendant continuellement de répondre aux évènements inattendus. Quand un évènement est signalé, le traitement correspondant est exécuté. Une fois totalement terminé, la main est redonnée au système pour continuer sa tâche antérieure.

En plus de l'efficacité de l'allocation du CPU, la programmation par évènement permet d'obtenir des opérations économiques en énergie. Il est très important aussi pour la consommation d'énergie que les applications signalent la fin de leurs évènements et l'utilisation du CPU. Dans TinyOS, les taches associées avec un évènement sont exécutées très rapidement après leurs signalisations. Une fois terminé, le CPU entre en veille en attendant une nouvelle réception d'évènement.

2. Tâches

Un des facteurs limitant la programmation par évènement est la longue exécution des tâches qui peut interrompre d'autres programmes importants. Si l'exécution d'un évènement ne finit jamais, toutes les autres fonctions vont être interrompues. Pour éviter ce problème, TinyOS fourni un mécanisme d'exécution appelé tâche.

Une tâche est un bout de programme qui s'exécute jusqu'à la fin sans interférer avec les autres évènements. Les tâches sont utilisées pour effectuer la plupart des blocs d'instruction d'une application.

A l'appel d'une tâche, celle-ci va prendre place dans une file d'attente de type FIFO mais elle ne sera exécuté que lorsque il n'y a plus d'évènements. En plus les tâches peuvent être interrompues à tout moment par des évènements.

6. Cibles possibles pour TinyOS

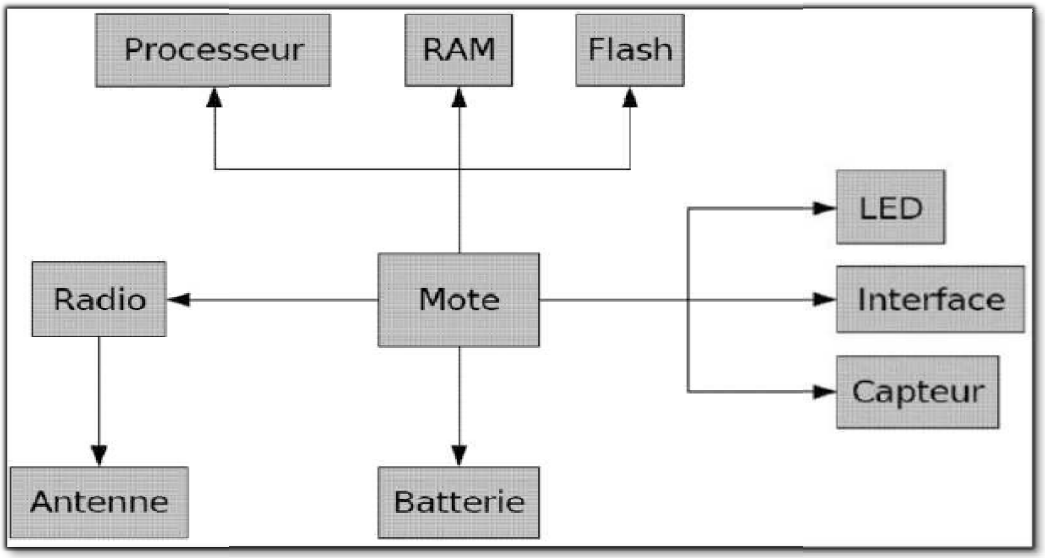


FIG.A.2. Architecture générale des cibles utilisant TinyOS

- **Mote, processeur, RAM et Flash**

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

- **Radio et antenne**

  Afin d'émettre, un capteur a besoin d'une antenne et d'une radio pour ajuster les fréquences hertziennes.

- **LED, interface, capteur**

  TinyOS est prévu pour mettre en place des réseaux de capteurs, on retrouve donc des équipement bardés de déférents types de détecteurs et autres entrées.

- **Batterie**

  Indispensable pour l'autonomie du capteur.

7. Package TinyOS

TinyOS est prévu pour fonctionner sur une multitude de plateformes, disponibles dès l'installation. En effet, TinyOS peut être installé à partir d'un environnement Windows (2000 et XP) ou bien GNU/Linux

8. Les outils de simulation TinyOS

TinyOS offre des outils de Simulation (Tinyviz/Tossim/PowerTossim)

A. Tossim

Le principal but de TOSSIM est d'effectuer une simulation très proche de ce qui ce passe dans les RCSFs dans le monde réel. De plus, il simule le comportement des applications de TinyOS à un niveau très bas. Le réseau est simulé au niveau des bits et chaque interruption dans le système est capturée. TOSSIM présente deux modèles de radios pour la communication :

- Le modèle simple : c'est le modèle utilisé par défaut où deux nœuds différents peuvent envoyer un paquet en même temps. Ce modèle ne présente aucune erreur de transmission.
- Le modèle lossy : les nœuds sont placés dans un graphe direct formé d'un couple (a, b). Ce modèle prend en considération les erreurs de transmission et la perte de paquets.

Pour une compréhension moins complexe de l'activité d'un réseau, TOSSIM peut être utilisé avec une interface graphique TinyViz. Cette application graphique réalisée en JAVA permet de visualiser à tout moment et de manière intuitive le comportement de chaque capteur au sein du réseau et offre une économie d'effort et une préservation du matériel.

B. PowerTossim

L'outil PowerTossim permet de faire des simulations de la même manière que TOSSIM sauf que celui-ci prend en considération la consommation d'énergie, 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.

- ✓ **Lancer PowerTOSSIM**

Pour récupérer l'énergie consommée par les nœuds du réseau, il faut passer par ces étapes:

1- Accéder à l'application à simuler et la compiler en tapant: **make pc**

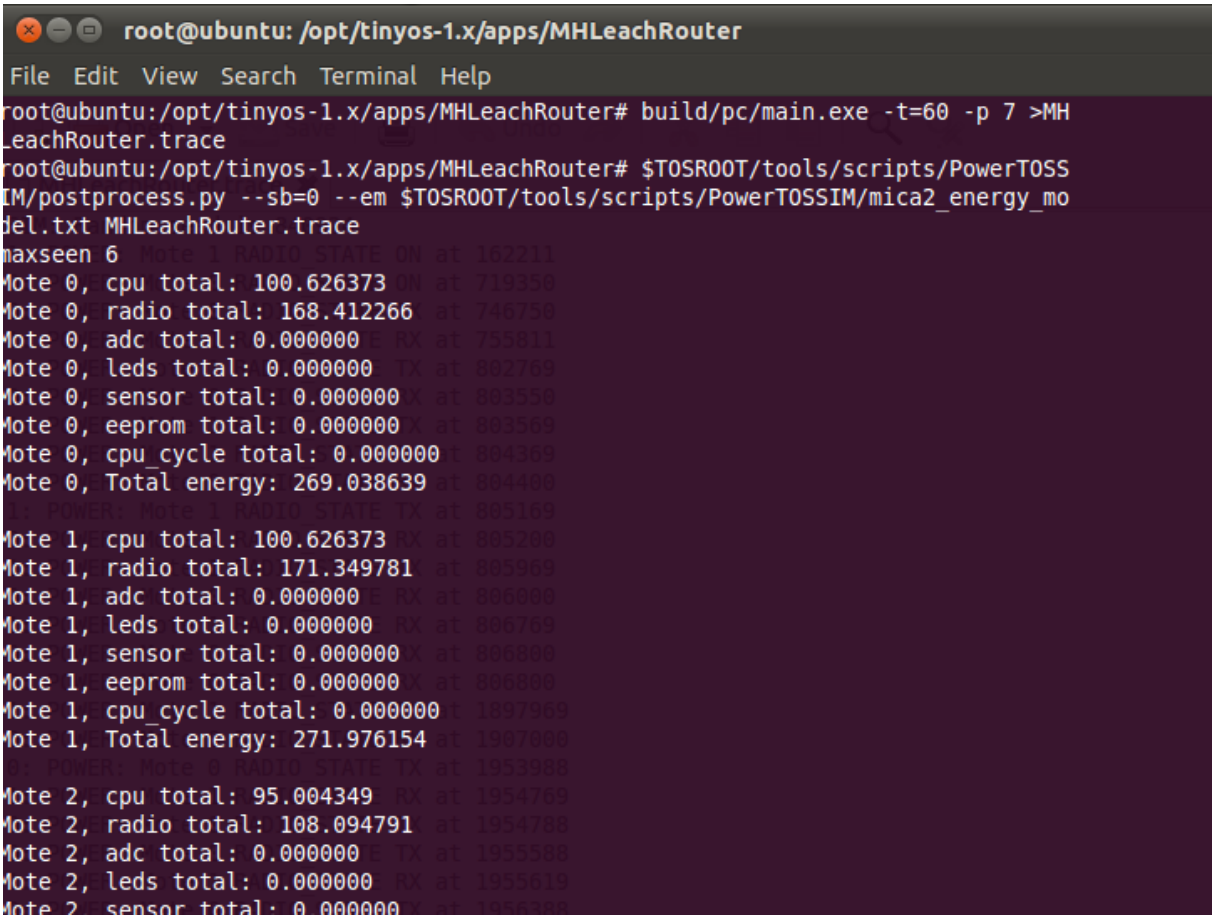
2- Taper **export DBG=power**

3- Exécuter **main.exe** en choisissant le temps de simulation avec **–t** et le nombre de nœuds du réseau avec **–p**. Une trace de simulation est enregistré dans un fichier dont l'extension est .trace .Pour se faire, taper **: /build/pc/main.exe -t=56 -p 7> NomApp.trace** (Le temps est égal à 60 secondes et le nombre de nœuds à 10)

4- Exécuter **postprocess.py** sur la trace de simulation en spécifiant les paramètres –sb et –em **/opt/tools/scripts/PowerTOSSIM/postprocess.py –sb=0 –em /opt/tools/scripts/Power TOSSIM/mica2_energy_model.txt NomApp.trace**

Le parameter –sb spécifie si les nœuds sont attachés à un autre noeud (i.e. embarqué). En outre, le paramètre –em spécifie le modèle d'énergie. Pour plus de détail sur l'utilisation d'autres paramètres de PowerTOSSIM, exécuter **postprocess.py --help**

5- Le résultat enregistre l'énergie totale utilisée par chaque composant sur chaque nœud. Il est sous la forme suivante :



6- Pour ne pas perdre ce résultat, il est commode de le sauvegarder dans un fichier texte. Pour se faire, Ajouter dans l'instruction de l'étape 4:

/opt/tools/scripts/PowerTOSSIM/postprocess.py –sb=0 –em /opt/tools/scripts/Power TOSSIM/mica2_energy_model.txt NomApp.trace > Result.txt

7- Pour avoir un résultat d'énergie plus détaillé, ajouter le paramètre **–detail** dans l'instruction de l'étape 4. Le résultat est enregistré automatiquement dans des fichiers textes dont le nombre est égal au nombre de nœuds simulés. Autrement dit, chaque fichier contient le détail de la consommation énergétique d'un seul nœud du réseau.

C. Tinyviz

Tinyviz est une application graphique qui donne un aperçu de notre réseau de capteurs à tout instant, ainsi que des divers messages qu'ils émettent. Il permet de déterminer un délai entre chaque itération des capteurs afin de permettre une analyse pas à pas du bon déroulement des actions, il possède aussi des options afin de pouvoir simuler la consommation d'énergie.

9. Guide d'installation

1. Procédure d'installation sous Windows XP

Ce guide propose l'installation du principal outil nécessaire au bon fonctionnement du système, notamment Cygwin (couche d'émulation de l'API Linux) qui permet d'avoir une interface Unix sous Windows. Cygwin est un environnement d'émulation Linux qui permet d'avoir un shell et de compiler et exécuter les programmes Linux (On dispose ainsi de gcc, apache, bash, etc.).

1- Télécharger le fichier tinyos-1.1.0-1is.exe de la source http://www.tinyos.net/dist-1.1.0/tinyos/windows/ .

2- Exécuter ce fichier pour installer la version 1.1.0 sous windows XP. L'installation se fait automatiquement. Un raccourci de Cygwin est sauvegardé sur le bureau.

3- Accéder à **C:\tinyos\cygwin\opt\tinyos-1.x\doc\tutorial\verifyhw.html** et suivre les étapes que contient cette page afin de vérifier si l'installation est bien réussie.

2. Procédure d'installation sous Ubuntu 10.10

Installation de TinyOS-1.x

- ✓ Ouvrir le fichier « sources.list » qui se trouve dans : /etc/apt/sources.list
- ✓ Copier se qui suit à la fin du fichier :

```
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
```

✓ Ajouter les packages suivant en tapant les commandes:

```
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
```

✓ **Les variables d'environnement:**
Copier se qui suit dans le fichier .bashrc:

```
# 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
# NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS 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-1.5.0-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/su
pport/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
```

- ✓ Ouvrir le Terminal sous Ubuntu et taper la commande suivante pour les mises à jour:

  $ sudo apt-get update

- ✓ Ensuite taper les commandes pour l'installation :
  - cvs -d:pserver:anonymous@tinyos.cvs.sourceforge.net:/cvsroot/tinyos login
  - cvs -z3 -d:pserver:anonymous@tinyos.cvs.sourceforge.net:/cvsroot/tinyos co tinyos-1.x
  - sudo mv tinyos-1.x /opt

10. Conclusion

Le plus gros avantage de TinyOS est qu'il est basé sur un fonctionnement événementiel, c'est-à-dire qu'il ne devient actif qu'à l'apparition de certains événements. Le reste du temps, le capteur se trouve en état de veille afin de garantir une durée de vie maximale aux faibles ressources énergétiques du capteur. TinyOS se distingue aussi par son caractère non préemptif, c'est-à-dire qu'il ne gère pas les interruptions entre tâches. Par contre il donne une priorité aux interruptions matérielles qui peuvent à tout moment stopper l'exécution d'une tâche. Pour terminer, TinyOS ne gère pas de "temps réel" car il n'est pas prévu pour manipuler des niveaux de priorité, pour mieux respecter les échéances dans les tâches.

ANNEXE B :

Le langage de programmation NesC

1. Présentation :

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

NesC est un langage conçus pour incarner les concepts structurant et le modèle d'exécution de TinyOS. C'est une extension du langage C orientée composant ; il supporte alors la syntaxe du langage C et il est compilé vers le langage C avant sa compilation en binaire.

2. Les Principales caractéristiques de NesC

NesC est constitue d'interfaces et de composants.

Une interface peut être utilisée ou peut être fournie. Les composants sont des modules ou des configurations.

Une application est un ensemble de composants, regroupés et reliés entre eux (voir figure B.1.).

Les interfaces sont utilisées pour les opérations qui décrivent l'interaction bidirectionnelle. Le fournisseur de l'interface doit mettre en application des commandes, alors que l'usager de l'interface doit mettre en application des événements.

Deux types de composants existent :

- Les modules, qui mettent en application des spécifications d'un composant.
- Les configurations, qui se chargeront d'unir différents composants en fonction des interfaces (commandes ou événements).

La figure suivante (figure B.2.) montre un diagramme de blocs dans lequel est décrit le processus de compilation pour une application TinyOS écrite en
NesC.

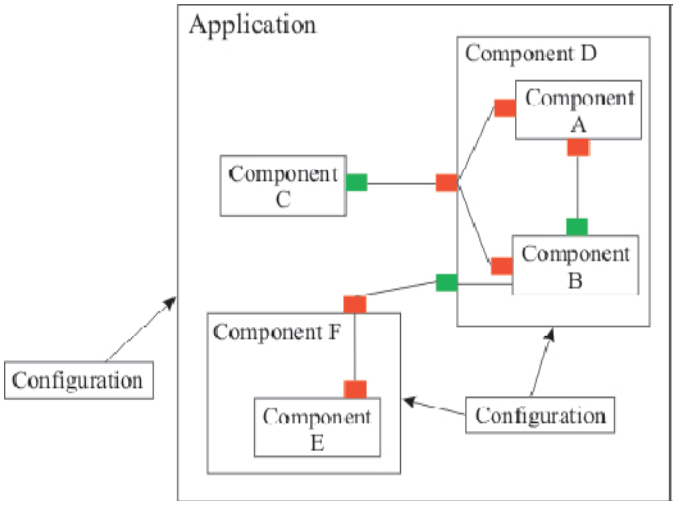


FIG.B.1.Architecture d'une application NesC

2. Les fichiers dans NesC

Les fichiers dans NesC sont classés en trois types : Interfaces, modules et configurations.

- ✓ **Interface** : Ce type de fichier déclare les services fournis et les services qui seront utilisés. Ils se trouvent dans le répertoire /tos/interface. (exemple :StdControl.nc).
- ✓ **Module** : Le type Module contient le code de l'application, en mettant en œuvre une ou plusieurs interfaces. (Exemple : BlinkM.nc)
- ✓ **Configuration** : Dans ces fichiers est déclarée la manière d'unir les différents composants et comment effectuer le contrôle des flux. (Exemple : Blink.nc).

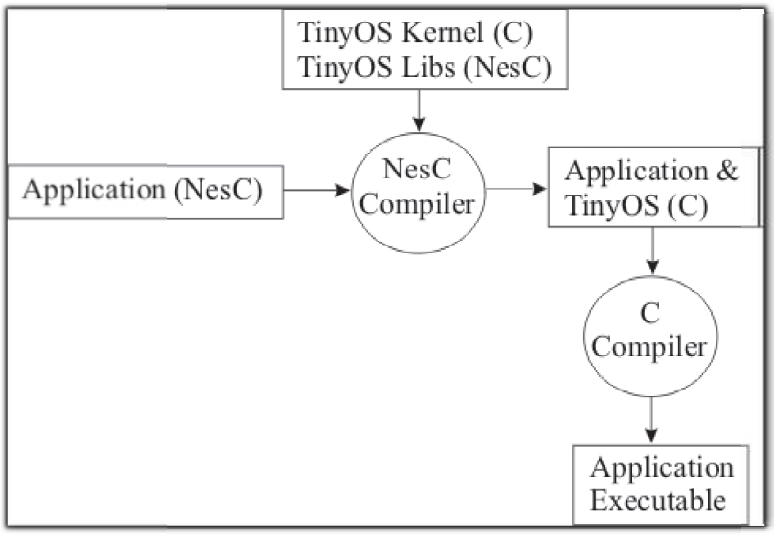


FIG.B.2.Processus de compilation

3. **Définition :**

1. Composants

TinyOS définit un nombre important de concepts qui sont exprimés dans NesC. D'abord, les applications NesC sont construites par des composants avec des interfaces bidirectionnelles définies. Aussi, NesC définit un modèle basé sur les tâches et les captures d'événements matériels, et détecte des éclatements d'information pendant la compilation.

Un composant, du point de vue de la programmation, est composé de plusieurs sections et l'ensemble de toutes ces sections donne lieu à la création de ce composant.

2. Implémentation

Cette section définit les connections entre les différents composants qu'utilise l'application. Dans cette section ((implementation)) sont donc principalement définis quels sont les composants qui fournissent les interfaces à notre application (ce sera généralement des composants primitifs). Habituellement nous devons utiliser les interfaces que nous fournissent d'autres composants, primitifs ou non primitifs, et en définitive pour chacune de ces interfaces, que nous utiliserons dans la création de notre composant, on doit obligatoirement définir des relations avec les composants qui fournissent ces interfaces. Le processus définissant ces relations s'appelle ((wiring)).

3. Configurations

C'est à cet endroit que l'on déclare les autres composants dont se servira l'application. Cette possibilité offerte par le langage permet de faire de la programmation modulaire et de réutiliser des composants préalablement définis.

La structure de la partie Configurations est la même que celle de la partie Implémentation.

4. Module

Cette partie du code est généralement plus étendue et c'est dans celle-ci que l'on programme réellement le comportement qu'on souhaite voir réalisé par l'application. Cette partie là est à son tour divisée en trois sous-sections :

Uses, Provides, Implementation.

4. **Types de données**

Les types de données qui peuvent être utilisés en NesC sont tous ceux que fournit le langage C standard plus quelques autres qui n'apportent pas de puissance de calcul mais qui sont très utiles pour la construction de paquets puisqu'ils fournissent à l'utilisateur le nombre de bits qu'ils occupent (ceci est important au moment de la transmission des informations par l'intermédiaire des ondes radio).

Quelques exemples de ces types additionnels :

– uint16 t : entier non signé sur 16 bits.

– uint8 t : entier non signé sur 8 bits.

– result t : utilisé pour savoir si une fonction a été exécuté avec succès ou non, c'est comme un booléen mais avec les valeurs SUCCESS et FAIL.(retour de fonction)

– bool : valeur booléenne qui peut être TRUE ou FALSE.

5. Types de fonctions en NesC

En NesC les fonctions peuvent être de types très variés. Il y a d'abord les fonctions classiques avec la même sémantique qu'en C et la façon de les invoquer est aussi la même qu'en C. Il y a aussi des types supplémentaires de fonctions : task, event, command. Les fonctions ((command)) sont principalement des fonctions qui sont exécutées de manière synchrone, c'est-à-dire que lorsqu'elles sont appelées elles sont exécutés immédiatement. La manière d'appeler ce genre de fonction est :

- ✓ call interface. nomFonction

Les fonctions ((task)) sont des fonctions qui sont exécutées dans l'application,
utilisant la même philosophie que les fils ou ((threads)), c'est principalement une fonction normale qui est invoquée de la manière suivante :

- ✓ post interface . nomTache

Immédiatement après son invocation, l'exécution du programme qui l'a invoqué se poursuit.
Les fonctions ((event)) sont des fonctions qui sont appelées quand on relèvera un signal dans le système, elles possèdent principalement la même philosophie que la programmation orientée événements, de sorte que, lorsque le composant reçoit un événement, on effectue l'invocation de cette fonction.

Il existe une méthode pour pouvoir invoquer manuellement ce type de fonctions :

- ✓ signal interface. nomEvenement

On utilise donc le mot clef call pour déclarer les événements et le mot clef post pour déclarer les tâches.

6. **Conclusion**

NesC est un langage de programmation qui présente de grands avantages pour le développement d'applications pour des systèmes embarqués, et particulièrement pour les réseaux de capteurs sans fil. Celui-ci apporte des outils pour une bonne programmation d'applications, d'abord en abstrayant les caractéristiques de modularité que présentent les réseaux de capteurs, et ensuite en facilitant la mise en œuvre de programmes pour ces systèmes.