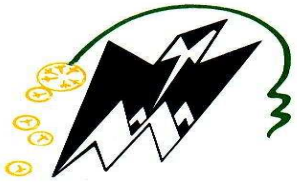
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou
Faculté du Génie Electrique et d'Informatique
Département d'Informatique



Mémoire Master II En Informatique
Spécialité : Systèmes Informatiques

Les Impacts De la Technique De Mise En Veille Sur la Consommation Energétique Dans Les Réseaux De Capteurs Sans Fil

Réalisé par :
Mlle ***MAIZ Zina***

Proposé par :
Mr ***DAOUI M.***

Promotion 2010/2011

Remerciements

Merci Dieu pour tout, pour le courage et l'entourage que tu m'as offert, pour tous ces moyens que tu as mis devant moi.

Je tien à remercier Mr DAOUI.M pour son encadrement, son soutien continuel et ses conseils judicieux. La qualité de son encadrement m'a offert une large marge de liberté intellectuelle et scientifique qui m'a permis de mener à bien ce laborieux travail. Je le remercie pour toutes les connaissances qu'il m'a transmises durant toutes ces années. Il était le meilleur enseignant et guide, la motivation qu'il nous fait révéler remplace tous les moyens.

Je remercie les membres du jury d'avoir accepté d'évaluer mon travail, je suis honoré par votre présence. Je vous offre ma reconnaissance et mon respect.

Je remercie également tous mes enseignants qui m'ont appris la persévérance, qui m'ont aidé à réussir et dépasser tout obstacle.

Ma gratitude la plus sincère s'adresse également à ma chère amie MAHIOUT Lilia, à son frère Said et ses très chers parents, je les remercie pour leur aide précieuse, pour leur tendresse, leur soutient, et à la place qui m'ont données dans leur vie. J'ai pour vous une gratitude infinie et une reconnaissance illimitée.

Mes remerciements spéciaux également pour Dr.MAHIOUT Maya, l'ange protecteur des enfants.

Je remercie Kahina CHELLI, Je remercie mes amis qui ont cru en moi et tous ceux qui m'ont encouragés.

Dédicaces

Mes parents ! Je vous dédie mon travail, je vous dédie mon succès, je vous dédie toute tâche que je réalise dans la vie, qui pourra vous rendre fières de moi !

Mes frères, ma sœur ! Je vous le dédie, mais pas pour que vous soyez fières de moi, je souhaite juste que vous fassiez autant que moi, ou même mieux!

A la mémoire de ma chère Grand-mère, ma meilleure école de tendresse et de persévérance, je souhaite que dieu ait pitié de son âme !

Je le dédie aux personnes souffrantes, aux orphelins, à ceux qui n'arrivent pas à trouver leurs voies, je souhaite que dieu leur offre une part de ces biens!

Résumé

La forte demande dans le domaine des réseaux ambiants, permet actuellement d'envisager le développement de réseaux de capteurs sans fil. Ces réseaux facilitent le suivi et le contrôle à distance de l'environnement physique avec une meilleure précision.

La mise en œuvre des réseaux de capteurs sans fil (WSN) pose de nombreux problèmes tel que le routage de données, la localisation, et les contraintes à la consommation d'énergie, ceci afin d'optimiser l'autonomie des noeuds qui constituent le réseau. Dans les réseaux de capteurs sans fil, la conservation d'énergie est un problème important.

Plusieurs travaux se concentrent sur l'optimisation de la consommation d'énergie dans la communication en concevant des protocoles de contrôle d'accès au medium (MAC) et des protocoles de routage spécialement pour ces réseaux. Le principe utilisé dans la plupart de ces travaux est de permettre aux noeuds capteurs de se mettre en veille au lieu de rester en mode d'écoute du canal(Idle) consommant beaucoup d'énergie. Mais, ils n'évitent pas les fréquentes transitions entre les modes "en veille" et "activité" (réception ou transmission) alors qu'elles consomment aussi de l'énergie.

L'objectif de notre travail est d'évaluer cette technique « la mise en veille dans les réseaux de capteurs » en terme de minimisation de la consommation d'énergie des nœuds capteurs dans un réseau ; d'où l'amélioration de l'efficacité d'une application utilisant ce type de réseau. Notre étude à montré en utilisant la simulation, les impacts de cette technique pour l'augmentation de la durée de vie des réseaux de capteurs.

Abstract

Extremely demand in the field of ambient networks, currently allows considering the development of wireless sensor networks. These networks facilitate monitoring and remote control of the physical environment with greater accuracy.

They may have several divers' applications (environmental, military, medical, etc.). Note that a sensor network consists of many units called nodes sensors. Each node is composed primarily of one or more sensors, a processing unit and a communication module (transceiver). These nodes communicate with each other in a fixed or mobile network topology to route information to a control unit (sink) or (base station) situating outside the measuring zone.

The implementation of wireless sensor networks (WSN) poses many problems such as routing data location, and energy consumption constraints, in order to maximize the autonomy of nodes that makes the network. In wireless sensor networks, energy conservation is an important issue.

Several studies focuses on optimizing energy consumption in designing communication protocols medium access control (MAC) and routing protocols specifically for these networks. The principle used in most of this works is to allow sensor nodes to sleep instead of staying in the listening transceiver mode (Idle) avoiding by that any extra consumption. But they do not avoid the frequent transitions between modes "sleeping" and "activity" (reception or transmission) witch also consume energy.

The aim of our work is to show the advantages of this technique "Sleeping In Sensor Networks" on minimizing energy consumption of sensor nodes in a network, hence improving the effectiveness of an application using this network type (WSN). Our study showed using a simulation, the impacts of this technique for increasing the lifetime of sensor networks.

Tables des matières

Chapitre III : *La technique de mise en veille dans les WSN*

Chapitre VI

Tables de figures

Tables

Acronymes

WLAN Wireless Local Area Network

WPAN Wireless Personal Area Networks

MAC Medium Access Control

DARPA Defense Advanced Research Agency

MANET Mobile Ad Hoc Networks

GPS Global Positioning System

CAP Contention Access Period

CFP Contention Free Period

MRAM magnetic Rondom Access Memory

MANET Mobile Ad hoc NETworks

WSN Wireless Sensor Network

RCSF réseaux de capteurs sans fil

MEMS Microelectromechanical Systems

PDR Packet Delivery Ratio

Ah Ampere heur

V volte

DNS Distributed Sensor Network

DARPA Defense Advanced Research Projects Agency

SensIT Sensor Information Technology

WATS Wide Area Tracking System

GDIMP Great Duck Island Monitoring Project

AILISA Assessment Units for Medical Monitoring and assistance technologies in gerontology

WiDeSys Wireless Deterministic System comprend

CAP Contention Access Period

CW Collision Window

Introduction générale

Depuis leur création, les réseaux de communication sans fil ont connu un succès sans cesse croissant au sein des communautés scientifiques et industrielles. Grâce à ses divers avantages, cette technologie a pu s'instaurer comme acteur incontournable dans les architectures réseaux actuelles. Le média hertzien offre en effet des propriétés uniques, qui peuvent être résumées en trois points : la facilité du déploiement, l'ubiquité de l'information et le coût réduit d'installation. Au cours de son évolution, le paradigme sans fil a vu naître diverses architectures dérivées, telles que : les réseaux cellulaires, les réseaux locaux sans fils et autres. Durant cette dernière décennie, les réseaux de capteurs sans fil s'infiltrent pratiquement dans tous les domaines.

Un réseau de capteurs sans fil (RCSF) se compose d'une multitude d'appareils instrumentés, nomades ou éparpillés, effectuant un grand nombre de mesures (température, acoustique, vibrations, pression, suivi de déplacement, détection de polluants). Ces unités autonomes sont dotées d'un microcontrôleur, d'une alimentation (pile ou batterie, le plus souvent), d'un émetteur-récepteur radio et d'un dispositif de mesure et de détection (*Technologie ABB s'appuyant sur un matériel banalisé à bas prix (émetteurs-récepteurs radio 2,4 GHz) et un protocole visant spécifiquement les contraintes temps réel des automatismes de terrain).*

Un réseau de capteurs sans fil (RCSF), ou "Wireless Sensor Network" (WSN), est composé d'un ensemble d'unités de traitements embarquées, appelées "motes", communiquant via des liens sans fil. Le but général d'un WSN est la collecte d'un ensemble de paramètres de l'environnement entourant les motes, telles que la température ou la pression de l'atmosphère, afin de les acheminer vers des points de traitement. Les RCSF sont souvent considérés comme étant les successeurs des réseaux ad hoc. En effet, les RCSF partagent avec les MANET (Mobile Ad hoc NETworks) plusieurs propriétés en commun, telles que l'absence d'infrastructure et les communications sans fil. Mais l'une des différences clés entre les deux architectures est le domaine d'application. Contrairement aux réseaux MANET, qui n'ont pas pu connaître un vrai succès, les RCSF ont su attirer un nombre croissant d'industriels, vu leur réalisme et leur apport concret.

En effet, le besoin d'un suivie continu d'un environnement donné est assez courant dans diverses activités de la société. Les processus industriels, les applications militaires de tracking, le monitoring d'habitat, ainsi que l'agriculture de précision ne sont que quelques exemples d'une panoplie vaste et variée d'applications possibles du suivi continu offert par les RCSF. Grâce à ce potentiel riche en applications, les RCSF on su se démarquer de leur origine MANET et attirer de grandes firmes à travers le monde, telles que IBM, Sun, Intel et Philips. Malheureusement, les RCSF ne sont pas facile à exploiter ou à utiliser ! A cause de leur faible coût et leur déploiement dans des zones parfois hostiles, les nœuds capteurs sont assez fragiles et vulnérables à diverses formes de défaillances : cassure, épuisement d'énergie, ... etc. Ces problèmes rendent les RCSF des systèmes à fragilité innée, qui doit être considérée comme une propriété normale du réseau.

Les enjeux sociétaux et économiques des réseaux de capteurs sans fil sont de grande importance puisqu'ils portent sur l'intégration de « l'intelligence » autour de l'Homme et de son environnement dans de nombreux domaines d'application : la santé, l'environnement, la sécurité (civile ou militaire), l'alimentation, le transport (maritime et terrestre), l'aéronautique et l'espace ou l'habitat. Ainsi, des avancées importantes ont été faites ces dernières années sur les aspects réseaux (sécurisation des données échangées, tolérance aux fautes de transmission, tolérance à la mort d'un ou de plusieurs nœuds, configuration ad hoc) ainsi que sur les technologies d'intégration et d'assemblage du capteur (capteur lui même entouré de l'électronique, de l'énergie et du module de communication).

Les applications typiques de réseau de capteurs sans fil requièrent que chaque nœud du réseau soit de petite taille, endurant vis-à-vis de son environnement (température, vibrations, humidité, agressions chimiques) et d'une durée de vie importante (supérieure à 10 ans). Ainsi**,** *l'alimentation énergétique et la gestion de l'énergie électrique est une question centrale dans la conception de réseaux de capteurs.* La trop faible densité d'énergie des stockages électrochimiques limite la durée de vie des réseaux déployés ou impose une opération de maintenance parfois périlleuse (milieux difficilement accessibles, risque de perturber l'environnement de mesure.

Beaucoup de problèmes restent encore à traiter et beaucoup de solutions embryonnaires proposées dans la recherche demeurent encore à améliorer, telles que les solutions proposées aux problèmes d'optimisation de l'énergie dépensée, de routage des données, d'auto-organisation en cas de

perte de nœuds, d'agrégation et de fusion des données, d'interrogation du réseau et de collecte des données, etc.

Parmi ces problèmes, la minimisation de l'énergie dissipée et la maximisation de la durée de vie du réseau sont sans doute ceux qui ont suscité le plus d'intérêt dans la recherche. En effet, de par leurs applications potentielles et leurs propriétés physiques, les capteurs sont généralement munis d'une source d'énergie très limitée et difficilement renouvelable. La conservation demeure donc cruciale. Cette problématique, très spécifique aux réseaux de capteurs, est au centre de la majorité des travaux rencontrés dans la littérature scientifique. Aussi, quand elle n'est pas l'objectif principal dans un travail de recherche, elle est toujours prise en compte dans tout processus de modélisation et de conception d'outils, d'algorithmes et de solutions relatifs aux réseaux de capteurs. Dans cet ordre d'idées, nous allons nous intéresser dans ce mémoire à un mécanisme d'optimisation de la durée de vie des réseaux de capteurs ***« la technique de mise en veille*** ».

Nous allons en premier lieu faire un état de l'art sur les réseaux de capteurs sans fil; ensuite nous présenterons quelques éléments de problématique, liés principalement à la contrainte énergétique des capteurs. Puis, nous allons visiter les différentes solutions basées sur une gestion optimisée des rôles des capteurs durant leur cycle de vie, afin de maximiser la durée de vie de ce dernier, dont les différents protocoles économes en énergie ; nous allons enchainer par un chapitre sur la mise en veille l'une des techniques utilisé dans les différentes solutions visités dans le chapitre II ; enfin nous allons faire une démonstration « simulation » pour montrer l'efficacité de cette technique en terme de consommation d'énergie.

Sommaire

I.1 Introduction

L'évolution de l'informatique a été marquée par différentes étapes dans la miniaturisation. Dernièrement, sont apparus d'infimes *systèmes micro-électromécaniques*, des dispositifs à bas coûts intégrant les fonctionnalités de captage, de traitement et de communication.

Un grand nombre de ces dispositifs est déployé dans la nature afin de créer un réseau de capteurs à des fins aussi bien de contrôle que de monitorisation. Ces dispositifs formant les noeuds, détectent les changements environnementaux et les signalent aux autres noeuds sur la base d'une architecture flexible du réseau. Les noeuds capteurs donnent la possibilité d'être déployés dans des environnements hostiles et sur de larges zones géographiques.

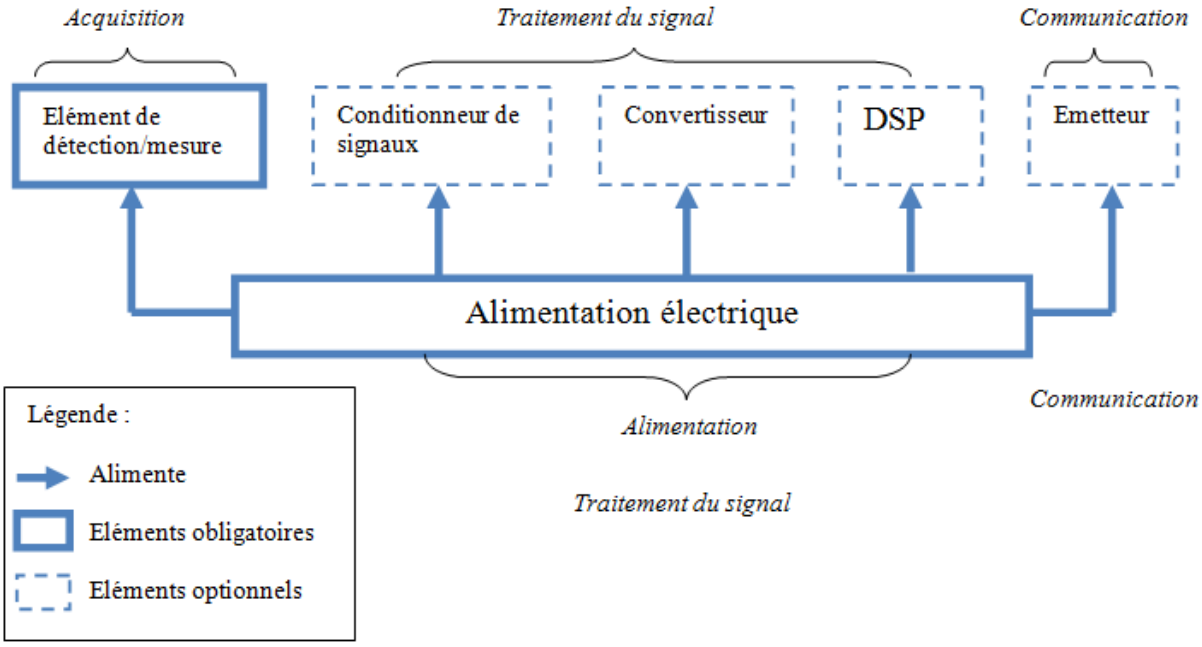
Nous allons dans ce chapitre présenter les réseaux de capteurs sans fil : leurs architectures de communication et leurs applications. Nous allons discuter également les principaux facteurs et contraintes qui influencent la conception des réseaux de capteurs sans fil. Par la suite, nous décrirons la problématique de la consommation d'énergie dans les réseaux de capteurs sans fil. Finalement, nous présenterons une vue globale des solutions proposées dans la littérature pour la gestion de la consommation de l'énergie.

I.2 Les réseaux de capteurs sans fil

Les réseaux de capteurs sans-fil(RCSF), plus connus sous le nom de Wireless Sensor Networks (WSNs) sont une technologie émergente issue des progrès de différents domaines. Ils ont en effet profité de la miniaturisation des composants électroniques, de l'augmentation de leur capacité (puissance de calculs, énergie, capacité de mémoire …), de la diminution des coûts de fabrication mais également des avancées dans les réseaux de communication sans-fil à travers l'essor des téléphones mobiles, des PDAs, etc. l'utilisation des réseaux de capteurs est devenu alors indispensable dans plusieurs domaines et rend la récoltes de données et d'informations plus efficaces et rapide.

I.3 Les capteurs traditionnels

Les réseaux de capteurs sans-fil visent à étendre les domaines d'application des capteurs. En effet, des capteurs sont déjà utilisés pour la création de systèmes d'acquisition de données car ils sont capables de détecter des phénomènes dans un environnement proche, de les quantifier et de transmettre les données ainsi obtenues à d'autres éléments du système en vue de leur traitement. Les éléments mesurés sont des grandeurs physiques telles que la pression, l'humidité, les vibrations, etc. **[2]**



-Figure I-1 : Schématisation d'un capteur « traditionnels »-

Un capteur traditionnel est constitué de différents éléments hardware dont l'architecture est montrée dans la **Figure I-1**.

- Un système d'acquisition de données composé d'un capteur et d'un afficheur digital.
- Un élément de détection qui transforme la grandeur physique en signal analogique.
- Un convertisseur pour digitaliser et afficher la mesurande.
- Une source d'énergie qui peut être soit le secteur, soit des batteries.
- Un module de communication si le destinataire des données (Ex : l'afficheur) est éloigné.
- Un conditionneur de signaux intervient afin d'améliorer la qualité des mesures.
- Un processeur de signal numérique qui permet de traiter le signal c'est-à-dire d'en extraire des données, de les modifier ou de les adapter en vue de la transmission ou du stockage.

I.4 Comparaison entre capteur(WSN) et capteur traditionnel

Les capteurs présents dans les Wireless Sensor Networks se différencient de ceux précédemment décrits par plusieurs points.

1 *Sans-fil*

- Les nœuds capteurs ne sont plus câblés mais communiquent avec les autres éléments via un module de communication sans-fil.
- Les capteurs précédents se contentaient d'envoyer des données à des éléments capables de les traiter.
- les nœuds capteurs peuvent envoyer et recevoir, ils ont été conçus afin de fonctionner en réseau et de servir de relais pour que les données collectées et les autres informations essentielles au maintien du réseau puissent être propagées à travers celui-ci. Cette transmission est rendue possible grâce à un émetteur-récepteur.

2 ***Une diminution des coûts de production et une miniaturisation de la taille des composants***

Bien que ces progrès s'appliquent aux capteurs décrits précédemment, il s'agit dans le cas des réseaux de capteurs d'une condition sine qua non. En effet, les capteurs ne sont désormais pas plus grands qu'une pièce de monnaie ou un bout de doit (voir **Figure I-2**). Tous ces facteurs permettent d'envisager le déploiement d'un nombre beaucoup plus conséquent de capteurs à savoir des centaines voire des milliers.

Cela permet de créer un réseau beaucoup plus dense où la fiabilité de chaque nœud n'est plus primordiale. En effet, en imaginant un déploiement aléatoire, plusieurs nœuds pourraient se retrouver au même endroit et couvrir la même région. Ce qui permettrait à un nœud défaillant d'être remplacé par un autre.

3 ***La source d'énergie***

Dans le cas des nœuds capteurs, il s'agit d'une alimentation embarquée contrairement aux précédents où la source d'énergie pouvait être fournie par une source extérieure.

Cela représente une contrainte importante pour la durée de vie des capteurs car la capacité des batteries est limitée et que, de plus, il n'est pas toujours possible de les remplacer lorsqu'elles sont déchargées. Cela se conçoit facilement si l'on envisage un très grand nombre de capteurs et/ou un environnement « hostile » de déploiement. Enfin, toute opération effectuée par les motes consomme de l'énergie entraînant ainsi une décharge plus ou moins importante des batteries.

4 ***Différence entre la phase d'acquisition des données et la phase de communication***

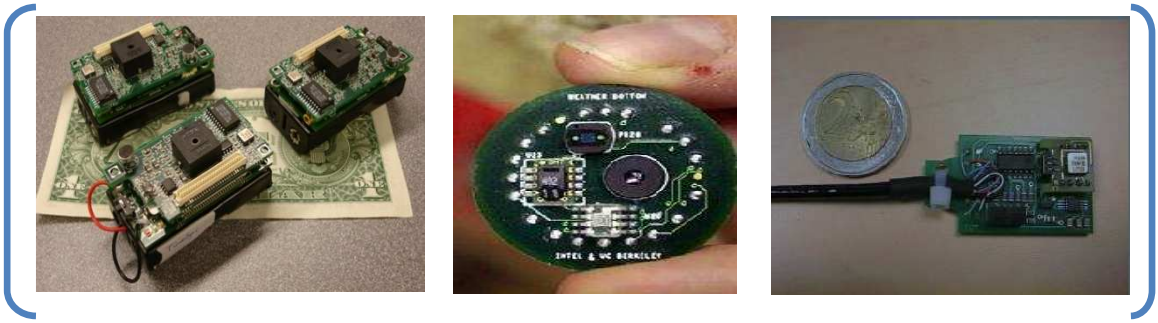
En effet, le principal avantage d'un mote est de disposer en plus d'une unité de traitement permettant non seulement de rapprocher le traitement des données de leur lieu de détection mais

également de les agréger. Cela s'intègre dans le cadre des politiques d'économie d'énergie essentielles aux WSNs.

Cela permet de diminuer la taille des messages à transmettre aux autres éléments du réseau et ainsi de diminuer le temps d'utilisation de l'émetteur-récepteur qui, pour rappel, est l'élément consommant le plus d'énergie dans les motes.

I.5 Architecture d'un nœud-capteur (WSN)

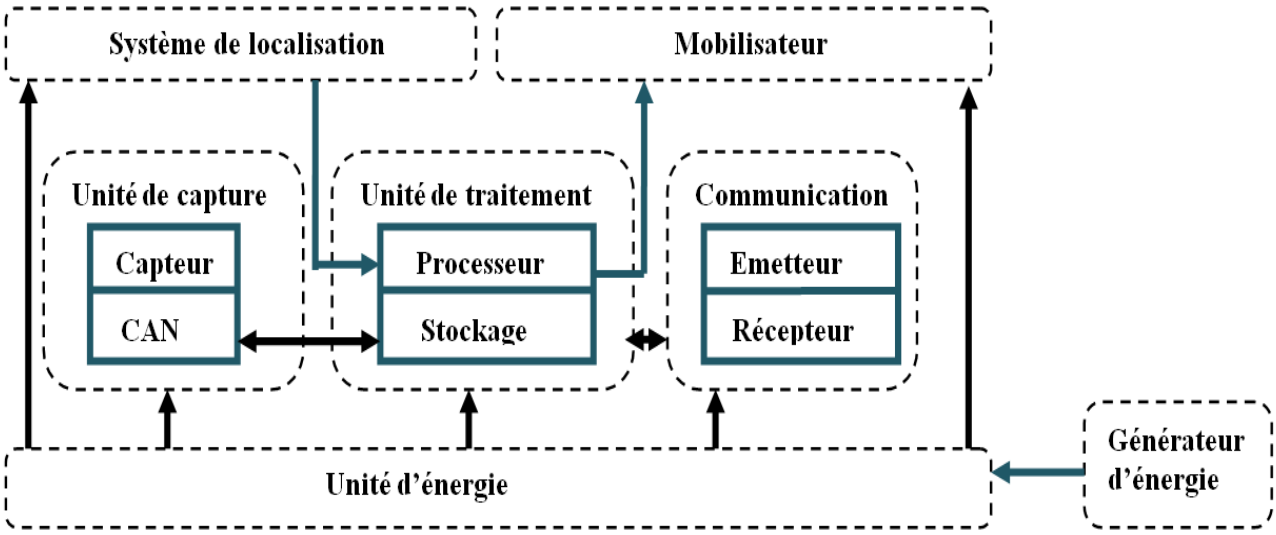
Un nœud-capteur est composé de plusieurs éléments ou modules correspondant chacun à une tâche particulière d'acquisition, de traitement, ou de transmission de données. Il comprend également une source d'énergie [**3**]



-Figure I-2 : Capteurs -

Généralement, un capteur (*« mote »* en anglais) est composé de quatre unités voir **Figure I-3** : une unité de capture (*Sensing unit*), une unité de traitement (*Processing unit*), une unité d'émission/réception (*Transceiver unit*) et une unité d'énergie (*Power unit*). Cependant, un capteur peut être muni de composants additionnels selon son domaine d'application, tels que : un générateur d'énergie, un système de localisation et un mobilisateur.

Parmi les plus célèbres, les *« mote »* **Figure I-2** MICAx et TelosBe de Crossbow.



- Figure I-3 : Composants d'un capteur [5] -

I.5-1 *L'unité d'acquisition des données*

Le principe de fonctionnement des détecteurs est souvent le même : il s'agit de répondre à une variation des conditions d'environnement par une variation de certaines caractéristiques électriques Les variations de tension sont ensuite converties par un convertisseur Analogique-Numérique afin de pouvoir être traitées par l'unité de traitement. On trouve aussi des structures plus complexes pour détecter d'autres phénomènes : les MEMS (pour Microelectromechanical systems) [**3**].

I.5-2 *L'unité de transmission de données*

Les composants utilisés pour réaliser la transmission sont des composants classiques. Ainsi on retrouve les mêmes problèmes que dans tous les réseaux sans-fil : la quantité d'énergie nécessaire à la transmission augmente avec la distance. Pour les réseaux sans-fil classiques (LAN, GSM) la consommation d'énergie est de l'ordre de plusieurs centaines de milliwatts, et on se repose sur une infrastructure alors que pour les réseaux de capteurs, le système de transmission consomme environ 20 mW et possède une portée de quelques dizaines de mètres. Pour augmenter ces distances tout en préservant l'énergie, le réseau utilise un routage multi-sauts **[3**].

Cette unité *(tranceiver)* gère la connexion d'un nœud au réseau. Typiquement, le module radio d'un nœud capteur a quatre modes opérationnels :

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

- ***Receive* :** le module radio est en train de recevoir un paquet.
- ***Sleep* :** le module radio est éteint.

A noter que ces modes seront expliquées ultérieurement.

L'unité d'émission/réception (*tranceiver*) communique selon trois modes : des lasers, des infrarouges ou des radiofréquences, de type optique (comme dans les nœuds *Smart Dust*). Les communications de type optique sont robustes vis-à-vis des interférences électriques. Néanmoins, elles présentent l'inconvénient d'exiger une ligne de vue permanente entre les entités communicantes. Par conséquent, elles ne peuvent pas établir de liaisons à travers des obstacles. Les unités de transmission de type radio-fréquence comprennent des circuits de modulation, démodulation, filtrage et multiplexage ; ce qui implique une augmentation de la complexité et du coût de production du micro-capteur. Concevoir des unités de transmission de type radio-fréquence avec une faible consommation d'énergie est un véritable défi. En effet, pour qu'un nœud ait une portée de communication suffisamment grande, il est nécessaire d'utiliser un signal assez puissant. Cependant, l'énergie consommée serait importante. L'autre alternative serait d'utiliser de longues antennes, mais ceci n'est pas possible à cause de la taille réduite des micro-capteurs **[6].**

I.5-3 L'unité de traitement des données

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

- le microcontrôleur commande aussi les autres unités comme le système de transmission.
- pour augmenter la taille de la mémoire on utilise un autre type de mémoires (*mémoire flash, Eprom*).

Un autre type de mémoire supplémentaire pourrait bientôt être envisagé : il s'agit de la MRAM. En effet, celle-ci est en cours de développement mais ses atouts pour une utilisation dans les réseaux de capteurs pourraient être multiples car elle est très rapide à l'écriture et à la lecture, non volatile, elle consomme peu d'énergie, peut être effacée et réécrite un nombre de fois illimité, son coût de fabrication est faible et elle ne subit pas les influences des radiations. **[2**]

I.5-4 L'unité d'énergie

Les capteurs sont généralement munis d'une ressource énergétique à durée de vie limitée. Toute intervention pour recharger ou changer les batteries est la plupart du temps exclue. 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 de réduire la consommation d'énergie en agissant sur les modules inactifs. [**5**]

Le mode de communication inter-capteurs pourrait être optique (e.g., infrarouge), mais la communication radio, généralement dans la bande UHF, reste la plus pratique car elle supporte les obstacles physiques. Des efforts de standardisation de la couche physique et de la couche d'accès au canal des RCSF ont donné naissance à la norme *IEEE 802.15.4* qui établit des bandes de fréquences et les types de modulation utilisables par les RCSF : 2.4 GHz comme fréquence globale, avec une modulation ***O-QBPSK***, 915 MHz avec une modulation ***BPSK*** pour l'Amérique du nord et 868 MHz avec une modulation ***BPSK*** pour l'Europe. Elle définit aussi l'accès au canal pour les topologies en étoile, en maille et hiérarchiques**. [8]**

Notons que la batterie d'un capteur peut être rechargée en utilisant l'énergie solaire ou la vibration mais la gestion et la préservation de la batterie reste indispensable car l'alimentation est une composante cruciale.

À titre indicatif, ce sera souvent une pile AA normale d'environ *2,2 - 2,5 Ah* fonctionnant à *1,5 V*

D'autres composants peuvent être ajoutés à un nœud capteur, comme un système de localisation (GPS), un composant de mobilité pour le rendre mobile, un générateur d'énergie, etc.

Il existe dans le monde plusieurs fabricants de capteurs. Nous citerons ***Crossbow***, ***Cisco***, ***Dalsa***, ***EuroTherm***, et ***Sens2B***. Parmi ces capteurs, il existe quelques uns qui sont capables de varier la puissance du signal émis afin d'élargir/réduire le rayon de communication et en conséquence la zone de communication.[**24**]

I.6 Architecture du réseau WSN

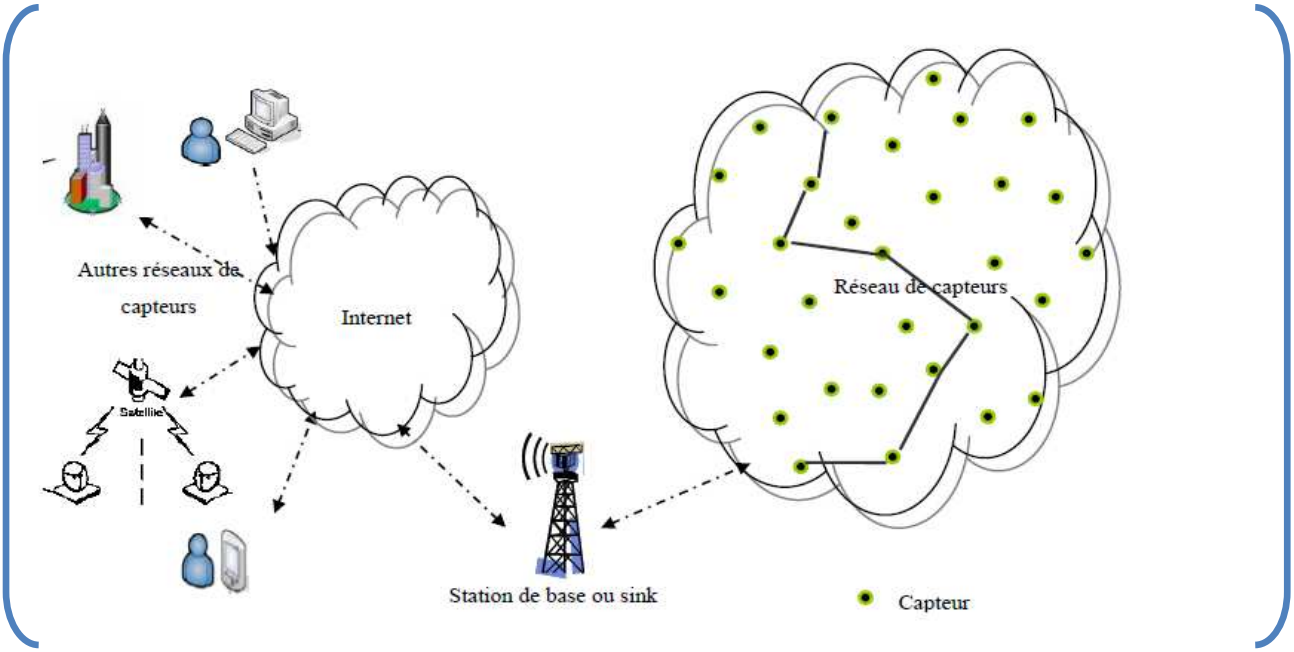


Figure I-4 : Exemple d'un réseau de capteurs sans fil [9]

I.6-1 Description

Les réseaux de capteurs forment une nouvelle génération de réseaux aux propriétés spécifiques, qui n'entrent pas dans le cadre des architectures classiques.

Un RCSF se compose de centaines ou de milliers de nœuds capteurs dispersés dans une zone géographique appelée champ de captage, afin d'étudier et surveiller l'environnement où il a été déployé et collecter des informations pour réaliser des traitements et des actions particulières *(selon le domaine d'étude).*

Les capteurs sont chargés de collecter périodiquement des données et de les envoyer vers un nœud particulier appelé nœud puits *(Sink ou station de base).* Ce dernier analyse ces données et transmet à son tour l'information collectée à l'utilisateur via internet ou satellite **figure I-4**. Le nœud puits est responsable, en plus de la collecte des rapports, de la diffusion des demandes sur les types de données requises par les capteurs via des messages de requêtes.

I.6-2 Mise en place d'un réseau de capteurs

Mettre en œuvre un réseau de capteurs sous entend le problème de communication et de sources d'énergie. Pour rassembler les informations requises et réussir à les émettre à bon escient, le choix de la qualité des capteurs est primordial. De plus, l'utilisation de ces capteurs doit soumettre des protocoles pour améliorer leurs durées de vie, d'où le processeur du capteur ne doit pas être utilisé trop intensivement pour consommer moins d'énergie possible. Il doit donc incorporer des éléments réactifs plutôt que cognitifs.

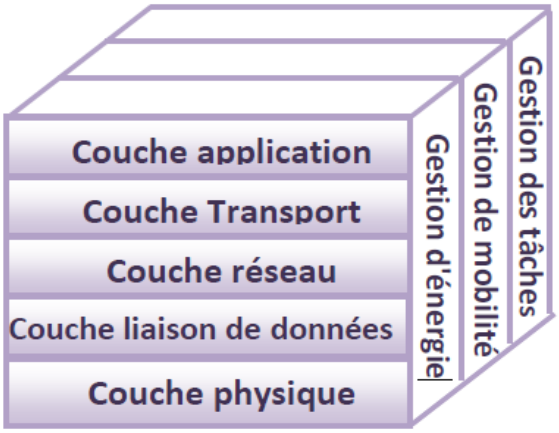
Afin d'assurer un bon débit, la portée des émetteurs-récepteurs doit être faible de l'ordre d'une dizaine de mètres. Donc le problème de routage de contrôle des erreurs et de gestion de l'alimentation s'imposent dans la mise en place d'un RCSF.

Du point de vue de la communication, l'environnement des protocoles IP est trop lourd et engendre un débit trop important et une surconsommation. Les solutions qui ont été dérivées des réseaux de terrain, ou réseaux industriels temps réel, présentent un meilleur compromis entre efficacité et énergie consommée. Comme les capteurs peuvent être diffusés par centaine au mètre carrée, l'adressage IPv6 semble le plus probable. Ce qui est intéressant c'est d'utiliser un environnement de paquet IP encapsulé dans des trames spécifique à déterminer.

Pour le moment, les problèmes de sécurités et de qualité de service sont mis au second plan par rapport aux problèmes de consommation. Un champ de recherche important est en tout cas ouvert pour rendre des réseaux de capteurs efficaces et résistants.

I.7 Pile protocolaire

La pile protocolaire **Figure I-4** est utilisée par la station de base ainsi que tous les autres capteurs du réseau. La pile protocolaire comprend la couche application, la couche transport, la couche réseau, la couche liaison de données, la couche physique, le plan de gestion de l'énergie, le plan de gestion de la mobilité et le plan de gestion des tâches.



- **Figure I-5: Pile protocolaire. [11]-**

I.7-1 La couche physique

Spécifie les caractéristiques matérielles, les fréquences porteuses, assure la transmission et la réception des données au niveau bit. En outre, les plans de gestion de l'énergie, de la mobilité et des tâches surveillent la puissance, le mouvement et la distribution des tâches, respectivement, entre les nœuds capteurs.

I.7-2 La couche liaison

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

La couche réseau prend soin de router les données fournies par la couche transport.

I.7-3 La couche réseau

Dans la couche réseau le but principal est de trouver une route et une transmission fiable des données, captées, des nœuds capteurs vers le puits "sink" en optimisant l'utilisation de l'énergie des capteurs. Ce routage diffère de celui des réseaux de transmission ad hoc sans fils par les caractéristiques suivantes:

- il n'est pas possible d'établir un système d'adressage global pour le grand nombre de nœuds.

- les applications des réseaux de capteurs exigent l'écoulement des données mesurées de sources multiples à un puits particulier.
- les multiples capteurs peuvent produire de mêmes données à proximité d'un phénomène (redondance).
- les capteurs capteur exigent ainsi une gestion soigneuse des ressources.

En raison de ces différences, plusieurs nouveaux algorithmes ont été proposés pour le problème de routage dans les réseaux de capteurs

I.7-4 La couche transport

Cette couche est chargée du transport des données, de leur découpage en paquets, du contrôle de flux, de la conservation de l'ordre des paquets et de la gestion des éventuelles erreurs de transmission. Elle aide à gérer le flux de données si le réseau de capteurs l'exige. Elle permet de diviser les données issues de la couche application en segments pour les délivrer, ainsi elle réordonne et rassemble les segments venus de la couche réseau avant de les envoyer à la couche application

I.7-5 La couche application

Cette couche assure l'interface avec les applications. Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par les logiciels. Suivant la fonctionnalité des capteurs, différentes applications peuvent être utilisées et bâties sur la couche application.

I.7-6 Plans de gestion

Les plans de gestion d'énergie, de mobilité et de tâche contrôlent l'énergie, le mouvement et la distribution de tâche au sein d'un nœud capteur. Ces plans aident les capteurs à coordonner la tâche de captage et minimiser la consommation d'énergie. Ils sont donc nécessaires pour que les capteurs puissent collaborer ensemble, acheminer les données dans un réseau mobile et partager les ressources entre eux en utilisant efficacement l'énergie disponible. Ainsi, le réseau peut prolonger sa durée de vie.

I.7-6 a) plan de gestion d'énergie

Un nœud de capteur sans fil, nécessite seulement une source d'énergie limitée (< 0.5 Ah, 1.2 V). La vie du nœud montre, une dépendance forte à l'égard de la vie de la batterie.

Cette partie gère la manière dont les nœuds utilisent leurs énergies. Par exemple le nœud doit se mettre en sommeil après la réception d'un message à partir d'un voisin. Il doit aussi informer ses

voisins de son incapacité à participer dans le routage quand sa réserve d'énergie atteint un seuil particulier.

I.7-9 b) Plan de gestion de mobilité

Détecte et enregistre le mouvement du capteur. Ainsi, un retour arrière vers l'utilisateur est toujours maintenu et le capteur peut garder trace de ses capteurs voisins. En déterminant leurs voisins, les capteurs peuvent balancer l'utilisation de leur énergie et la réalisation de tâche ;

I.7-10 c) Plan de gestion de tâche

Balance et ordonnance les différentes tâches de captage de données dans une région spécifique. Il n'est pas nécessaire que tous les capteurs de cette région effectuent la tâche de captage au même temps ; certains capteurs exécutent cette tâche plus que d'autres selon leur niveau de batterie.

I.8 Facteurs des WSN

Les noeuds capteurs ayant un petit volume (le plus petit et de quelques millimètres cube), sont limités dans la quantité d'énergie qu'ils peuvent stocker. En outre, ces noeuds sont sensibles à l'échec, suite à l'épuisement des batteries ou bien aux influences de l'environnement. Dans cette section, nous allons montrer les différentes caractéristiques liées aux réseaux de capteurs telles que le déploiement, la couverture, la connectivité, l'énergie, la mobilité, etc. [**12**]

La conception et la réalisation des réseaux de capteurs sans fil sont influencées par plusieurs paramètres. Ces facteurs servent comme directives pour le développement des algorithmes et protocoles utilisés dans les RCSF.

I.8-1 Durée de vie du réseau

C'est l'intervalle de temps séparant le moment de déploiement du réseau de l'instant d'expiration du premier nœud (*l'énergie du nœud s'épuise*). La durée de vie d'un réseau de capteurs diffère d'un type à un autre selon le domaine d'utilisation (*de quelques heures à plusieurs années*).

I.8-2 Ressources limitées

En plus de l'énergie, les nœuds capteurs ont aussi une capacité de traitement et de mémoire limitée. En effet, les industriels veulent mettre en œuvre des capteurs simples, petits et peu coûteux qui peuvent être achetés en masse. De plus, les capteurs sont caractérisés par une bande passante limitée. Typiquement, le débit utilisé est de quelques dizaines de Kb/s. Le but de ce faible débit est de

minimiser l'énergie consommée lors de transfert de données entre les nœuds. Un débit de transmission réduit n'est pas handicapant pour un réseau de capteurs où les fréquences de transmission ne sont pas importantes.

I.8-3 Facteur d'échelle

Le nombre de nœuds déployés pour une application peut atteindre des milliers. Dans ce cas, le réseau doit fonctionner avec des densités de capteurs très grandes. Un nombre aussi important de nœuds engendre beaucoup de transmissions inter-nodales et nécessite que la station de base soit équipée de mémoire suffisante pour stocker les informations reçues.

I.8-4 Topologie dynamique

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

- Les nœuds capteurs peuvent être déployés dans des environnements hostiles (champ de bataille par exemple), la défaillance d'un nœud capteur est, donc très probable.
- Un nœud capteur peut devenir non opérationnel à cause de l'expiration de son énergie.
- Dans certaines applications, les nœuds capteurs et les stations de base sont mobiles.

I.8-5 Agrégation de données

Dans les RCSF, les données produites par les nœuds capteurs voisins sont très corrélées spatialement et temporellement. Ceci peut engendrer la réception par la station de base d'informations redondantes. Réduire la quantité d'informations redondantes transmises par les capteurs permet de réduire la consommation d'énergie dans le réseau et ainsi d'améliorer sa durée de vie. L'une des techniques utilisée pour réduire la transmission d'informations redondantes est l'agrégation des données. Avec cette technique, les nœuds intermédiaires agrègent l'information reçue de plusieurs sources. Cette technique est connue aussi sous le nom de fusion de données [**11**].

I-9 Contraintes d'un réseau de capteurs

Les réseaux sans fils et sans infrastructure, comme les réseaux ad hoc et les réseaux de capteurs, sont limité dans leurs performances et leurs interactions avec l'environnement.

Ces réseaux posent plusieurs défis à relever à cause des ces facteurs :

- ***Manque d'infrastructure* :** par exemple jeter les nœuds d'un avion dans une forêt ;

- ***Non intervention humaine* :** les nœuds doivent se reconfigurer tous seuls en cas d'instabilité de fonctionnement ;
- ***Absence de ressource d'énergie :*** les nœuds n'ont que leurs batteries initiales et donc il faudra une consommation optimale pour les communications et les traitements ;
- ***Changements dynamiques :*** les nœuds doivent s'adapter très vite aux changements imprévus comme l'ajout d'un nœud dans le réseau.

Plusieurs recherches ont été lancées pour résoudre ces problèmes. Notre inspection concentrera spécialement sur le domaine de la maîtrise de la consommation énergétique d'un nœud.

La consommation d'énergie représente un facteur déterministe dans la vie d'un réseau de capteurs parce que les nœuds disposent en générale des ressources limitées en énergies et il est impossible de remplacer ou bien de recharger un nœud en fin de vie. Ceci rend l'optimisation de l'énergie beaucoup plus compliqué parce que il ne s'agit pas seulement de la réduction de la consommation mais aussi de la prolongation de la vie d'un réseau [**10**].

La consommation d'un nœud s'effectue dans cet ordre décroissant :

- ***radio*** *(communication) ;*
- ***protocole*** *(MAC, routage) ;*
- ***CPU*** *(calculs, agrégation) ;*
- ***Capteur*** *(acquisition) ;*

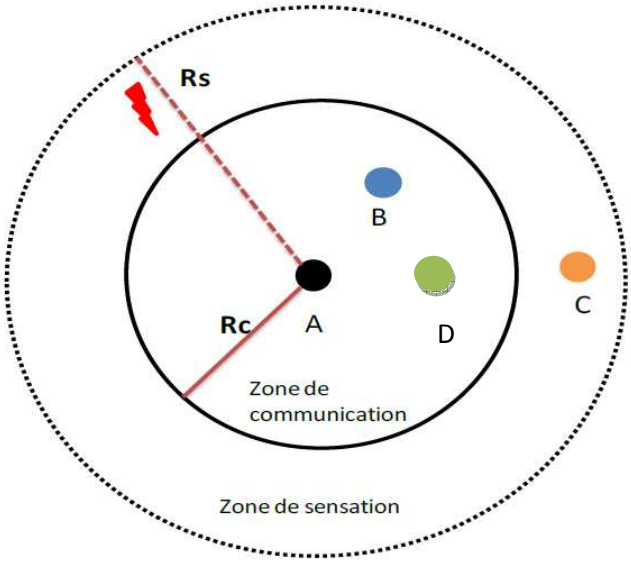
I.9-1 L'hétérogénéité des nœuds

Généralement, les nœuds d'un RCSF sont homogènes et disposent des mêmes capacités de calcul, de stockage et de ressources énergétiques. Cependant, selon les besoins des applications, les nœuds peuvent avoir des rôles différents. Ainsi, selon la tâche assignée au capteur, les besoins en ressources de calcul, de stockage, de communication et d'énergie peuvent varier d'un nœud à un autre.

Pour remédier à ce problème, une solution envisagée par certaines applications consiste à intégrer des nœuds spéciaux plus puissants que les autres et qui seront chargés d'effectuer les tâches les plus coûteuses en termes de ressources énergétiques. Cependant, l'intégration d'un ensemble de nœuds hétérogènes dans un seul réseau impose de nouvelles contraintes liées au routage de données. Par conséquent, la conception des protocoles de routage doit prendre en compte les différents types de nœuds, et les contraintes qui en résultent [**3**].

I.9-2 Zone de couverture

Les capteurs fonctionnent avec un modèle à seuil. Un capteur possède deux zones : une zone de perception ***(RS)*** et une zone de communication ***(RC).*** Pour schématiser, on considère que ces zones sont représentées par deux cercles qui ont pour centre le capteur **Figure I-6** :



- Figure I-6 : Rayon de communication et de sensation [24] -

En influant sur le rapport entre le rayon du ***RS*** et le rayon du ***RC***, on va modifier les contraintes. Ainsi, on va pouvoir minimiser le nombre de nœuds actifs et maximiser la durée de vie du réseau.

Les zones ***RC*** et ***RS*** représentent la zone de couverture d'un capteur. Pour qu'une zone soit complètement couverte, il faut que la densité de capteurs soit suffisante. Comme les capteurs sont généralement disposés sur la zone à couvrir de façon aléatoire, il est nécessaire de disposer d'une densité importante de capteurs. Si la densité de capteurs est trop importante et que la zone que l'on veut surveiller est "trop" couverte, alors des capteurs vont fonctionner inutilement.

Dans la **figure I-6** les nœuds A, B, D peuvent tous communiquer avec C, mais puisqu'ils appartiennent tous les trois à une seule zone de captage il se peut que la donnée captée par chacun soit la même, dans ce cas l'un d'eux peut se mettre en veille.

Les capteurs sont chargés de relever et de router les informations relevées sur la zone couverte vers le point de collecte, également appelé puits. Le puits récupère les informations remontées par les différents capteurs et les transmet au centre de traitement. Les capteurs disposés de manière aléatoire forment la zone de couverture.

I.9-3 Collecter les informations

Il y a deux méthodes pour collecter les informations d'un réseau de capteurs.

I.10-3 A) A la demande

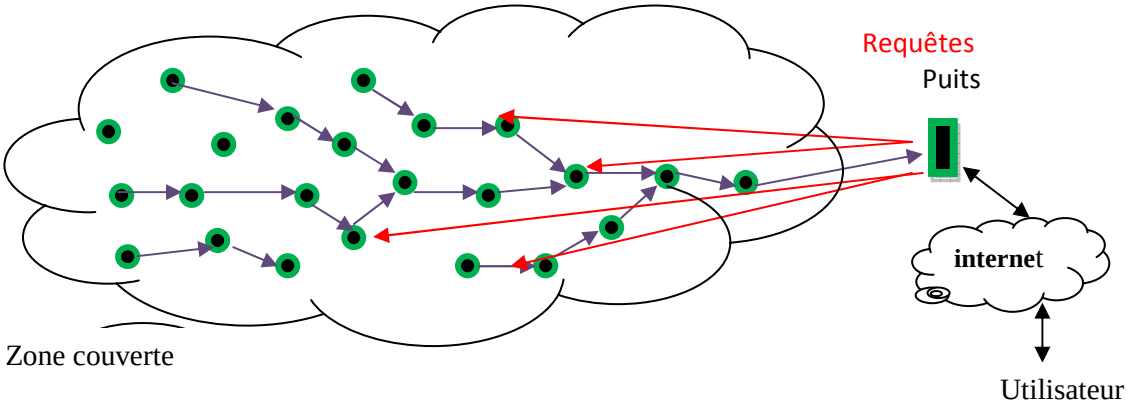
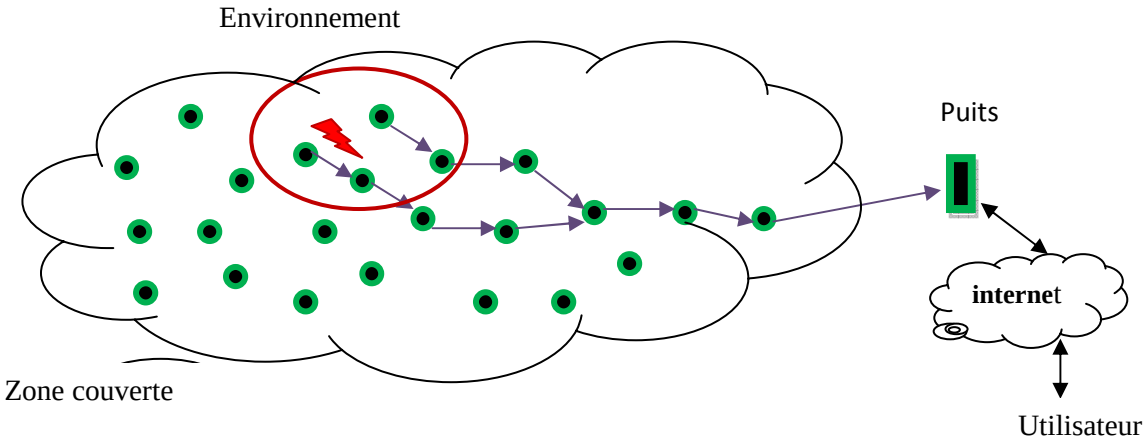


Figure I-7: collecte d'informations à la demande-

Lorsque l'on souhaite avoir l'état de la zone de couverture à un moment T, le puits émet des brodcasts vers toute la zone pour que les capteurs remontent leur dernier relevé vers le puits, comme illustré dans la **figure I-7**. Les informations sont alors acheminées par le biais d'une communication multi-sauts.

I.10-3 B) Suite à un événement



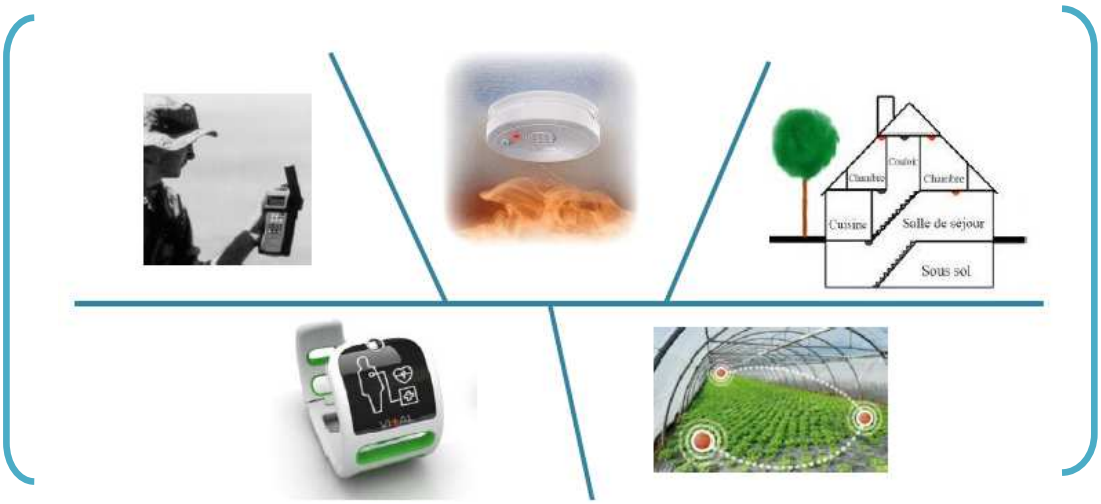
- Figure I-8: Collecte d'informations suite à des événements-

Un événement se produit en un point de la zone de couverture (changement brusque de température, mouvement...), les capteurs situés à proximité remontent alors les informations relevées et les acheminent jusqu'au puits **Figure I-8**.

I.11 Domaines d'applications existants

Les réseaux de capteurs sans fil ont été classés parmi les 21 technologies les plus importants du 21 ième siècle. En effet, la recherche dans le domaine des capteurs est en train de vivre une révolution importante, ouvrant des perspectives d'impacts significatifs dans de nombreux domaines figure I-9 (militaire, environnemental, domotique, médecine, agriculture). D'ou, la classification des applications des RCSF [**11**] :

- Orientées temps(*time driven*).
- Orientées événements(*event driven*)
- Orientées requêtes(*query driven*)
- Hybrides



- Figure I-9 : Domaines d'application des WSN-

I.11-1 *Application orientées temps*

Dans cette catégorie l'acquisition et la transmission des données capturées sont liées au temps (instant précis, période d'acquisition), cette dernière varie de quelques secondes jusqu'à quelques heures où même des jours.

Ainsi, la quantité de données échangées dans le réseau dépend de la périodicité des mesures à effectuer sur l'environnement local.

I.11-2 *Application orientées événements*

Dans ce cas, les capteurs envoient leurs données seulement si un événement spécifique se produit. On peut citer l'exemple de surveillance des feux dans les forêts où un capteur envoie des alarmes à la station de base dès que la température dépasse un certain seuil. Au départ, cette classe d'application était conçue à des fins militaires, comme la surveillance du déplacement d'objets dans le champ de bataille. Par la suite, cette classe a rapidement trouvé de nouvelles perspectives comme le contrôle industriel, le contrôle médical des patients, la surveillance d'édifices

I.11-3 *Applications orientées requêtes*

Dans ce cas, un capteur envoie de l'information uniquement suite à une demande explicite de la station de base. Cette classe d'application est destinée aux applications adaptées à l'utilisateur. Ce dernier peut requérir des informations à partir de certaines régions dans le réseau ou interroger les capteurs pour acquérir des mesures d'intérêts. Dans ce cas, des connaissances sur la topologie du réseau et l'emplacement des capteurs sont nécessaires.

I.11-4 *Applications hybrides*

Ce type d'application met en œuvre les trois modes de fonctionnement décrits précédemment. Par exemple, dans un réseau conçu pour le suivi d'objets, le réseau peut combiner entre un réseau de surveillance (*time driven*) et un réseau de collecte de données par événements (*event driven*). Par exemple, pendant les longues périodes d'inactivité des capteurs et lorsque aucun objet n'est présent, le réseau peut assurer une fonction de surveillance.

I.12 Exemples dans les domaines de développement

I.12-1 Applications militaires

-Figure I-10 : Tracé du chemin d'un véhicule militaire [12] -

Ces applications son faites pour la surveillance des champs de bataille afin de fournir des renseignements concernant l'emplacement, le nombre, le mouvement, et l'identité des soldats et des véhicules, ou bien encore pour la détection des agents chimiques, biologiques et nucléaires, exemple illustré dans la **figure I-10**.

- Le projet DSN (*Distributed Sensor Network*) au DARPA (Defense Advanced Research Projects Agency) était l'un des premiers projets dans les années 80 ayant utilisés les réseaux de capteurs pour rassembler des données distribuées **[3]**.

- Une grande partie de la croissance rapide dans la recherche et le développement des réseaux de capteurs sans fil a été apportée par des programmes financés par l'Agence américaine pour les Projets de Recherche Avancée de Défense (*DARPA pour Defense Advanced Research Projects Agency*), notamment grâce à un programme connu sous le nom de « SensIT » (*Sensor Information Technology*) de 1999 à 2002.

- Lors de la guerre froide les Américains ont placé dans l'océan un système appelé SOSUS composé de capteurs acoustiques pour surveiller les sous-marins silencieux russes.

- Les chercheurs du laboratoire national Lawrence Livermore ont mis en place le réseau WATS (Wide Area Tracking System). Ce réseau est composé de détecteurs des rayons gamma et des neutrons pour détecter et dépister les dispositifs nucléaires. Il est capable d'effectuer la surveillance constante d'une zone d'intérêt. Il utilise des techniques d'agrégation de données pour les rapporter à un centre intelligent.

- JBREWS *(Joint Biological Remote Early Warning System)* est un réseau WSN pour avertir les troupes dans le champ de bataille des attaques biologiques possibles. Un réseau de capteurs peut être déployé dans un endroit stratégique ou hostile, afin de surveiller les mouvements des forces ennemies, ou analyser le terrain avant d'y envoyer des troupes (détection des armes chimiques, biologiques ou radiations). L'armée américaine a réalisé des tests dans le désert de Californie. [**15**]

- Les réseaux développés pour des raisons militaires peuvent également être utilisés pour surveiller des habitations et contribuer au confort domestique, en transformant les logements en

environnements intelligents dont les paramètres (température, pression, humidité, luminosité, etc.) s'adaptent automatiquement au comportement des individus [**16**].

I.12-2 Applications environnementales

Ces réseaux ont été conçus en générale pour la collecte périodique des données environnementales puis leurs transmissions vers une station de base. Les nœuds du réseau peuvent avoir plusieurs fonctionnalités et différents types de capteurs. Ce type de réseau nécessite généralement un flux de données faibles, une durée de vie importante et une topologie statique et par contre elle ne présente pas de contraintes de types temps de latence.

- Un exemple de tel réseau est GDIMP dans lequel les nœuds mesurent la température, l'humidité et les radiations solaires au sein d'un champ agriculture. [**10**]

- Dans le domaine de l'agriculture, les capteurs peuvent être utilisés pour réagir convenablement aux changements climatiques par exemple le processus d'irrigation lors de la détection de zones sèches dans un champ agricole. Cette expérimentation a été réalisée par *Intel Research Laboratory and Agriculture and Agri-Food Canada* sur une vigne à *British Columbia*. [**15**]

- Les thermo-capteurs peuvent également signaler des fuites de produits *toxiques (gaz, produits chimiques, éléments radioactifs, pétrole et autres)* grâce à leur utilisation sur des sites industriels, dans les centrales nucléaires ou dans les pétroliers [**25].**

- [**13**] les auteurs décrivent leurs efforts de mise en œuvre d'un réseau de capteurs sur le volcan « Reventator » dans la partie occidentale de l'Amazone en Équateur.

- Un autre exemple concernant les applications environnementales est le projet ARGO [**20**]. Le but de ce projet est de surveiller l'eau de l'océan, sa température, sa salinité ainsi que sa vélocité. Le projet utilise des noeuds équipés par des capteurs de température et de salinité. Les noeuds sont déployés à partir de navires ou bien d'avions. Ils s'enfoncent à une profondeur de 2000 mètres tous les dix jours. Les données recueillies au cours des déplacements sont transmises à un satellite tandis que des noeuds sont toujours à la surface. La durée de vie des noeuds est d'environ 45 ans.

I.12-3 Supervision de structures et phénomènes sismiques

Ces structures peuvent être des bâtiments, des ponts et des routes, voire des aéronefs. À l'heure actuelle, la sûreté de ces structures est principalement apportée par le biais d'inspections manuelles ou visuelles ou occasionnellement par des technologies onéreuses en temps et en argent, telles que les rayons X et les ultrasons. Des techniques de détection réseau permettent d'automatiser le processus, en fournissant en temps opportun de riches informations sur un début de fissure ou d'autres dommages structuraux. [**3**]

I.12-4 Dans l'industrie et le commerce

Dans une usine de traitement chimique à plusieurs étapes, il peut y avoir des capteurs placés en différents points dans le processus afin de surveiller la température, la concentration chimique, la pression, etc.

Le principal avantage de la création des réseaux de capteurs sans fil dans ces milieux est qu'ils peuvent améliorer de manière significative à la fois le coût et la souplesse inhérente à l'installation, mais encore l'entretien et la modernisation des systèmes filaires. Il convient de noter qu'il existe déjà plusieurs sociétés de développement et de commercialisation de ces produits à l'image des technologies standards telles que *le standard IEEE 802.15.4*, et de la collaboration industrielle telle que *l'Alliance Zigbee*.

- le détecteur de gaz Cobra détecte pratiquement toutes les sortes de gaz utilisés actuellement lors d'un cambriolage. Il peut ainsi alerter lors d'une tentative de vol, mais aussi d'une fuite au niveau de votre installation de gaz avant que le seuil critique de l'explosion soit atteint.

I.12-5 Applications médicales

- AILISA:*Assessment Units for Medical Monitoring and assistance technologies in gerontology* a pour objectif de mettre en place des plateformes pérennes pour l'évaluation de technologies de télésurveillance médicale et d'assistance en gérontologie. Il a démarré au début 2004 grâce à un financement du RNTS dans le cadre de l'Institut de la Longévité [**18**].

- Dans le domaine de la médecine, les réseaux de capteurs peuvent être utilisés pour assurer une surveillance permanente des organes vitaux de l'être humain grâce à des micro-capteurs qui pourront être avalés ou implantés sous la peau (surveillance de la glycémie, détection de cancers, ..). Ils peuvent aussi faciliter le diagnostic de quelques maladies en effectuant des

mesures physiologiques telles que : la tension artérielle, battements du cœur, ... à l'aide des capteurs ayant chacun une tâche bien particulière. Les données physiologiques collectées par les capteurs peuvent être stockées pendant une longue durée pour le suivi d'un patient. D'autre part, ces réseaux peuvent détecter des comportements anormaux (chute d'un lit, choc, cri, ...) chez les personnes dépendantes (handicapées ou âgées).[**15**]

- Dans [**26**], les auteurs ont réalisé un réseau de capteurs pour la surveillance des « signes vitaux ». Le système comprend quatre composantes : un identifiant formé d'un noeud capteur attaché au patient qui contient toutes les données qui le concernent (par exemple son nom), un capteur (par exemple un électrocardiogramme), un dispositif d'affichage qui examine les données du patient et ses signes vitaux, et enfin un stylo conservé par le personnel responsable (médecin), qui permet d'établir l'association entre les différents dispositifs ; ce stylo émet par infrarouge, un identifiant unique pour chaque patient, et peut recevoir les informations et les identificateurs du réseau humain.

- Un autre exemple d'application médicale présenté dans [**27**], est illustré dans **la figure I-11.** Les auteurs présentent un système en temps réel pour suivre les patients. Ce système intègre des capteurs de signes vitaux, la position des capteurs, et un réseau ad-hoc pour permettre la surveillance à distance du patient. Cela facilitera la communication entre les pompiers sur scène (en cas de catastrophe), et les spécialistes dans les hôpitaux locaux qui seront disponibles pour la consultation à distance.

-Figure I-11 : Le flux d'information d'un patient [12]-

I.12-6 La domotique et aéronautique

- Dans l'article **[19]** le système proposé pour la vérification de l'état des équipements critiques dans un avion, s'appelle WiDeSys, l'utilisation de couches physiques (PHY) pouvant fonctionner à des débits entre 1Mb/s et 1Gb/s selon les environnements.

- Le Tracker aimanté est un dispositif GPS personnel de tracking et de secours, autonome et de très petite taille qui fonctionne en mode Sms et c'est la seule balise à donner sa position en claire (adresse postale) directement par SMS. Le Tracker aimanté est extrêmement facile à utiliser, fiable d'une autonomie exceptionnelle de plusieurs jours selon l'utilisation **[20]**

Avec le développement technologique, les capteurs peuvent être embarqués dans des appareils, tels que les aspirateurs, les fours à micro-ondes, les réfrigérateurs, les magnétoscopes, ...

Ces capteurs embarqués peuvent interagir entre eux et avec un réseau externe via Internet pour permettre à un utilisateur de contrôler les appareils domestiques localement ou à distance.

Le déploiement des capteurs de mouvement et de température dans les futures maisons dites intelligentes permet d'automatiser plusieurs opérations domestiques telles que : la lumière s'éteint et la musique se met en état d'arrêt quand la chambre est vide, la climatisation et le chauffage s'ajustent selon les points multiples de mesure, le déclenchement d'une alarme par le capteur anti-intrusion quand un intrus veut accéder à la maison.

II.13 Les protocoles de routage

II.13-1 Introduction

Le routage est une méthode d'acheminement des informations vers une destination donnée dans un réseau de connexion. Comme nous l'avons déjà vu, l'architecture des réseaux Ad-hoc est caractérisée par l'absence d'infrastructure fixe préexistante, à l'inverse des réseaux de télécommunication classiques. Un réseau Ad-hoc doit s'organiser automatiquement de façon à être déployé rapidement et pouvoir s'adapter aux conditions du trafic et aux différents mouvements pouvant intervenir au sein des noeuds mobiles.

Dans le but d'assurer la connectivité du réseau, malgré l'absence d'infrastructure fixe, chaque noeud est susceptible d'être mis à contribution pour participer au routage et pour retransmettre les

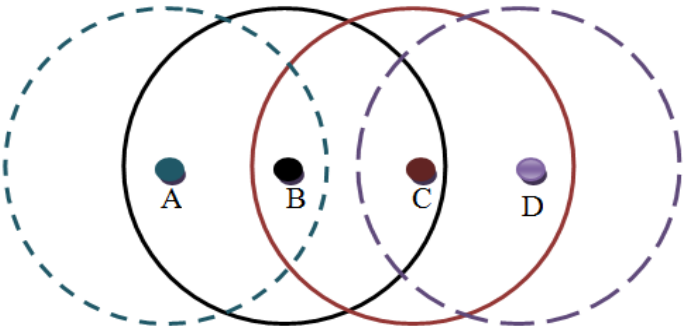
paquets d'un noeud qui n'est pas en mesure d'atteindre sa destination directement ; tout noeud joue ainsi le rôle de poste de travail et de routeur. C'est le cas du réseau de capteurs qui est un réseau Ad-hoc avec des contraintes plus fortes d'énergie, de capacité, de calcul et de stockage.

Chaque noeud participe donc au routage ce que lui permet de découvrir les chemins existants afin d'atteindre les autres noeuds du réseau. Le fait que la taille d'un réseau puisse être importante, surtout dans le cas des réseaux de capteurs, souligne que les techniques de routage dans les réseaux classiques nécessitent des modifications. Le problème qui se pose dans le contexte de ces réseaux est l'adaptation de la méthode d'acheminement utilisée à un grand nombre de noeuds possédant de modestes capacités de calculs et de sauvegarde et parfois présentant des changements de topologie.

Il est impossible qu'un noeud puisse garder les informations de routage concernant tous les autres noeuds, car le réseau peut être volumineux. Ce problème ne se pose pas dans le cas des réseaux de petites tailles, car l'inondation (la diffusion pure qui fait propager un paquet dans le réseau entier) faite pour ce but dans ces réseaux n'est pas coûteuse ; par contre dans un réseau volumineux, le manque de données de routage concernant les destinations peut impliquer une large diffusion dans le réseau. Cela – si on considère seulement la phase de découverte des routes - peut dégrader considérablement les performances du système caractérisé principalement par une faible bande passante et une capacité énergétique limitée.

Dans le cas où le noeud destination se trouve dans la portée de communication du noeud source, le routage devient évident et aucun protocole de routage n'est initié, ce qu'on appelle envoi direct ou à un seul saut. Mais ce cas est généralement rare dans les réseaux Ad-hoc et les réseaux de capteurs. Un noeud source peut avoir besoin de transférer des données à un autre noeud qui ne se trouve pas dans sa portée de communication. La Figure I.12 montre un exemple d'un réseau constitué de quatre noeuds.

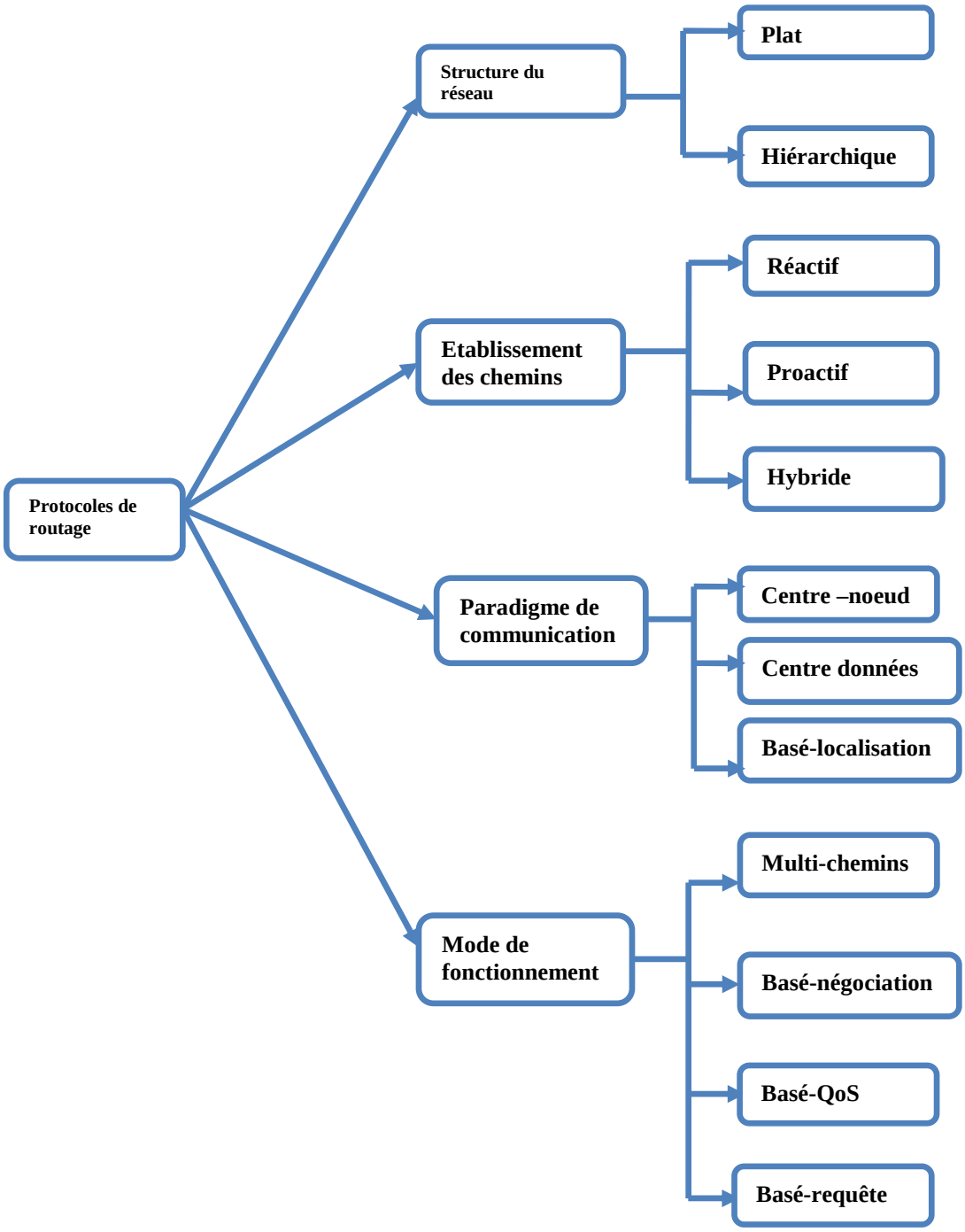
Le noeud A envoie directement un paquet à B sans besoin de routage puisque B est dans la portée de communication de A (envoi direct). D'ailleurs, si le noeud A veut envoyer un paquet au noeud D, il doit utiliser les « services » des noeuds intermédiaires B et C puisque le noeud D n'est pas dans la portée de A. A envoie le paquet à B ; B transmet le paquet à C ; C, à son tour, transmet le paquet au noeud D. Cette technique est appelée routage multi-sauts (multi-hops).



-Figure I-12 : Communication multi-sauts entre A et D *[24]*-

Dans la suite, nous décrirons des protocoles de routage dans les réseaux Ad-hoc et de capteurs. Nous les présenterons sous deux catégories : hiérarchique et non hiérarchique (ou à plat).

II.13-2 Classification des protocoles de routage



-Figure I-13 : Classification des protocoles de routage *[5]* -

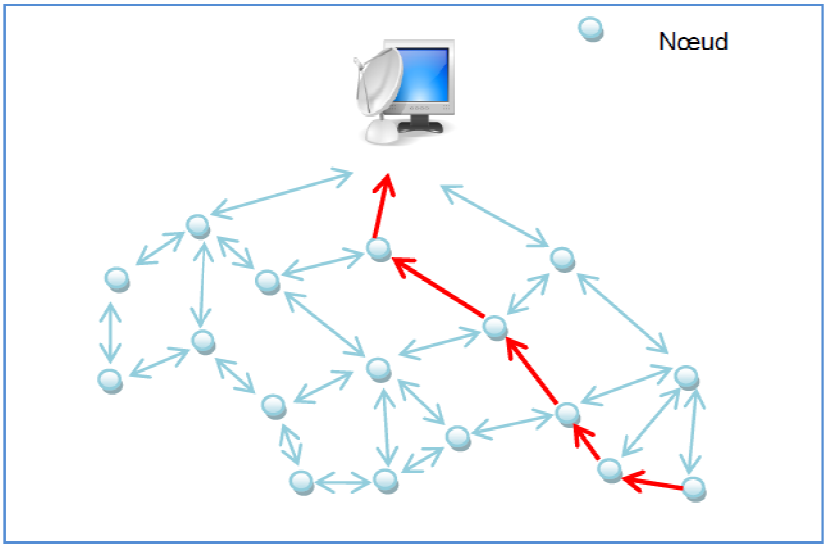
II.13-3 Classification selon la structure du réseau

La topologie détermine l'organisation des capteurs dans un RCSF. Globalement, il existe deux topologies dans les RCSFs : la topologie plate et la topologie hiérarchique.

a. Topologie plate

Dans une topologie plate tous les noeuds sont au même niveau et participent de la même manière au routage des données en adoptant un routage multi-sauts. Seul le noeud puits est chargé de la collecte des données issues des différents noeuds capteurs.

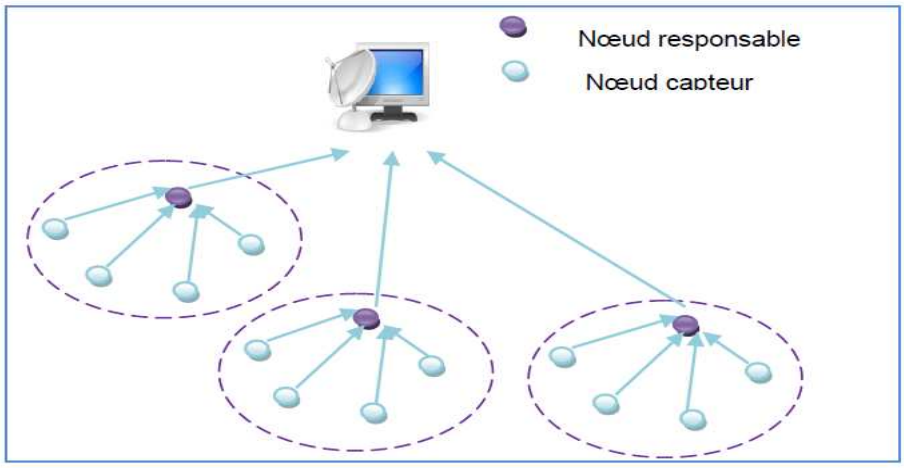
Cette architecture est caractérisée par : un coût de maintien réduit, une grande tolérance aux pannes ainsi qu'une habilité à construire de nouveaux chemins suite aux changements de topologie. Cependant, Les noeuds proches de la station de base vont participer plus que les autres aux tâches de routage. De plus, cette architecture présente une faible scalabilité due au fonctionnement identique des noeuds et d'une manière distribuée nécessitant ainsi un grand nombre de messages de contrôle.



-Figure I-14: Topologie plate *[5]*-

Selon l'application, les membres peuvent être des voisins directs ou indirects du responsable du groupe. L'architecture hiérarchique présente de grands avantages tels que l'agrégation des données collectées ainsi qu'une grande scalabilité qui assure l'ajout de nouveaux noeuds sans dégrader les performances du réseau.

Cependant, cette approche présente l'inconvénient de la surcharge des responsables qui induit un déséquilibre de la consommation d'énergie. Pour remédier à ce problème, les responsables peuvent être des capteurs spécifiques avec plus de ressources énergétiques et plus de capacités de traitement ou bien ils peuvent être élus dynamiquement et ainsi garantir un équilibre de la consommation d'énergie et augmenter la tolérance aux pannes.



-Figure I-15 : topologie hiérarchique *[5]*

b. Topologie hiérarchique

Afin d'augmenter la scalabilité du système, les topologies hiérarchiques ont été introduites en répartissant les nœuds sur plusieurs niveaux de responsabilité. Le réseau est divisé en groupes, où chaque groupe est formé d'un ensemble de nœuds membres et d'un nœud

II.13-4 Classification selon l'établissement des chemins

Le mode d'établissement des chemins permet de classer les protocoles de routage en trois catégories : proactif, réactif ou hybride.

a. Protocole proactif

Les protocoles de routage proactifs établissent au préalable les meilleures routes pour chaque noeud vers toutes les destinations possibles. Chaque nœud envoie périodiquement à tous ses voisins, sa table de routage contenant l'état de tous ses liens connus et permet ainsi, de garder une vision globale à jour de l'état du réseau.

Dans le cas d'un réseau dense, les tables de routages deviennent vite volumineuses ce qui constitue un réel inconvénient. De plus, des routes peuvent être sauvegardées sans qu'elles ne soient utilisées.

b. Protocole réactif

Dans un protocole réactif les routes sont créées à la demande. Lorsque le réseau a besoin d'une route, une procédure de découverte est lancée. Une fois la donnée routée, la route est immédiatement

détruite. Cependant, le routage à la demande induit une lenteur à cause de la durée nécessaire à rechercher les chemins, ce qui peut dégrader les performances des applications interactives.

c. Protocole hybride

Les protocoles hybrides combinent les deux idées des protocoles proactifs et réactifs. Ils utilisent des méthodes proactives pour établir les routes vers les plus proches voisins (par exemple le voisinage à deux ou à trois sauts) et des méthodes réactives au-delà de la zone de voisinage.

II.13-5 Classification selon le paradigme de communication

Le paradigme de communication détermine la manière dont les noeuds sont interrogés. Dans les RCSFs, nous pouvons distinguer trois paradigmes de communication *[**80**]* :

a. Centré-nœuds

Utilisé dans les réseaux conventionnels où il est nécessaire de connaître et d'identifier les noeuds communicants, ce modèle ne reflète pas la vision des RCSFs quant à leurs applications. En effet, dans de tels réseaux la donnée transmise est plus importante que le noeud lui-même. Cependant, le paradigme centré-noeuds reste utilisé dans certaines applications de RCSFs nécessitant une interrogation individuelle des noeuds.

b. Centré-données

Ce modèle est utilisé dans les réseaux où il n'existe pas un système d'identification global, toutes les communications sont identifiées par leurs données, l'interrogation et le routage doivent être réalisé en se basant sur cette propriété.

Le réseau peut être vu comme une base de données distribuée où les noeuds forment des tables virtuelles alimentées par les données captées.

Le protocole Directed Diffusion *[**81**]* est considéré comme l'un des protocoles de référence dans le routage centré-données.

c. Basé-localisation

Cette approche est utilisée dans les applications où il est plus intéressant d'interroger le système en se basant sur la localisation des noeuds et où on peut tirer profit des positions des noeuds pour prendre des décisions qui minimisent le nombre de messages transmis pendant le routage. Cependant, ce type de mécanismes nécessite le déploiement d'une solution de positionnement, dont le degré de

précision requis dépend de l'application ciblée. Une solution simple est l'utilisation d'un système de localisation global ou GPS mais ce genre de système reste trop coûteux pour un RCSF. Néanmoins, d'autres méthodes de localisation et de positionnement des capteurs ont été développées telles que la triangulation et la localisation par centroïdes.

II.13-6 Classification selon le mode de fonctionnement du protocole

Le mode de fonctionnement définit la manière avec laquelle les données sont propagées dans le réseau. Selon ce critère, les réseaux de capteurs sans fils peuvent être regroupés en quatre catégories.

a. Routage multi-chemins (Multipath)

Les protocoles basés sur ce type de routage utilisent des chemins multiples au lieu d'un seul dans le but d'assurer une bonne fiabilité et d'augmenter les performances du réseau. Des chemins alternatifs sont créés entre la source et la destination et sont maintenus grâce à l'envoi périodique de messages de contrôle. Malgré leur grande tolérance aux pannes, ces protocoles requièrent plus de ressources énergétiques et plus de messages de contrôle.

b. Basé sur les requêtes

Dans ce type de protocole, le puits propage des requêtes vers les noeuds capteurs. Ces derniers ayant des données à transmettre, répondent en émettant les données via le chemin inverse des requêtes. Les deux protocoles : Directed Diffusion *[***81***]* et Rumour Routing **[90]** se basent sur ce principe.

c. Routage basé sur la négociation

Les protocoles basés sur la négociation utilisent des descripteurs de données de haut niveau (métadonnées) afin de décrire la donnée avant de l'émettre. Ainsi, le récepteur éventuel peut décider de recevoir ou pas le paquet. Cette procédure garantit que seules les informations utiles seront transmises et élimine la redondance des données. Le protocole SPIN appartient à la classe des protocoles de routage basés sur la négociation.

d. Routage basé-QoS

Les protocoles de routage basés-QoS sont utilisés dans les applications qui ont des exigences temps-réel. Par exemple, dans le domaine de la sécurité, la détection d'intrusion doit être acheminée au plus bref délai vers le noeud puits. Ce type de protocoles essaye de répondre à quelques exigences de qualité de service (délai de transmission ou niveau de fiabilité) et doit faire l'équilibre avec la consommation d'énergie.

II.13-7 Exemple de protocoles centres données

Le principe de ce type de protocole est qu'une station de base (puits) envoie des requêtes à des noeuds de certaines régions et en attend une réponse. Quand une donnée est demandée à travers une requête une identification basée sur un attribut (type, instance, position…etc.) est nécessaire pour indiquer les propriétés de la donnée. Voici une sélection des protocoles les plus connus dans cette catégorie.

II.13-7 A) Flooding

C'est un mécanisme classique de transmission de donnée dans un réseau de capteur sans fil sans algorithme de routage ni algorithme de maintenance de topologie. Dans le flooding quand un noeud reçoit un paquet de donnée il le diffuse à tous ses voisins. Ce processus ne se termine que si le paquet est arrive a destination ou si le nombre maximum de saut est atteint. Le seul avantage du flooding est qu'il est facile a implémenter néanmoins il présente plusieurs inconvénient : Un noeud peut recevoir plusieurs copies de la même donnée on appel ca une implosion. Deux capteurs peuvent détecter la même donnée et l'envoyer aux mêmes voisins, c'est un overlap. La diffusion de toutes les données consomme une quantité considérable d'énergie.

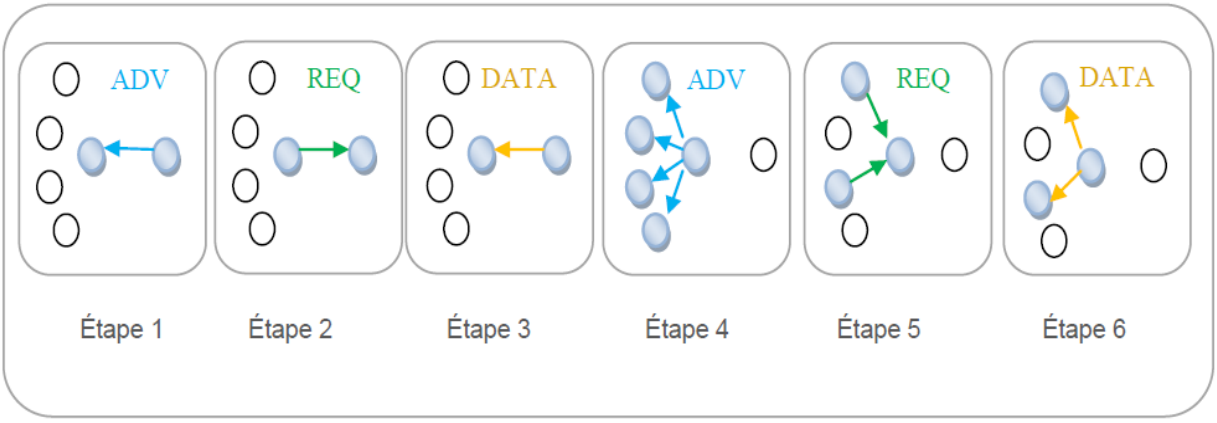
II.13-7 B) Gossiping

C'est un mécanisme de diffusion plus souple que le flooding. Un noeud envoie un paquet de données à un nombre aléatoire de voisins. Les voisins font suivre le paquet de la même manière jusqu'à l'arrivée de l'information à destination. Le Gossiping réduit les redondances de données mais l'arrivée des données à destination est retardée.

II.13-7 C) SPIN

Détecteur d'information par négociation (*Sensor Protocol for Information via Negotiation*)

Le protocole SPIN permet de disséminer des informations sur le réseau de manière ciblée. Le fonctionnement du protocole SPIN permet de réduire la charge du réseau par rapport aux méthodes de diffusion traditionnelles telles que l'inondation ou l'algorithme de Gossipin.



-Figure I-16 : Fonctionnement du protocole SPIN-

Le protocole SPIN utilise essentiellement trois types de paquets ADV/REQ/DATA. Un noeud voulant émettre une donnée commence par envoyer un paquet ADV. Ce paquet ADV consiste d'une métadonnée sur les données à émettre. Les métadonnées peuvent décrire plusieurs aspects comme le type des données et la localisation de son origine. Les noeuds qui reçoivent ce paquet vérifient si les données les intéressent. Si oui, ils répondent par un paquet REQ.

Le noeud qui a initié la communication envoie alors un paquet DATA pour chaque réponse REQ reçue (voir la Figure I-16). Un noeud peut parfaitement ne pas répondre aux messages ADV, par exemple dans le but d'économiser son énergie. Ensuite chaque noeud qui fait office de relais peut très bien agréger ses propres données aux données qui sont déjà contenues dans le paquet. *[**24**]*

Premier protocole centre-donnée basé sur la négociation entre les noeuds. Élimine les données redondantes et économise de l'énergie.

Identifie les données avec un descripteur haut niveau ou une métadonnée. La caractéristique principale de SPIN est qu'avant une transmission, des métadonnées sont échangées à travers un mécanisme d'annonce de donnée. Chaque noeud qui reçoit une nouvelle donnée l'annonce a tous ses voisins. Les voisins intéressés récupèrent la donnée en envoyant un message de demande (une requête).

Les métadonnées résolvent les problèmes classiques du flooding (donnée redondante) et économisent l'énergie. Il n'y a pas de standard de métadonnée. Il y a trois types de paquets définis dans SPIN :

• **ADV :** pour allouer un noeud et annoncer la métadonnée. Ce paquet contient la métadonnée

• **REQ :** une requête spécifique à la donnée.

• **DATA :** la donnée elle-même.

Les avantages de SPIN

• Il permet d'annoncer la disponibilité d'une donnée.

• les changements de topologie sont localisés quand un noeud a besoin de connaitre la position de ses voisins.

• il permet de multiplier l'énergie conservée par un facteur de 3.5.

• Il n'y a pas de redondance de donnée.

Les inconvénients de SPIN

Il ne garantit pas l'arrivée de l'information à la destination. Si le noeud intéressé est éloigne et que les noeuds intermédiaires ne le sont pas la donnée n'arrivera jamais au nœud intéresse.

II.13-7 D) Directed Diffusion

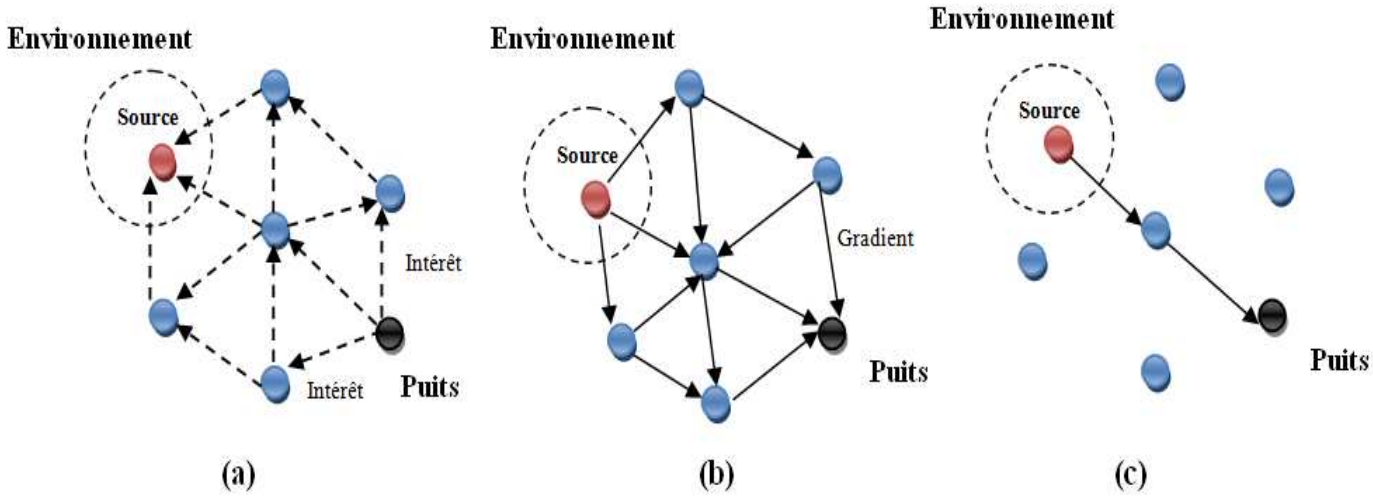
Il s'agit d'une percée dans le routage centre-donnée. D'autres protocoles ont été conçus en se basant sur ce protocole ou sur un principe similaire.

Ce protocole évite les opérations de routage inutiles. Les requêtes sont définies par des intérêts. Chaque intérêt est nommé par attribution de pairs (attribut, valeur). Le champ attribut représente un nom d'objet (par exemple : intervalle, durée d'envoie des données,..).

Le champ Valeur représente la valeur qu'a prise l'attribut durant le captage. L'intérêt (la requête) est diffusé par le puits (la station de base) vers les voisins. Chaque noeud qui reçoit un intérêt l'enregistre dans sa mémoire cache pour une utilisation ultérieure. Les intérêts sont des descriptions de taches pour les noeuds. Ils sont utilisés pour la comparaison avec les données reçues.

Le gradient est un cout acheminé vers le voisin d'ou provient l'intérêt pour renforcer le trafic dans ce chemin. Ce chemin est caractérisé par : le débit de données, la durée d'envoi des données, la date d'expiration obtenue à partir des intérêts reçus. Les chemins sont établis entre le puits (station de base) et les sources grâce aux gradients et intérêts. En se basant sur les caractéristiques des chemins, un chemin est choisi et renforce si le débit de données est faible. Le puits envoie le message original a travers le chemin choisi et renforce les nœuds sources de ce chemin. Le renforcement des chemins est réalisé dans le but d'envoyer des données plus fréquemment.

La réparation d'un chemin entre une source et le puits est effectuée s'il y a une panne. Un nouveau chemin est identifié pour remplacer le chemin en panne. La réinitialisation du renforcement se fait en se basant sur les chemins où sont envoyées les données avec un débit réduit. *[***54***]* propose de sélectionner des le début plusieurs chemins pour éviter les réparations. La **Figure I-17** illustre le mécanisme de Directed Diffusion.



-Figure I-17 : Fonctionnement de Directed Diffusion *[8]* -

En résumé le protocole comprend trois phases : la dissémination d'intérêt, la construction de gradients et la livraison des données.

Avantages de Directed Diffusion

• Il n'exige pas de mécanisme d'adressage de noeuds.

• Il utilise l'agrégation de donnée ce qui réduit les redondances de données.

• Il permet d'économiser l'énergie.

• Il n'effectue pas de maintenance de topologie ce qui réduit le nombre de transmissions donc réduit la consommation d'énergie.

Inconvénients de Directed Diffusion

• Si une application transmet des données en continue il n'est pas recommandé d'utiliser Directed Diffusion (*a cause du surcout du modèle de livraison de donnée qui est basé sur les requêtes*).

II.13-7 E) Energy Aware Routing (EAR) « routage efficace en consommation d'énergie »

Le problème avec les protocoles actuels est qu'ils trouvent le chemin optimal en énergie et l'utilisent pour toutes les communications. Cependant ce n'est pas la meilleure des méthodes pour prolonger la durée de fonctionnement d'un réseau. Utiliser fréquemment le même chemin optimal induit l'épuisement de la batterie des noeuds qui constituent ce chemin et par conséquent le réseau sera

partitionné. EAR a été introduit pour contrer ce problème par *[**84**]* Le principe de base est que, pour augmenter la durée de fonctionnement du réseau il est nécessaire d'utiliser un sous ensemble de chemins optimaux occasionnellement.

C'est une sorte d'équilibrage de charge dans le réseau. Pour assurer ce principe plusieurs chemins sont découverts, des sources vers la destination (la station de base). Un chemin est choisi aléatoirement selon une probabilité. C'est-a-dire qu'il n'y a pas de chemin qui soit utilisé fréquemment. EAR est un protocole réactif, la communication est initiée par la destination. La station de base initie le chemin et le maintient. Il est semblable à la diffusion dans certains cas. Plusieurs chemins sont maintenus entre la source et la station de base. Contrairement à Directed diffusion qui envoie les données à travers tous les chemins a intervalle régulier, EAR utilise un seul chemin à chaque fois.

Grace au choix probabiliste des chemins, il peut continuellement évaluer différents chemins et choisir les probabilités.

Il y a 3 phases dans le protocole :

- **Phase 1 :** *phase d'installation*, on produit une inondation localisée pour trouver les itinéraires et créer les tables de routage.
- **Phase 2 *:*** *phase de transmission de données*, chaque nœud expédie le paquet en choisissant aléatoirement un nœud de sa table de routage en utilisant les probabilités.
- **Phase 3 :** *phase d'entretien d'itinéraire*, une inondation locale est effectuée de temps à autre afin de maintenir tous les chemins vivants.

L'approche décrite est semblable à Directed Diffusion dans la manière dont les chemins potentiels entre la source et le puits sont découverts. Dans la diffusion dirigée, des données sont envoyées par les chemins multiples, l'un d'entre eux étant renforcés pour envoyer à des taux plus élevés.

D'une part, dans Energy Aware Routing, on choisit un chemin simple aléatoirement à partir des solutions de rechange multiples afin d'économiser l'énergie. Par conséquent, une fois comparée à Directed Diffusion, elle fournit une amélioration globale de 21.5% économie d'énergie et une augmentation de 44%de la durée de vie du réseau.

Cependant, l'approche exige recueillir les informations de localisation et d'installer un mécanisme d'adressage pour les nœuds. *[**8**]*

II.13-8 Exemple de protocoles hiérarchiques

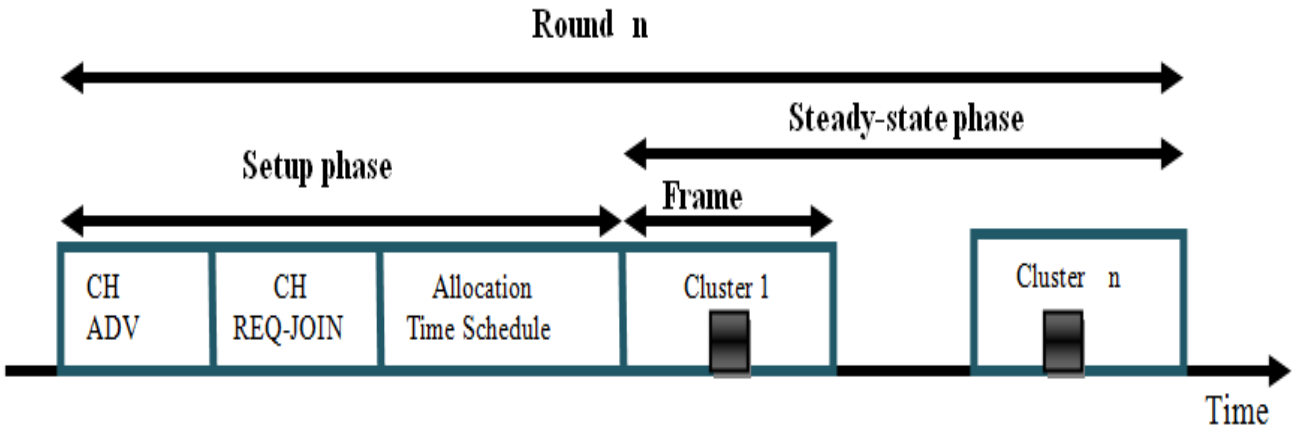
Construction de clusters (groupe de noeuds) avec un chef par cluster qui se chargera de transmettre les messages générés par son cluster aux autres chefs de clusters pour atteindre la destination finale (sink). Le choix du chef de cluster (cluster head) est fait soit à tour de rôle, soit selon le nombre de voisins en considérant comme cluster head le noeud avec le plus de voisins, soit selon le niveau de l'énergie résiduelle.

II.13-8 A) LEACH (Low Energy Adaptive Clustering Hierarchy)

A été proposé par **[23*]*** ; c'est un algorithme de clustering distribué appelé LEACH pour le routage dans les réseaux de capteurs homogènes. LEACH choisit aléatoirement les nœuds cluster-heads et attribue ce rôle aux différents noeuds selon la politique de gestion Round-Robin (c'est-à-dire tourniquet) pour garantir une dissipation équitable d'énergie entre les noeuds. Dans le but de réduire la quantité d'informations transmises à la station de base, les cluster-heads agrègent les données capturées par les noeuds membres qui appartiennent à leur propre cluster, et envoient un paquet agrégé à la station de base.

LEACH est exécuté en deux phases : la phase « set-up » et la phase « steady-state » suivant la **Figure I-18** Dans la première phase, les cluster heads sont sélectionnés et les clusters sont formés, et dans la seconde phase, le transfert de données vers la station de base aura lieu. Durant la première phase, le processus d'élection des cluster heads est déclenché pour choisir les futurs cluster heads.

Ainsi, une fraction prédéterminée de noeuds s'élisent comme cluster heads selon le schéma d'exécution suivant : durant une période T, un noeud n choisit un nombre aléatoire nb dont la valeur est comprise entre 0 et 1 ($0 < nb < 1$). Si nb est inférieure à une valeur seuil alors le noeud n deviendra cluster head durant la période courante, sinon le noeud n devrait rejoindre le cluster head le plus proche dans son voisinage *[***24***]*.



-Figure I-18 : Formation de clusters dans LEACH-

Cependant, bien que LEACH puisse augmenter la durée de vie du réseau, il présente certaines limitations. LEACH suppose que tous les noeuds puissent transmettre des données avec une grande puissance pour atteindre la station de base et que chaque noeud a une puissance de calcul lui permettant de supporter différentes couches MAC. Par conséquent, LEACH ne convient pas aux réseaux déployés dans de vastes régions. En outre, LEACH choisit aléatoirement la liste des cluster heads et il ne pose aucune contrainte sur leur distribution ainsi que sur leur niveau d'énergie. Ainsi, les cluster heads peuvent se concentrer dans un même endroit et par conséquent, il pourrait exister des noeuds isolés (sans cluster head) pouvant se déclarer. D'autre part, dans LEACH, l'agrégation des données est centralisée et est exécutée périodiquement. Or, dans certains cas, la transmission périodique des données pourrait ne pas être nécessaire, ce qui épuise rapidement l'énergie limitée des capteurs.

Une variante de LEACH appelée LEACH-C a été conçue pour améliorer les performances de LEACH. Cette variante utilise une architecture centralisée pour choisir les clusters heads tout en impliquant la station de base et l'information de localisation des capteurs. Cependant, elle augmente considérablement le surcoût du réseau puisque tous les capteurs devront envoyer leurs informations de localisation à la station de base en même temps pendant chaque phase d'élection de cluster heads. Plusieurs travaux présentés dans la littérature ont prouvé qu'une telle architecture centralisée ne supporte pas le passage à l'échelle, étant plus particulièrement appropriée à des réseaux de petite taille ***[24]***.

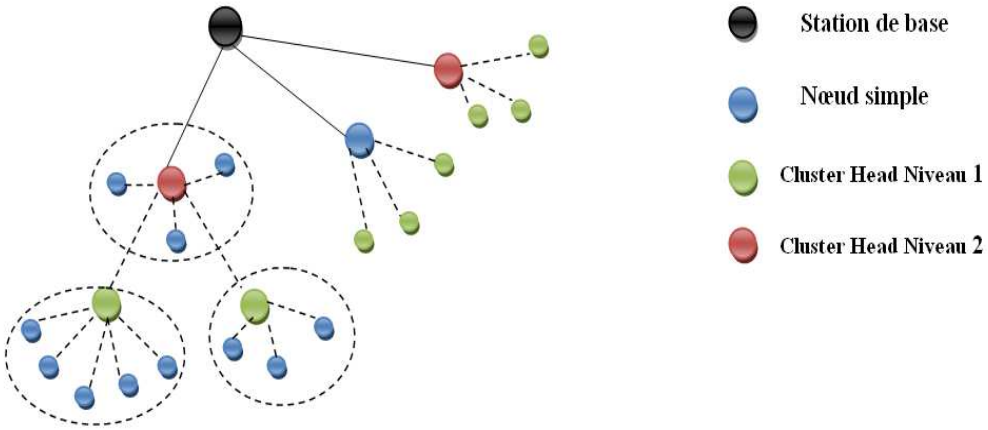
5% des noeuds sont élus pour être des têtes de clusters. Election périodique du cluster head en fonction du % de nombre de cluster, nombre de tour depuis lequel le noeud n'était pas cluster. TDMA est utilisé dans le cluster. Agrégation des données générées dans un cluster limite la redondance *[**14**]*.

II.13-8 B) TEEN (Threshold-sensitive Energy Efficient sensor Network protocol)

Manjeshwar et Agrawal *[**83**]* ont proposé une technique de clustering appelée TEEN pour les applications critiques où le changement de certains paramètres peut être brusque. L'architecture du réseau est basée sur un groupement hiérarchique à plusieurs niveaux où les noeuds les plus proches forment des clusters. Puis ce processus de clustering passe au deuxième niveau jusqu'à ce que la station de base soit atteinte. Après la formation des clusters **Figure I-19**, chaque cluster-head transmet à ses membres deux seuils : un seuil Hard HT (hard threshold), qui est la valeur seuil du paramètre contrôlé (surveillé) et un seuil Soft ST (soft threshold) représentant une petite variation de la valeur du paramètre contrôlé.

L'occurrence de cette petite variation ST permet au noeud qui la détecte de la signaler à la station de base en transmettant un message d'alerte. Par conséquent, le seuil Soft réduira le nombre de transmissions puisqu'il ne permet pas la transmission s'il y a peu ou pas de variation de la valeur du paramètre contrôlé.

Au début, les noeuds écoutent le médium continuellement et lorsque la valeur captée du paramètre contrôlé dépasse le seuil Hard, le noeud transmet l'information. La valeur captée est stockée dans une variable interne appelée SV. Puis, les noeuds ne transmettront des données que si la valeur courante du paramètre contrôlé est supérieure au seuil hard HT ou diffère du SV d'une quantité égale ou plus grande que la valeur du seuil Soft ST. Puisque la transmission d'un message consomme plus d'énergie que la détection des données, alors la consommation d'énergie dans TEEN est moins importante que dans les protocoles proactifs ou ceux qui transmettent des données périodiquement tels que LEACH. Cependant, l'inconvénient principal de ce protocole est que, si les seuils HT et ST ne sont pas reçus, les noeuds ne communiqueront jamais, et aucune donnée ne sera transmise à l'utilisateur, ainsi la station de base ne connaît pas les noeuds qui ont épuisé leur énergie. TEEN ne convient pas aux applications qui nécessitent des envois périodiques de données *[**24**]*.



-Figure I-19 : Clustering Hiérarchique dans TEEN *[8]* -

II.13-9 Exemples de protocoles basés sur la position

Dans les réseaux de capteurs, on considère que la position du noeud est plus importante que son identité (adresse). Ce type de protocoles considère que les noeuds connaissent leur position et sont capables de connaître la position des autres noeuds. Ainsi, cette information est utilisée pour diriger les messages vers la région dans laquelle se trouve la destination.

II.13-9 A) GEAR

Ce protocole de routage découpe le réseau en régions. Chaque noeud connaît le coût pour atteindre chaque région. L'acheminement des packets suit les étapes suivantes :

- acheminer le paquet jusqu'à la région, en envoyant le paquet au noeud le plus proche de la région parmi ses voisins,
- acheminer le paquet dans la région de destination par une sorte de diffusion. Chaque paquet contient la région destination. Chaque noeud connaît sa position et son énergie résiduelle et la position et l'énergie résiduelle de ses voisins (à la demande). Un lien existe entre 2 noeuds quand ils sont à portée et leur niveau d'énergie leur permet d'effectuer l'envoi *[***14***]***.**

II.13-10 Exemples de protocoles basés sur la QoS

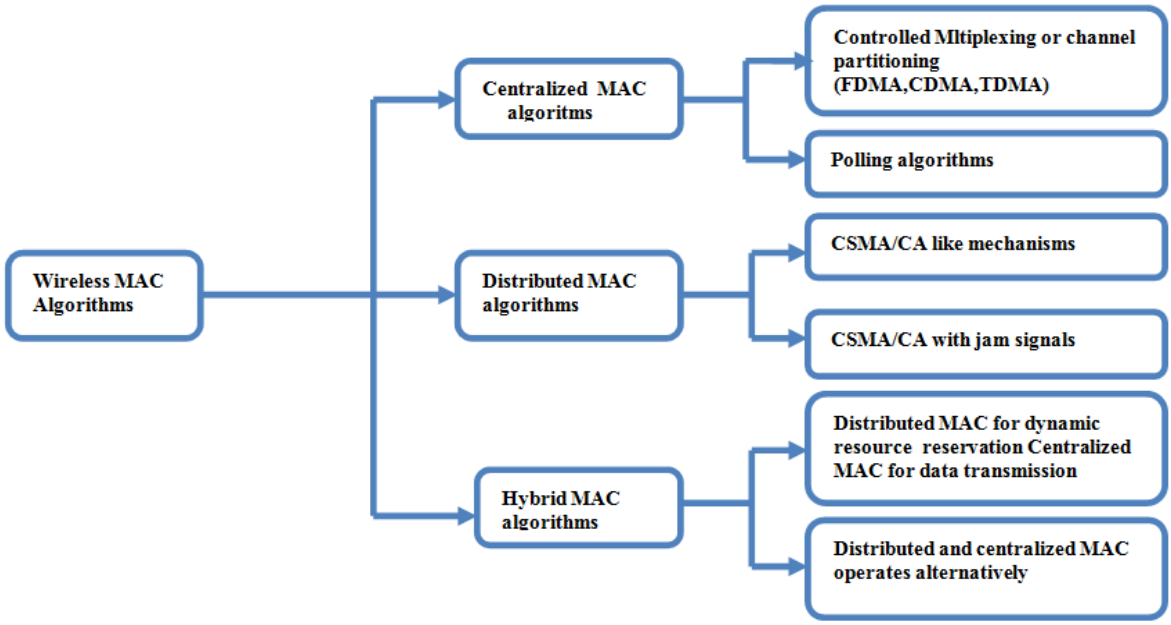
Dans ce type de protocoles, les performances de réseaux sont prises en compte pour garantir un délai de bout-en-bout raisonnable qui répond aux besoins de l'application. C'est surtout le cas des applications industrielles et militaires.

II.13-10 A) SPEED

Une métrique supplémentaire par rapport à GEAR : le délai. Se base sur une table de positions. Il estime le délai sur chaque saut en calculant le délai d'aller retour (en retranchant le temps de traitement côté récepteur). Ceci ne peut pas être une métrique stable. Le prochain saut est choisi parmi les voisins qui sont plus proche de la destination que le nœud *[**14**]*.

II.14 Les Protocoles du niveau MAC

Dans cette partie nous allons parler sur les différents protocoles MAC existant dans la littérature en s'intéressant à ceux qui traitent la problématique de l'énergie. Comme nous l'avons rencontré dans différentes références les protocoles de la couche MAC sont partagés en plusieurs classes : les protocoles fondés sur TDMA, les protocoles utilisant la contention et les protocoles hybrides.



-Figure I-20 : Classification des protocoles MAC *[77]*-

II.14-1 Protocoles à accès centralisé

Cette approche se base sur le fait d'avoir une autorité (point d'accès) qui règle l'accès au medium. Cette entité de contrôle coordonne l'accès au canal entre les nœuds selon trois techniques : FDMA (*Frequency Division Media Access*), CDMA (*Code Division Media Access*) et TDMA (*Time Division Media Access*).

II.14-1 A) TDMA

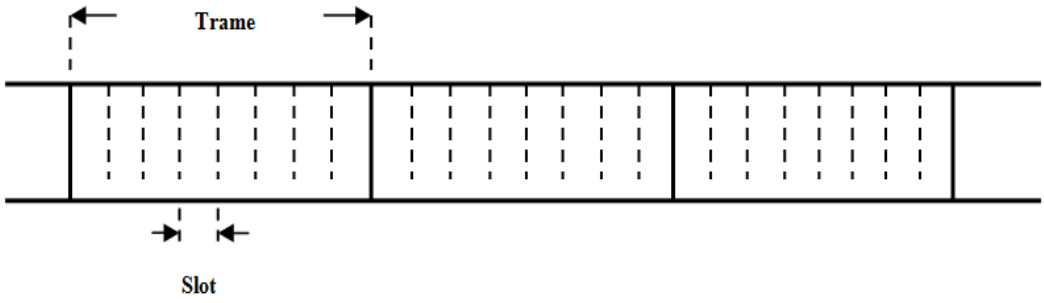
Dans les protocoles MAC fondés sur la méthode TDMA le temps est divisé en trames (périodiques) et chaque trame se compose d'un certain nombre de slots de temps. À chaque nœud est attribué un ou plusieurs slots par trame, selon un certain algorithme d'ordonnancement. Il utilise ces slots pour l'émission/réception de paquets de/vers d'autres nœuds. Dans de nombreux cas, les nœuds sont regroupés pour former des clusters avec un clusterhead qui est chargé d'attribuer les slots de temps pour les nœuds de son cluster **[3]**.

Ces protocoles sont généralement économes en énergie car ils évitent les collisions, ils limitent le idle ***listening*** et le ***overhearing***, et mettent les nœuds en ***mode sommeil*** durant les slots de temps réservés aux autres nœuds. De plus, ils garantissent un délai borné de bout-en-bout si un algorithme d'allocation prend en compte les caractéristiques du trafic. En revanche, l'aspect centralisé et le besoin d'une synchronisation rendent ce type de protocoles rigide et non-adapté pour les topologies dynamiques et mobiles **[14]**.

TDMA évite les collisions et utilise une bonne consommation d'énergie par contre le problème de latence apparaît. Imaginons le cas où il y a ***N*** nœuds donc ***N slots*** et chaque ***slot*** a un temps ***Ts***. La durée d'une trame est alors de :

$$Tf = N * Ts$$

Le récepteur doit identifier chaque paquet dans un ***slot*** afin de lire les informations qui lui sont destinées. L'approche TDMA est basée autour d'un élément central appelé point d'accès qui permet d'associer à chaque nœud un ***slot***. Lorsqu'un ***slot*** est attribué à un nœud, les autres nœuds peuvent ***éteindre leurs radios*** **[11]**.



-Figure I-21 : Les trames TDMA [11] -

II.14-1 B) TRAMA

TRAMA (TRaffic-Adaptive Medium Access control) basé sur l'algorithme TDMA, il divise le temps en slots de temps et se base sur le trafic annoncé par les nœuds pour désigner les émetteurs et les récepteurs pour chaque slot de temps.

L'algorithme de réservation des slots est le suivant. Tout d'abord, les nœuds cherchent des informations sur un voisinage à deux sauts, qui sont nécessaires pour établir un ordonnancement sans collisions. Ensuite, les nœuds commencent une procédure d'élection afin d'associer chaque slot à un seul nœud.

Chaque nœud aura une priorité pour être le propriétaire d'un slot. Cette priorité est calculée avec une fonction de hachage de l'identifiant du nœud et du numéro du slot. Le nœud avec la plus grande priorité devient le propriétaire du slot. Enfin, les nœuds envoient un paquet de synchronisation contenant la liste des voisins destinataires pour les transmissions suivantes. Par conséquent, les nœuds peuvent se mettre d'accord sur les slots où ils doivent être éveillés. Les slots inutilisés peuvent être annoncés par leurs propriétaires pour être réutilisés par d'autres.

TRAMA applique trois mécanismes :

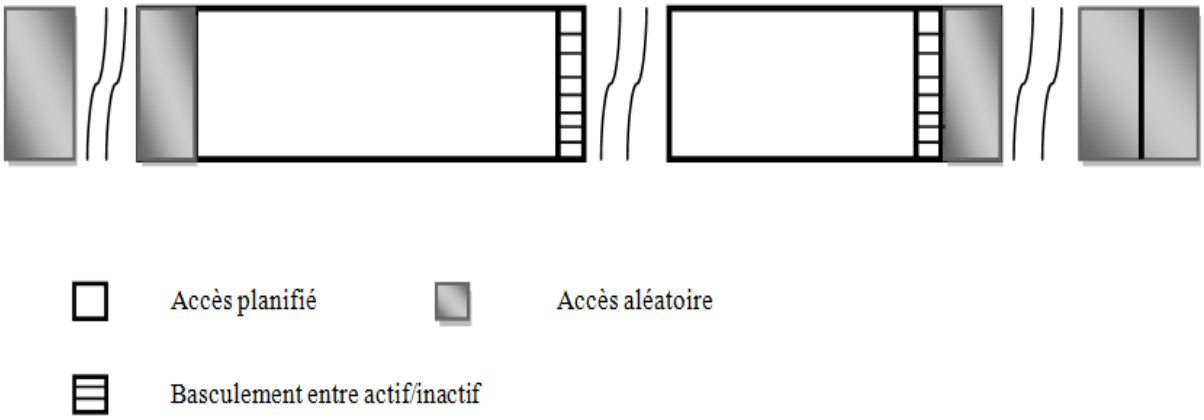
1) un protocole de voisinage appelé ***NP (Neighbor Protocol)*** pour échanger la liste des voisins à un saut entre les nœuds, ce qui permet d'établir une connaissance des voisins à deux sauts,

2) un protocole d'échange de calendriers appelé ***SEP (Schedule Exchange Protocol)*** où chaque nœud annonce ce qu'il a comme trafic à envoyer en précisant les récepteurs concernés,

3) un protocole d'élection adaptative ***AEA (Adaptative Election Algorithm)*** qui choisit les émetteurs et les récepteurs pour un slot de temps donné en fonction des informations collectées par ***NP*** et ***SEP***.

TRAMA découpe le temps en alternant des intervalles de temps à accès planifié et des intervalles de temps à accès aléatoire. Chaque intervalle est constitué de slots de temps. ***TRAMA*** commence par

des intervalles à accès aléatoire pour permettre au réseau de s'établir. Les échanges pour la découverte de voisinage du protocole ***NP*** sont effectués durant les intervalles à accès aléatoire. Ces intervalles servent aussi pour permettre aux nouveaux nœuds de rejoindre le réseau. L'envoi de trames de données se fait durant les intervalles à accès planifié voir **Figure I-22 [14]**.



-Figure I-22 : Découpage temporel TRAMA-

Avantages

- L'accès programmé aux *slots* de données limite les collisions des messages et réduit l'énergie nécessaire de l'unité de communication.
- L'accès aléatoire permet au protocole de s'adapter rapidement aux changements de trafic.
- Pour agrandir le temps de ***mise en veille*** d'un nœud capteur, TRAMA regroupe les slots de données d'un nœud vers la fin de la trame.
- La probabilité de collisions est largement inférieure à celle des protocoles CSMA/CA.
- Les communications peuvent être de différents modèles : *unicast*, *multicast* ou *broadcast*

Inconvénients

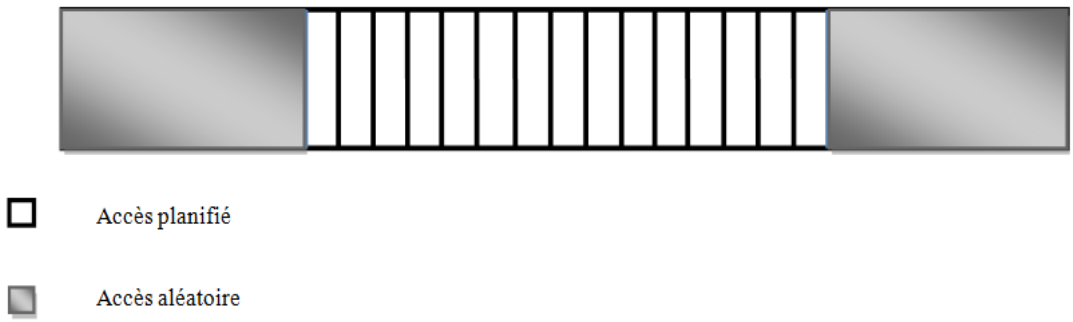
- L'algorithme de TRAMA est compliqué.
- TRAMA utilise des ressources plus intensivement par rapport à d'autres protocoles. Les nœuds doivent avoir des informations sur leurs voisins à deux sauts, ce qui implique la nécessité d'une grande mémoire, vu que les RCSF sont très denses.

II.14-1 C) FLAMA

FLAMA (FLow-Aware Medium Access) est proposé comme une amélioration de ***TRAMA***. Comme *TRAMA*, *FLAMA* divise le temps selon deux modes d'activité : intervalles de temps à accès planifié et intervalles de temps à accès aléatoire. *FLAMA* a aussi besoin de connaitre le voisinage d'un nœud à deux sauts et des informations concernant le flux de données des voisins à un saut **[14]**.

En revanche, *FLAMA* ne diffuse pas de calendriers de trafic durant les intervalles de temps à accès planifié (**Figure I-23),** mais échange durant les intervalles de temps à accès aléatoire des informations par rapport aux flux de données liés à l'application. En outre, *FLAMA* établit une synchronisation globale du réseau basé sur l'estampillage des trames et une topologie arborescente pour le relais de données.
Pour déterminer les émetteurs et les récepteurs concernés par chaque slot de temps lors de l'intervalle à accès planifié, *FLAMA* applique un algorithme plus simple que celui utilisé dans *TRAMA*.

Comme *TRAMA*, *FLAMA* assure des transmissions sans collision en ne permettant qu'à un seul nœud d'émettre dans un voisinage à deux sauts. *FLAMA* est plus performant que *TRAMA* en termes d'économie d'énergie et de perte de trames. De plus, *FLAMA* améliore le délai de bout en bout par rapport à *TRAMA* grâce à l'exploitation des routes définies et partir d'un arbre pour le cheminement multi-sauts. En revanche, *FLAMA* surcharge le réseau par des échanges pour connaître le flux de l'application et établir une synchronisation globale du réseau de proche en proche.



- Figure I-23 : Trame FLMA-

II.14-1 D) FDMA

Dans l'approche ***FDMA (pour Frequency Division Multiple Access),*** la bande passante est divisée en canaux et chaque nœud peut communiquer en utilisant son propre canal. Ceci permet aux nœuds de communiquer simultanément. Le problème des collisions est minimisé puisque les nœuds communiquent à travers des canaux de radio séparés. Cependant, une faible bande passante est disponible et ainsi la consommation d'énergie augmentera. Contrairement à *TDMA*, *FDMA* élimine le problème de latence. Pour implémenter *FDMA* dans les réseaux de capteurs, les nœuds doivent être équipés d'un système de radio complexe qui sera capable de recevoir des signaux provenant de plusieurs canaux, c'est pour cela que l'utilisation de *FDMA* dans les réseaux de capteurs est limitée **[11]**.

Dans ce type de protocoles MAC, chaque nœud possède plusieurs bandes de fréquence différentes et connait la fréquence de ses voisins. Quand il veut transmettre des trames de données à un de ses voisins, il choisit la fréquence correspondante pour transmettre. Les nœuds peuvent faire plusieurs transmissions simultanément dans la portée des autres sans les interférer **[14]**.

FDMA requiert une conception complexe avec des circuits complémentaires, ce qui augmente le prix des dispositifs. De ce fait, FDMA ne constitue pas un bon candidat comme protocole MAC pour réseaux de capteurs sans fil **[5].**

Le FDMA est simple à mettre en œuvre mais son inconvénient principal est qu'il faut un récepteur dédié par canal à écouter. Cette technique de multiplexage est la plus ancienne ; elle est encore largement utilisée dans les diffusions comme la télévision et la radio **[32].**

II.14-1 E) CDMA **(*Code Division Multiple Access*)**

La technique ***CDMA (pour Code Division Multiple Access),*** autorise l'allocation de la totalité de la bande de fréquences. Tous les nœuds communiquent simultanément. Pour ce faire, un code spécifique est octroyé à chaque nœud qui l'utilisera pour transmettre l'information qu'il désire communiquer en format binaire d'une manière orthogonale aux autres communications. Toutefois, un problème d'auto interférence entre en jeu qui s'intensifie au fur et à mesure que le nombre de communications simultanées augmente. Dans un réseau multi saut, un nœud qui relaye les données

doit disposer d'une grande capacité mémoire pour stocker les codes de la plupart ou de la totalité des nœuds, ce qui peut dépasser la capacité mémoire d'un capteur **[11].**

Dans **CDMA** on considère que différentes trames de données peuvent se chevaucher de façon linéaire. Ainsi, chaque nœud possède un code différent et garde à son niveau une liste des codes de tous ses voisins. Quand un nœud veut transmettre une trame de données, il code sa trame de données avec son propre code et l'envoie au récepteur. Plusieurs nœuds peuvent transmettre les données codées en même temps en utilisant leur propre code. Le récepteur utilise le code de son émetteur pour recevoir ses trames. En utilisant le code de son émetteur, tous les bruits créés par les autres émetteurs sont détruits et le récepteur peut recevoir une bonne trame de données même s'il y a des interférences. Cependant, le protocole **CDMA** nécessite d'importantes capacités de calcul **[5].**

II.14-1 F) E-MAC

E-MAC (Event MAC) découpe le temps en slots, chaque slot est lui-même découpé en trois parties respectivement prévues pour accueillir les requêtes de communication, le trafic de contrôle et le trafic de données. E-MAC définit trois types de nœuds : les nœuds actifs possédant un slot, les nœuds passifs ne possédant pas de slot et ne transmettant qu'après avoir fait une requête pour utiliser le slot d'un voisin durant la partie dédiée aux requêtes de communication du slot du voisin, et les nœuds dormants qui ***dorment*** la plupart du temps et choisissent ***un mode passif*** ou actif quand ils se réveillent. Chaque nœud diffuse une information concernant les slots utilisés par ses voisins durant la partie trafic de contrôle de son slot. Les nœuds doivent tous se réveiller durant la partie trafic de contrôle des slots de leurs voisins **[14].**

L'allocation des slots commence par une station de base qui choisit un slot et annonce ce choix par une diffusion. Ensuite, ses voisins choisissent aléatoirement un slot. En cas de choix d'un même slot, les nœuds signalent ce fait durant la partie trafic de contrôle. Les slots sont réutilisés à partir de trois sauts.

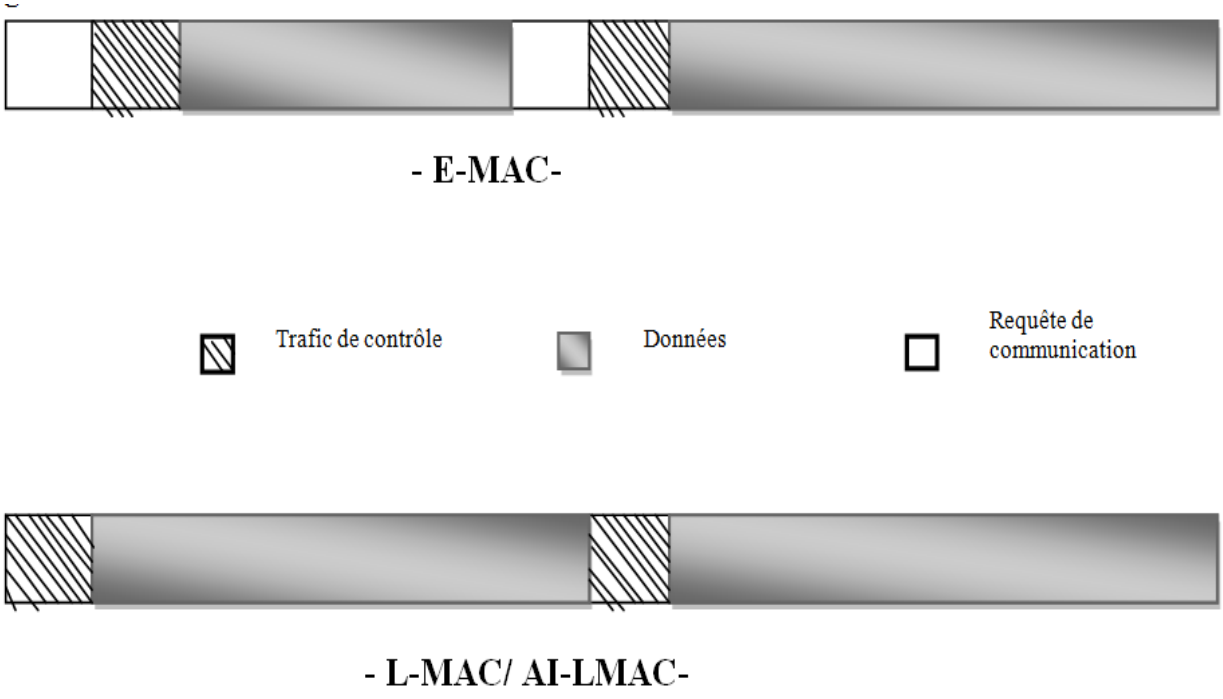
II.14-1 G) L-MAC

L-MAC (Lightweight-MAC) adopte le même mécanisme d'allocation de slots et de découpage temporel que *E-MAC*. En revanche, *L-MAC* force tous les nœuds à avoir au moins un slot.

Ainsi, la partie requête de communication du slot n'est plus présente dans *L-MAC* et un slot est fractionné en 2 parties seulement.

II.14-1 K) AI-LMAC

AI-LMAC (Adaptive, Information-centric and Lightweight MAC) est une amélioration de *L-MAC* qui prend en compte les conditions du trafic applicatif et offre la possibilité aux nœuds d'avoir plusieurs slots. *AI-LMAC* regroupe les nœuds avec des relations père-fils. Le père surveille les conditions de trafic de ses fils et demande à ses fils de s'allouer plus de slots ou de se désallouer des slots selon leur charge.



-Figure I-24 : montre le découpage temporel en slots des trois protocoles-

II.14-2 Protocoles basés sur la contention

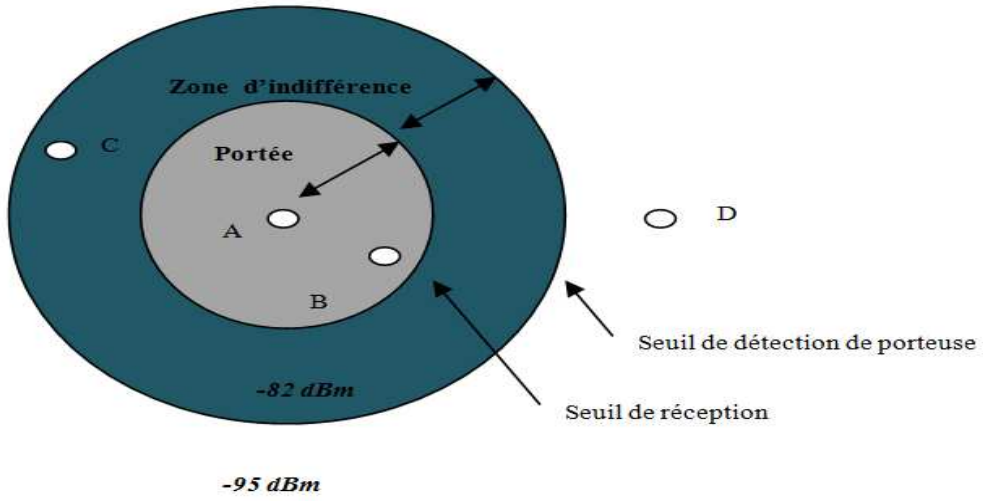
Dans ce type de protocoles *MAC*, les nœuds accèdent au médium durant le même intervalle de temps en utilisant un algorithme de la famille *CSMA/CA*, chacune de ses variantes essayant d'éviter les collisions. Le point fort de ce type de protocoles est sans doute l'extensibilité et le passage à l'échelle.

Nous allons commencer cette partie en décrivant brièvement l'algorithme de *CSMA/CA* de la norme *IEEE 802.11* en mode *DCF*, sur lequel est basée la plupart des protocoles qui exploitent de différentes façons une période de contention. Selon **[77] [14]** que nous avons utilisé pour tirer profit de leurs définitions « ces protocoles n'offrent pas de délai borné du fait qu'ils ne garantissent pas l'accès au médium dès que la charge du réseau augmente ».

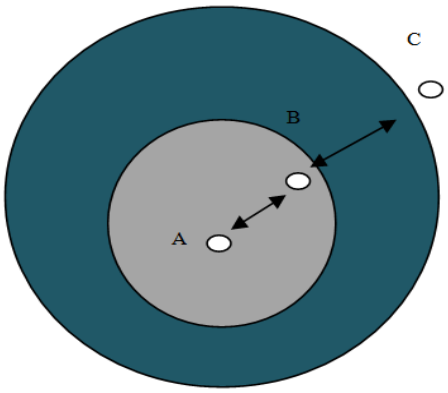
Les protocoles avec contention sont les plus populaires et représentent la majorité des protocoles *MAC* proposés pour les réseaux de capteurs sans fil. Ils assurent le *duty-cycle* par une intégration étroite des fonctionnalités d'accès au canal avec un régime *sleep/wakeup*. La seule différence est que, dans ce cas, l'algorithme *sleep/wakeup* n'est pas un protocole indépendant **[3]**.

II.14-2 A) CSMA/CA de la norme IEEE 802.11

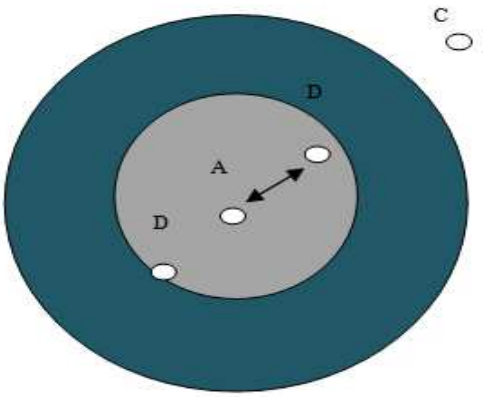
CSMA/CA est une méthode d'accès de la même famille que CSMA/CD (Carrier Sense Multiple Access with Collision Detection) puisqu'elle impose à un émetteur de s'assurer que le canal est libre avant d'émettre. Dans CSMA/CA, les collisions ne peuvent pas être détectées comme dans le CSMA/CD, un nœud essaie d'éviter les collisions (sans vraiment les éviter à 100 %). Ceci à cause de l'effet d'aveuglement du médium sans fil (Near Far Effect) qui empêche une entité de recevoir quand elle est en train d'émettre *[***14***]*.



-Figure I-25 : Portée et zone d'interférence-



-Portée et zone d'interférence-



-FigureI-26 :(communication CSMA/CA) A communique avec D. B veut communique avec C mais ne le fait pas parce que A occupe le médium, alors que si B communique avec C aucune collision n'aura lieu

II.14-2 B) IEEE 802.15.4

IEEE 802.15.4 est un standard à faible débit et à faible puissance pour les réseaux personnels (PAN pour Personal Area Networks). Un PAN est formé d'un PAN coordinator qui gère l'ensemble du réseau et, éventuellement, d'un ou plusieurs coordinateurs qui gèrent les sous-ensembles de nœuds du réseau. D'autres nœuds (ordinaires) doivent s'associer à un coordinateur afin de communiquer. Les topologies de réseau possibles sont l'étoile (saut unique), le cluster -tree et le mesh (multi-sauts). Le standard *IEEE 802.15.4* prend en charge deux méthodes d'accès aux différents canaux : un mode beacon-enabled et un mode non-beacon enabled.

Le mode *beacon enabled* fournit un mécanisme de gestion d'énergie sur la base du ***duty-cycle***. Concrètement, il utilise une structure de super trame qui est délimitée par des balises. D'autres trames de synchronisation sont générées périodiquement par les nœuds coordinateurs. Chaque super trame consiste en une période active et ***une période inactive***. Dans la période d'activité les dispositifs communiquent avec le coordonnateur auquel ils sont associés. La période active peut être divisée en une période d'accès avec contention *CAP* et une période sans contention *(CFP*).

Au cours de la CAP un algorithme *CSMA/CA* discrétisé est utilisé pour accéder au canal, tandis que, durant la CFP, un certain nombre de slots garantis (GTS) peuvent être attribués à chaque nœud. Au cours de ***la période inactive*** les dispositifs entrent en mode faible puissance pour économiser l'énergie. Dans le mode non-beacon enabled, il n'y a pas de structure en super trame, les nœuds sont toujours à l'état actif et utilisent l'algorithme Unslotted CSMA/CA pour l'accès au canal et la

transmission de données. Dans ce cas, la conservation d'énergie a lieu au niveau des couches supérieures.

Les protocoles fondés sur la contention sont robustes et garantissent le passage à l'échelle. En outre, ils ont généralement un délai plus faible que ceux reposant sur TDMA et ils peuvent facilement s'adapter aux conditions de trafic. Malheureusement, leur dissipation d'énergie est plus élevée que celle des protocoles TDMA à cause de la contention et des collisions. Des mécanismes *Duty-cycle* peuvent contribuer à réduire la surconsommation d'énergie, mais ils doivent être conçus avec soin pour être flexibles et à faible latence.

II.14-2 C) S-MAC

S-MAC (Sensor-MAC) est conçu pour assurer une méthode d'accès économe en énergie pour les réseaux de capteurs sans fil. Pour ce faire, les nœuds se mettent en *mode sommeil* pendant une certaine durée et se réveillent pour écouter le médium pendant une autre durée. [78] Il est basé sur la méthode CSMA/CA, utilise le mécanisme RTS/CTS dans le but de traiter le problème des nœuds cachés. S-MAC introduit une période d'activité et de *mise en veille*. Les nœuds doivent être synchronisés pour pouvoir communiquer. Pour ce faire, les nœuds sont organisés en *clusters* virtuels. Chaque nœud diffuse périodiquement son programme de *scheduling* (sa période d'activité et de *mise en veille*) dans un paquet SYNC aux autres nœuds dans le même *cluster*.

Si un nœud appartient à deux *clusters*, il doit réaliser un compromis entre les deux programmes de *scheduling*. S-MAC utilise la notion de *message passing* qui permet aux fragments d'un long message d'être envoyés en rafale. Comme dans le cas de IEEE 802.11, la durée de la transmission est connue par le NAV qui est inclus dans les paquets de contrôle RTS et CTS. S-MAC ajoute la durée de la transmission restante dans chaque fragment et dans chaque ACK (**Figure I-27)**, cela permettra aux nœuds qui se réveillent au milieu de la transmission de *retourner dormir* **[79].**

Avantages

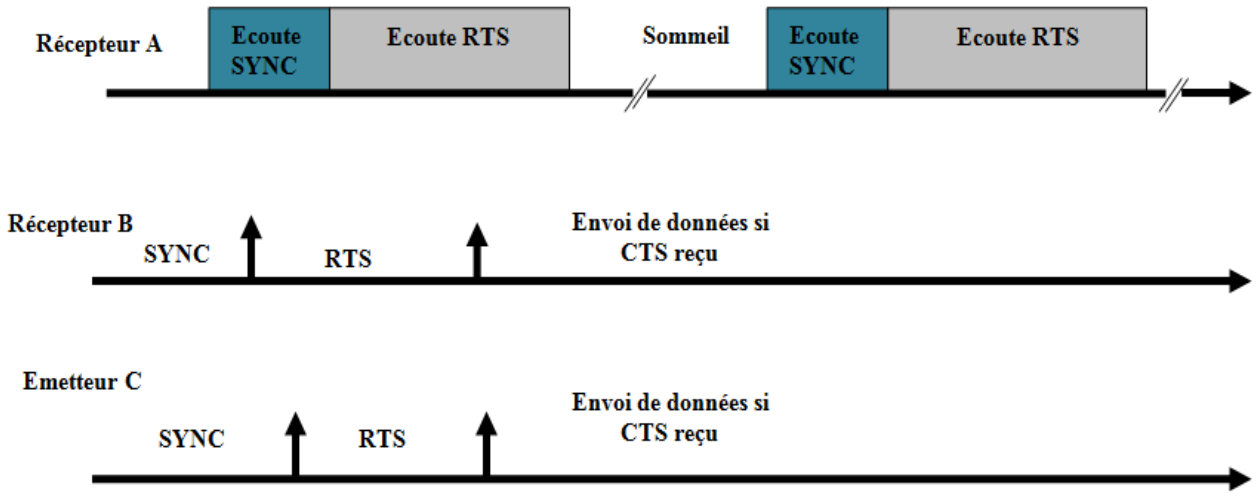
- La perte d'énergie causée par l'écoute au canal libre est réduite par le cycle de *mise en veille.*
- Le protocole est simple.

Inconvénients

- Le protocole n'est pas *scalable.* L'augmentation de nombre de nœuds implique l'augmentation de nombre des *schedule* sauvegardés dans chaque nœud.
- La latence est augmentée puisque les messages doivent attendre la période d'écoute pour être envoyés.
- L'utilisation d'une période d'écoute fixe cause l'écoute à un canal libre s'il n'y a pas de données à transmettre pendant cette période.
- La période fixe est déterminée avant le déploiement des nœuds, ce qui n'est pas adaptatif car le réglage optimal dépend aussi du taux des événements observés qui peut changer avec le temps.
- Le débit moyen est réduit, car seulement la période d'activité est utilisée pour la communication.

Les nœuds échangent leur calendrier de périodes d'écoute en le diffusant à leurs voisins à un saut. Ainsi, chaque nœud connaît le calendrier de ses voisins et sait quand il faut se réveiller pour communiquer avec un nœud à sa portée.

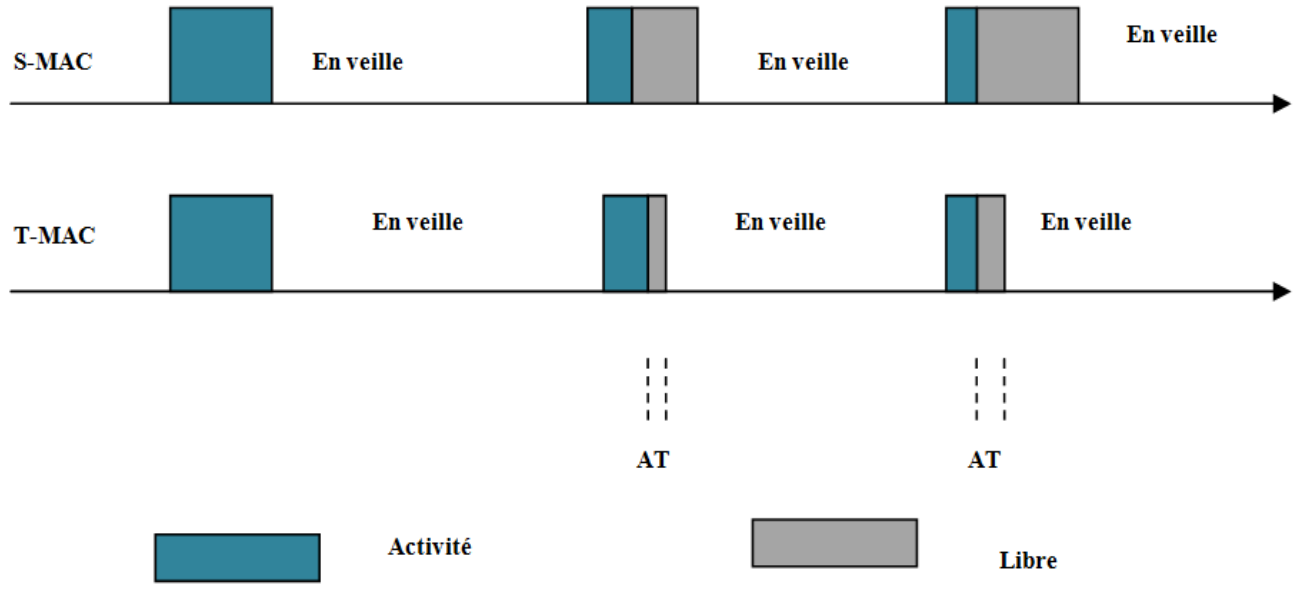
Plusieurs nœuds peuvent avoir le même intervalle de temps comme période d'écoute. Les nœuds accèdent au médium en utilisant le CSMA/CA de IEEE 802.11 avec le mécanisme RTS/CTS. En outre, un champ supplémentaire est ajouté à tous les messages (y compris les messages RTS/CTS et les acquittements) indiquant la durée de l'´echange, ce qui permet aux nœuds non concernés de ***dormir*** pendant cette durée. Pour maintenir une synchronisation des horloges, les nœuds émetteurs envoient des messages de synchronisation SYNC au début de la période d'écoute de leurs voisins.



-Figure I-27 : Séquencement des périodes d'écoute et de sommeil dans S-MAC[14]

II.14-2 D) TMAC

Le protocole **T-MAC** (pour *Timeout* MAC) est proposé pour remédier au problème de la période d'écoute fixe de S-MAC. Dans T-MAC, la période d'écoute se termine quand il n'y a plus d'activité sur le canal pendant une période adaptive ***TA***. Cette dernière représente le temps minimum d'écoute à un canal libre. En utilisant cette période adaptive, les nœuds sauvegardent leurs énergies en minimisant l'écoute à un canal libre.



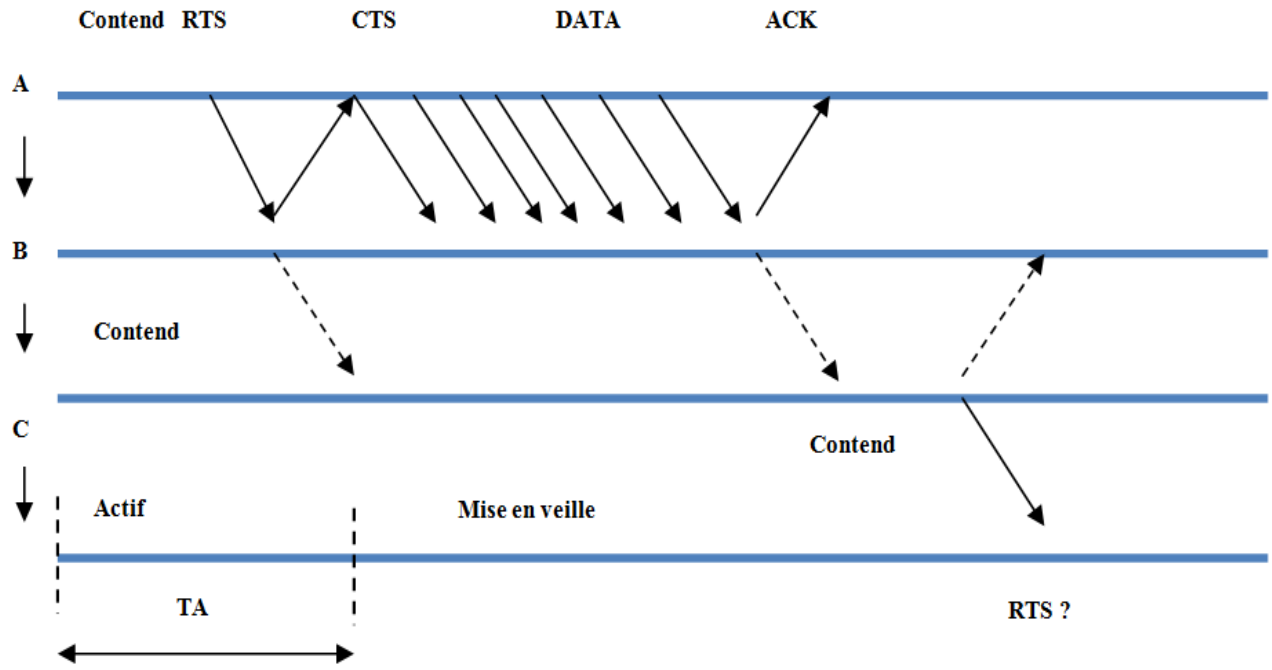
-Figure I-28 : La période d'écoute adaptative de T-MAC vs la période d'écoute fixe de S-MAC-[14]

Avec cette approche, T-MAC fait dormir un nœud sans s'être assuré que ses voisins n'ont plus de données à lui envoyer. En effet, les données à envoyer du voisin ont pu être retardées à cause d'un échec d'accès au canal.

Suivant le schéma si dessous ; considérons le cas où le nœud C qui a entendu le CTS envoyé par B en réponse au RTS de A. C doit rester silencieux pendant la transmission donc il ne communique pas avec D. Puisque D n'a rien entendu et sa période d'écoute est terminée alors il se met

en mode sommeil. Quand A termine sa transmission, B émet un paquet d'acquittement que ses voisins A et C reçoivent. C veut communiquer avec D mais ce dernier *étant en sommeil*, il ne peut pas recevoir le paquet RTS de C.

Une solution est proposée à ce problème en introduisant un paquet de contrôle FRTS (pour *Futur Request To Send*). Quand le nœud C reçoit le CTS de B, il envoie un paquet FRTS à D qui contient la longueur de la communication en cours. Ainsi, le nœud D peut déterminer qu'il sera ensuite le destinataire d'un paquet RTS et il ne se met pas en *mode sommeil*. T-MAC ajoute l'information *full-buffer-priority*. Les nœuds qui ont le *buffer* plein envoient un message RTS en réponse d'un RTS des autres nœuds **[79].**



-Figure I-29 : Le problème de mise en veille prématurée-

Avantages

- T-MAC conserve l'énergie plus que S-MAC puisqu'il évite l'écoute à un canal libre même dans la période d'écoute.
- T-MAC est aussi simple que S-MAC.
- T-MAC contient beaucoup de messages de contrôle (*overhead*).

Inconvénients

- T-MAC comme S-MAC n'est pas *scalable*.
- La diffusion de paquets n'utilise pas le mécanisme RTS/CTS, ce qui augmente la probabilité de collisions.
-

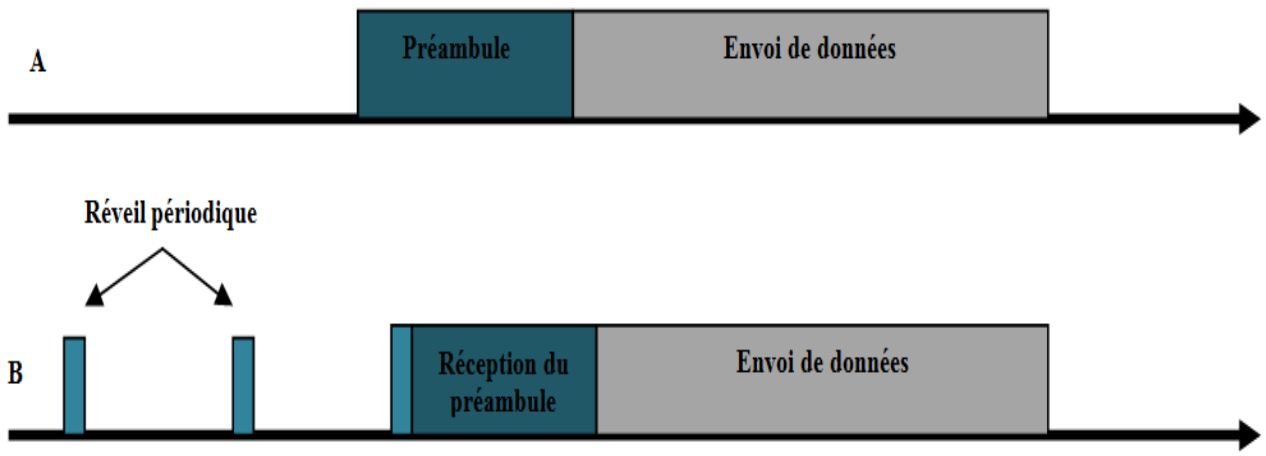
II.14-2 D) D-MAC

Early sleep : Ce problème est appelé *le sommeil prématuré*.

D-MAC (Data gathering MAC) propose un séquencement des périodes d'activité qui favorise la collecte d'informations dans une topologie arborescente. Les noeuds de même niveau se réveillent en même temps.

II.14-2 E) LPL : B-MAC, WiseMAC, B-MAC+, X-MAC et DW-LPL

Le LPL (Low Power Listening) est l'une des premières approches pour réduire l'idle listening en introduisant *une période d'inactivité au niveau de la couche physique*. L'idée est de pénaliser les émetteurs en envoyant un préambule long pour économiser l'énergie des récepteurs. Les récepteurs activent leur module radio périodiquement pour détecter la présence d'un préambule. La durée de transmission du préambule doit être de la même longueur que l'intervalle entre deux réveils d'un récepteur.



-Figure I-30 :l'émetteur utilise un long préambule pour permettre au récepteur d'activer son module radio seulement de temps en temps-[14]

II.14-2 E) BMAC

BMAC (MAC Berkeley), avec une faible complexité et une faible consommation induite par le système d'exploitation TinyOS. L'objectif de B-MAC est de fournir quelques fonctionnalités de base et un mécanisme efficace en énergie pour l'accès au canal.

Il met d'abord en œuvre les caractéristiques de base du contrôle d'accès au canal : un algorithme de backoff, une estimation efficace du canal et des acquittements optionnels. Deuxièmement, pour atteindre un faible duty-cycle, B-MAC utilise un plan *sleep/wakeup* asynchrone fondé sur l'écoute périodique appelée *Low Power Listening (LPL).* Les nœuds se réveillent périodiquement pour vérifier l'activité sur le canal.

La période entre deux réveils est nommée intervalle de vérification. Après le réveil, les nœuds restent actifs pour un temps de réveil, afin de détecter d'éventuelles transmissions. Contrairement au temps de réveil qui est fixé, l'intervalle de vérification peut être spécifié par l'application.

Les paquets *B-MAC* sont constitués d'un long préambule et d'une charge utile. La durée du préambule est au moins égale à l'intervalle de vérification, afin que chaque nœud puisse toujours détecter une éventuelle transmission au cours de son intervalle de vérification. Cette approche ne nécessite pas que les nœuds soient synchronisés. En fait, quand un nœud détecte l'activité sur le canal, il reste actif et reçoit le préambule en premier puis la charge utile **[3].**

S'il y a une activité, ils se réveillent pour recevoir le paquet. Ce mécanisme réduit l'écoute à un canal libre en augmentant les coûts de transmission et de réception, puisque l'émetteur doit envoyer un préambule de longueur plus grande que le cycle d'échantillonnage des nœuds. En plus, BMAC a implémenté des interfaces qui permettent aux services du réseau d'ajuster les mécanismes de BMAC comme *CCA* (*Clear Channel Assessment*), l'acquittement, *backoff* pour l'arbitrage au canal et *LPL* (*Low Power Listening*) pour la conservation d'énergie.

Avantages

- BMAC évite l'*overmiting* en utilisant le préambule *wake-up*.
- BMAC ne nécessite pas une synchronisation pour que les nœuds *dorment* et se réveillent en même temps.
- BMAC est simple, *scalable*, flexible et s'adapte aux applications grâce à l'utilisation des interfaces.
- BMAC est aussi évalué par une implémentation réelle d'un réseau de capteur sans fil.
- BMAC augmente la capacité utile car il n'utilise pas le mécanisme RTS/CTS.

Inconvénients

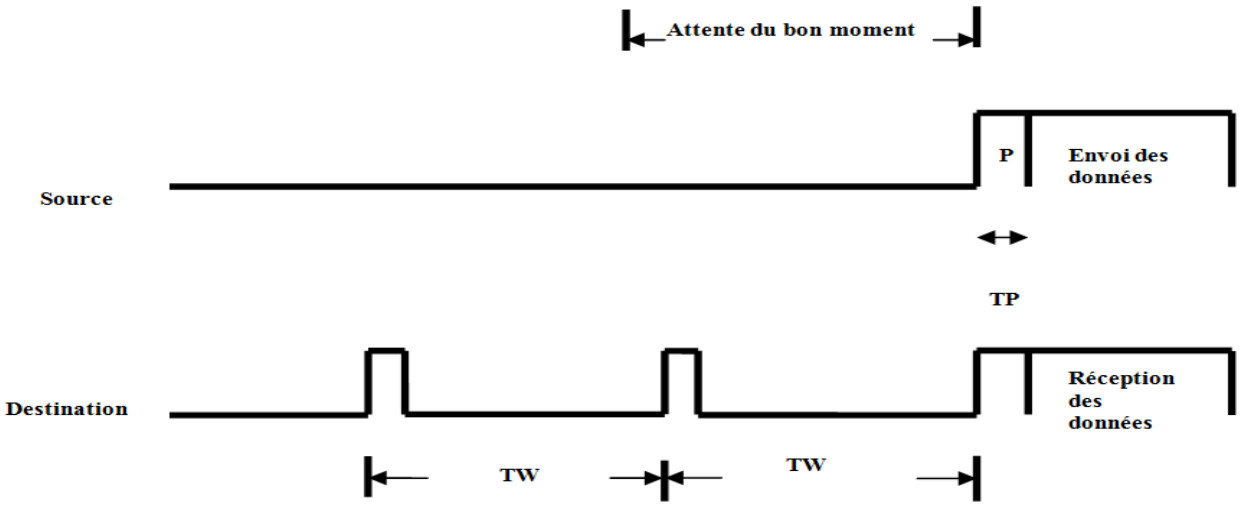
- Aucune solution n'est donnée pour le problème du nœud caché.
- BMAC minimise l'écoute au canal libre en augmentant le prix de la transmission et réception et de l'*overhead* **[11].**

II.14.2 F) WISEMAC

Le protocole WISEMAC est conçu pour le réseau de capteurs sans fil WISENET. Il utilise la technique de l'échantillonnage de préambule (*Preamble Sampling*). Cette technique consiste à échantillonner régulièrement le canal en écoutant sa radio pour une petite durée. Tous les nœuds capteurs dans le réseau échantillonnent le canal avec une même durée appelée TW. Si le canal est occupé, il continue à écouter jusqu'à ce qu'il reçoive un paquet ou que le canal redevient libre. Au niveau de l'émetteur, un préambule *wake-up* est transmis avant chaque message pour assurer que le récepteur sera réveillé quand la portion des données du message arrive.

Ce préambule introduit une consommation d'énergie causée par un *overhead*. Pour minimiser cet *overhead*, les nœuds peuvent avoir l'information sur le décalage entre le *scheduling* des échantillonnages de leurs voisins et les siens. En connaissant le *scheduling* de l'échantillonnage du récepteur, le nœud envoie les messages au bon moment avec un préambule *wake-up* de longueur minimale TP. Chaque nœud maintient une table à jour avec les différences de *schedule* d'échantillonnage de ses voisins directs. Pour avoir ces informations, la durée restante pour le prochain échantillonnage du préambule est incluse dans chaque paquet ACK.

Pour minimiser les collisions, WISEMAC utilise l'écoute de la porteuse sans persistance, avec un *backoff* choisi comme un entier aléatoire multiplié par le temps de changement d'état de l'unité de communication. Pour prévenir les collisions, un préambule de réservation du médium de durée aléatoire est ajouté avant le préambule *wake-up*. Les collisions causées par les nœuds cachés peuvent représenter une source importante de la perte d'énergie.



-Figure I-31 : Techniques de préambule dans WISEMAC (P *préambule*)-

Avantages

- WISEMAC évite l'*overmiting* en utilisant le préambule *wake-up*.
- WISEMAC ne nécessite pas une synchronisation pour que les noeuds dorment et se réveillent au même temps.
- Les résultas de simulation montrent que WISEMAC consomme moins d'énergie que SMAC et T-MAC.

Inconvénients

- Aucune solution n'est donnée pour le problème du noeud caché.
- WISEMAC minimise l'écoute au canal libre en augmentant le prix de la transmission et réception.

II.14-2 F) DW-LPL

DW-LPL (Dual Wake-up LPL) limite le problème de l'overhearing en employant la technique du LPL pour les messages diffusés uniquement. L'envoi de messages unicast utilise une technique de signalement par des trames spécifiques envoyées par les récepteurs pour informer leur entourage qu'ils sont prêts à recevoir des trames en unicast. Les instants et la périodicité d'envoi des messages de signalement sont indépendants d'un nœud à l'autre **[14]**.

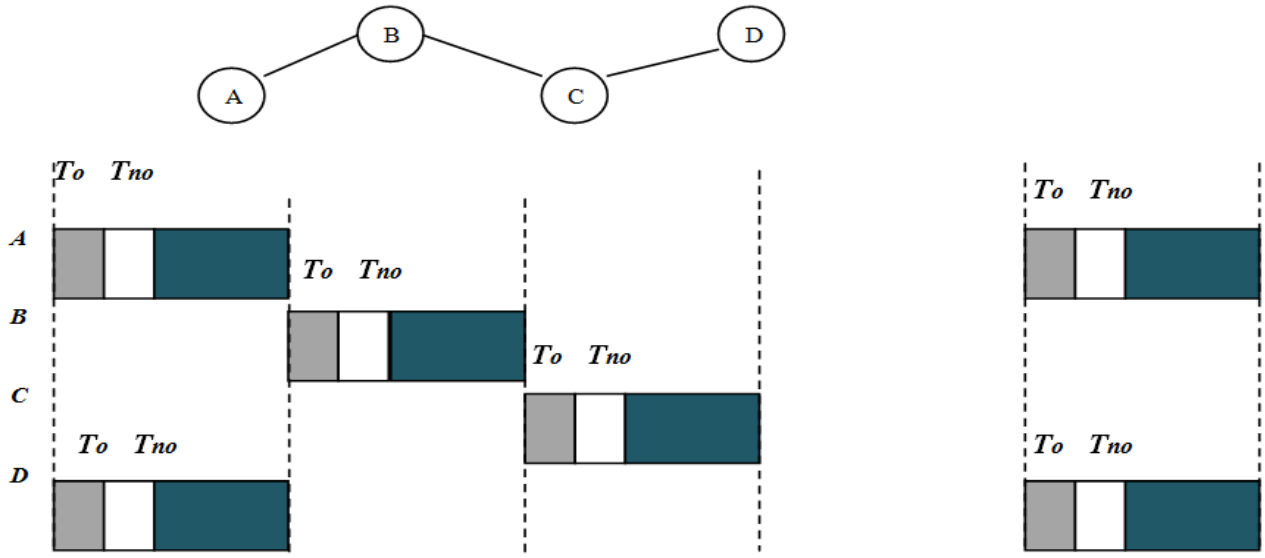
II.14-3 Protocoles hybrids

Comme discuté précédemment, les protocoles de la couche MAC ne peuvent pas toujours fournir un résultat optimal dans le terme d'efficacité d'énergie dans les WSNs ;
Les protocoles MAC hybrides tentent d'intégrer le contrôle des protocoles centralisés avec une flexibilité dans la distribution des protocoles.

II.14-3 A) ZMAC

Z-MAC (Zebra-MAC*)* **[82]** est un protocole hybride qui alterne des périodes de TDMA et CSMA/CA selon le nombre d'entités en concurrence en un instant donné. Z-MAC utilise un découpage temporel basé sur TDMA et gère les accès durant les slots avec le CSMA/CA de IEEE 802.11.

Une fois le réseau déployé, Z-MAC commence par une phase de découverte du voisinage à deux sauts suivie par une assignation de slots aux nœuds en utilisant DRAND (Distributed Ran- domized TDMA Scheduling For Wireless Adhoc Networks). DRAND est un protocole distribué qui assure qu'un slot de temps n'est pas assigné à deux nœuds situés à moins de trois sauts l'un de l'autre. La synchronisation nécessaire est obtenue à l'aide de deux protocoles utilisés de façon complémentaire. Pour accéder au médium, si le nœud est le propriétaire du slot courant, il attend un temps aléatoire plus petit qu'une valeur To puis effectue un CCA. Si le canal est libre, il émet. Sinon, il attend que le canal devienne libre et il recommence la même démarche. Si le slot actuel appartient à un voisin à deux sauts et si le nœud a reçu une indication de forte contention d'un de ses voisins à deux sauts, le nœud n'a pas le droit d'utiliser ce slot. Sinon, il attend un temps aléatoire compris entre To et Tno avant d'effectuer un CCA.



-Figure I-32 : Découpage temporel de Z-MAC. Il existe un temps pendant que les médias ne sont pas utilisés-

Avantages

- Z-MAC est facile et rapide à s'adapter aux conditions de trafic.
- Le mécanisme hybride de CSMA/CA et TDMA économise beaucoup d'énergie.
- Les collisions sont plus évitées dans Z-MAC que dans les protocoles basés sur CSMA/CA.
- La latence est plus petite que dans TDMA.

Inconvénients

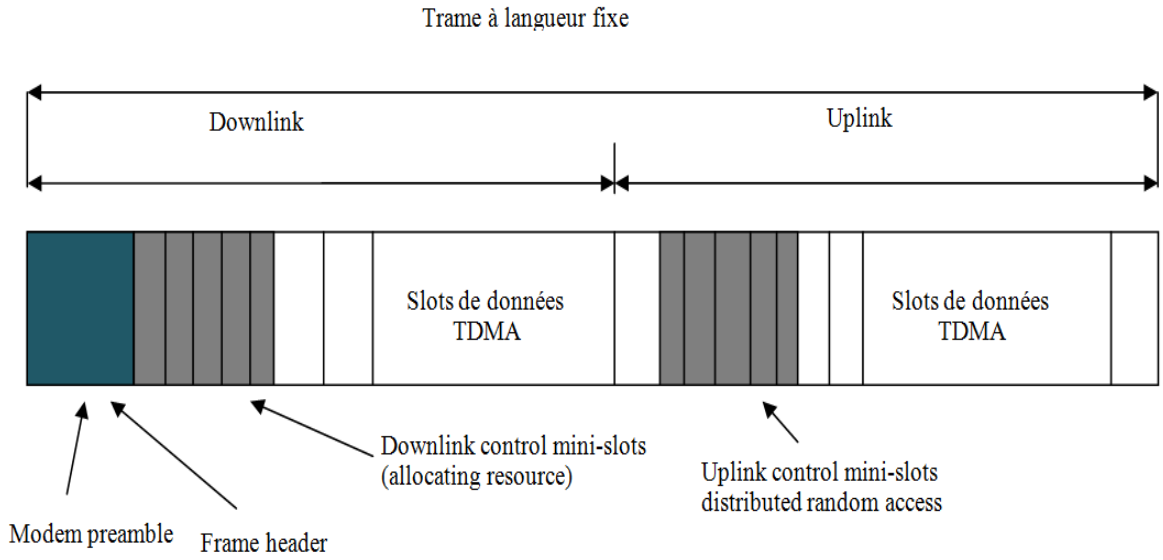
- La synchronisation des nœuds reste un problème à cause de la grande densité des nœuds dans les réseaux de capteurs.
- L'*overhead* est augmenté à cause du maintien du mécanisme d'accès TDMA.

II.14-3 B) DR-TDMA

Est proposé pour les réseaux ATM, et il a été adapté pour les WSNs, la figure suivante montre la structure de la trame DR-TDMA ; la langueur fixe de la trame est divisé en deux intervalles de temps UPLINK et DOWNLINK. Durant la phase de contrôle de l'intervalle UPLINK des collisions distribuées basé sur le schéma MAC (la priorité de fragmentation pseudo-Bayesion du protocole aloha) est utilisé pour les nœuds afin de transmettre des requêtes au temps réservé pour la prochaine trame

vers la station de base. Des données UPLINK sont transmises dans le mode TDMA basé sur un slot de temps fixé par la station de base dans l'intervalle de la procédure DOWNLINK. Dans cet intervalle, un protocole MAC centralisé est utilisé pour faire des transferts dans les slots réservés, et (même chose quand le message vient du puits vers les nœuds).

La réservation de ressources et les transferts sont ajustés dynamiquement. Les nœuds peuvent être en veille pendant qu'ils ne transmettent ou ne reçoivent rien. Cependant, le gaspillage d'énergie due à la collision idle sensing et l'overhering peuvent être réduis Des slots de taches effectués dynamiquement peuvent aussi fournir la flexibilité pour les WSNS.



-Figure I-33 : La trame DR-TDMA-

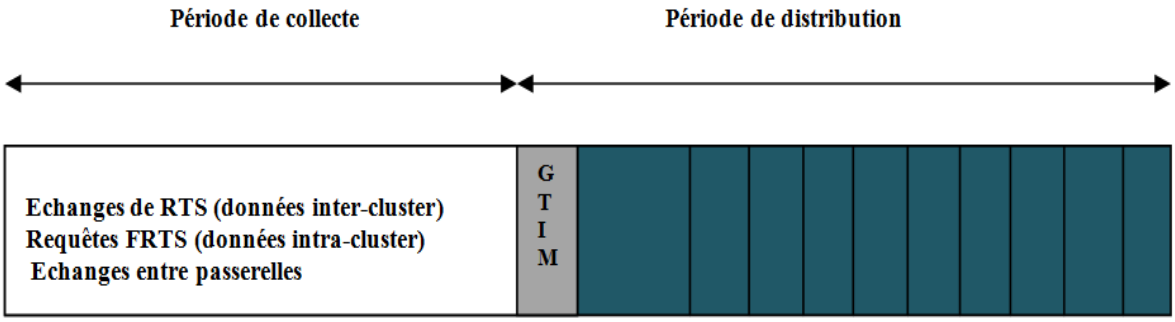
II.14-3 C) G-MAC

G-MAC (Gateway MAC) organise les échanges à l'intérieur d'un cluster. Dans ce domaine, un cluster est un ensemble de nœuds géré par une station appelée passerelle. G-MAC découpe le temps en deux périodes : une période de collecte où les échanges se font en CSMA/CA, et une période de distribution où les échanges se font en TDMA.

Durant la période de collecte, les nœuds envoient à leur passerelle deux types de trafic : les requêtes FRTS (Future Request To Send) pour réserver un slot de temps dans la période de distribution

pour échanger avec d'autres nœuds appartenant au même cluster, et le trafic inter- cluster destiné aux nœuds appartenant à des clusters différents.

Durant un slot, les échanges se font selon le séquencement RTS-CTS-données-acquittement. La période de distribution commence par une diffusion d'un message GTIM (Gateway Traffic Indication Message) qui contient les indications temporelles des deux périodes suivantes ainsi que le séquencement des échanges entre les nœuds du cluster qui ont réussi à envoyer une requête FRTS. Pendant la période de collecte et pendant les slots non réservés de la période de distribution la passerelle échange les données inter-cluster avec d'autres passerelles une fois la collecte terminée **[14].** GMAC élit périodiquement un nouveau nœud *gateway* pour distribuer équitablement l'énergie parmi les nœuds en utilisant l'algorithme RAVE (pour *Resource Adaptive Voluntary Election*) qui est basé sur le niveau d'énergie et les ressources mémoire disponibles



-Figure I-34 : Découpage temporel de G-MAC-

Avantages

- GMAC minimise l'écoute au canal libre plus que dans S-MAC et T-MAC.
- Le pourcentage de *mise en veille* est très élevé.
- Les deux périodes de GMAC fournissent un réseau à grande échelle, favorisent l'échange équitable de données et utilisent la bande passante d'une manière efficace.

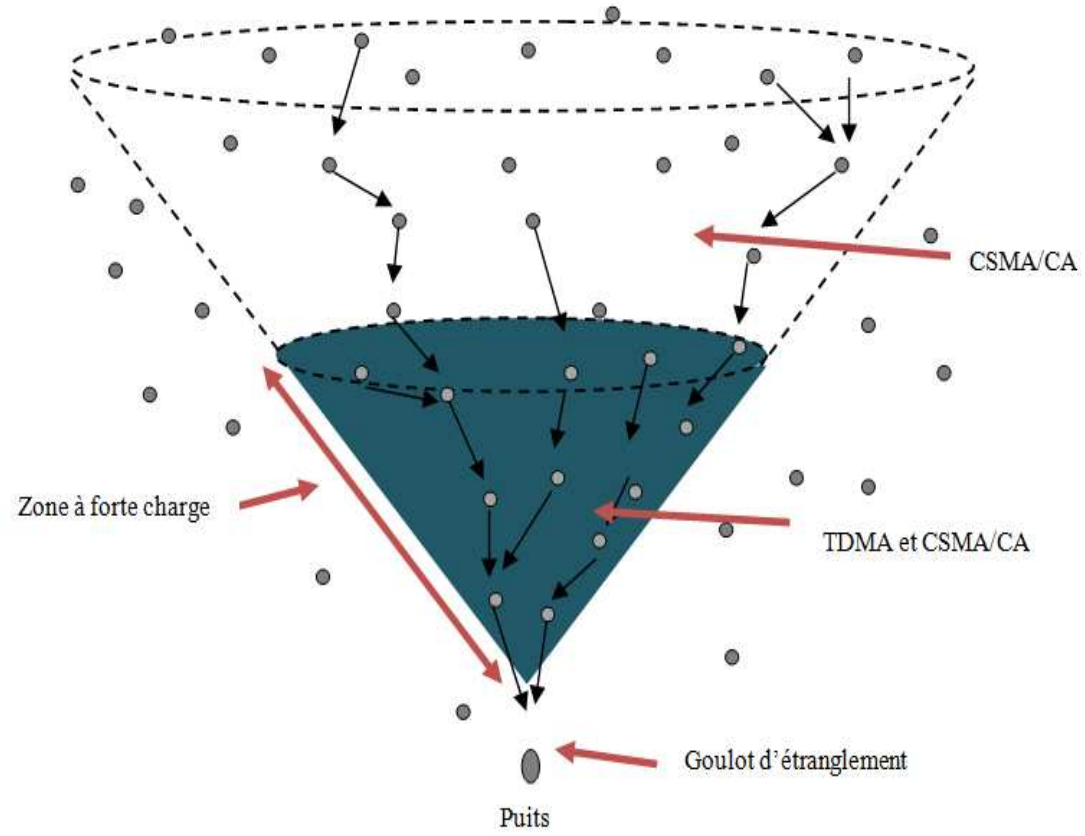
Inconvénients

- L'hypothèse du réseau à un seul saut et des nœuds qui sont tous dans une même portée de transmission est trop forte.

- Les nœuds doivent avoir des informations sur leurs voisins de même *cluster* pour différencier entre le trafic *inter-network* et *intra-network.*

II.14-3 D) Funneling-MAC

Funneling-MAC est un protocole MAC qui prend en compte le goulot d'étranglement dont souffrent la plupart des applications des réseaux de capteurs. Ce phénomène survient quand une station du réseau joue le rôle d'un puits de données vers lequel un ensemble de capteurs dirige son trafic. Cela est représenté sur la figure suivante sur laquelle est identifiée une zone à forte charge.

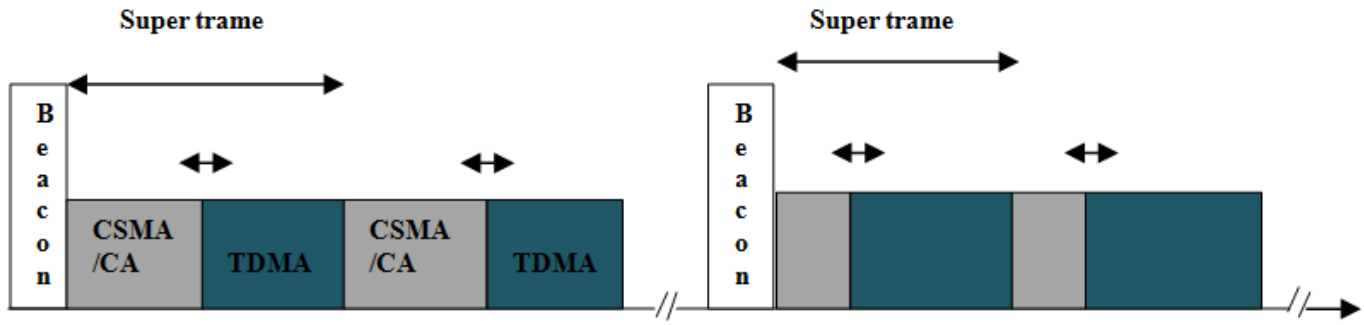


-Figure I-35 : Gestion du goulot d'étranglement dans Funneling-MAC [14] -

Funneling-MAC adopte une méthode d'accès en CSMA/CA dans l'ensemble du réseau durant un intervalle de temps suivi par un intervalle de temps durant lequel une méthode d'accès en TDMA est utilisée pour la zone à forte charge uniquement pour offrir plus de temps d'accès aux nœuds à proximité du puits. Ces deux intervalles de temps constituent une super trame. Ce découpage est représenté sur la **Figure I-36**.

La zone à forte charge est dimensionnée par une diffusion d'un beacon par le nœud puits. Les nœuds qui ne reçoivent pas ce beacon appliquent le CSMA/CA.

Le puits se base sur le chemin parcouru par les trames qu'il reçoit pour définir le séquencement et le dimensionnement des slots TDMA alloués aux nœuds de la zone à forte charge. Seuls les nœuds appartenant à la zone à forte charge mettent à jour le chemin parcouru par une trame. Pour éviter que les nœuds qui se trouvent au delà de la zone à forte charge interfèrent avec le découpage temporel du TDMA et le beacon transmit par le nœud puits, les nœuds de la zone.



-Figure I-36 : La trame Funneling-MAC-

II.14-3 IEEE 802.11

En 1999, IEEE a publié le standard 802.11 spécifiant la couche physique et MAC. L'équipement IEEE 802.11 opère dans les fréquences 2.4 GHz ou 5 GHz, et peut opérer dans un mode infrastructure et un mode *ad hoc*. Dans le premier, les nœuds communiquent à travers une entité centrale appelée point d'accès (AP pour *Access point*) en utilisant une fonction de point de coordination (PCF pour *Point Coordination Function*). Dans le mode *ad hoc*, les nœuds communiquent entre eux directement en utilisant la fonction DCF (pour *Distributed Coordination Function*).

PCF et DCF utilisent un mécanisme d'accès au canal similaire à CSMA/CA et utilise des accusés de réception pour la fiabilité.

Chaque nœud maintient un *backoff* qui permet de contrôler l'accès au canal. Quand un nœud veut transmettre, il se met à l'écoute du canal, si ce dernier est occupé alors il reportera sa transmission. Si le canal est libre pendant un temps spécifique appelé DIFS (pour *Distributed Inter Frame Space*), alors le nœud peut transmettre. La transmission de données est suivie par un ACK pour confirmer le succès de la réception. Le standard a défini un temps SIFS (pour *Short Inter Frame Space*) entre chaque

paquet (entre RTS, CTS, DATA et ACK). La durée de SIFS est plus courte que celle de DIFS. Cela permet de donner la priorité à la communication en cours. Dans le cas d'échec de transmission ou après la transmission avec succès, un nœud adopte un mécanisme de *Binary Exponential Backoff* (BEB) pour choisir un *backoff* aléatoire. Ce nombre aléatoire est tiré dans un intervalle [1, *CW*] où *CW* présente la fenêtre de contention (CW pour *Collision Window*).

Pour répondre au problème de nœud caché, IEEE 802.11 définit l'écoute à la porteuse virtuelle. Lorsqu'un nœud transmet, il commence avec un RTS qui inclut la durée de la séquence DATA/ACK qui vient. La destination répond avec un CTS qui ajoutera la même durée. Ceci permet aux nœuds voisins qui ont entendu les paquets de contrôle de définir leur *Network Allocation Vector* (NAV) et de retarder leur transmission jusqu'à ce que le NAV expire. Pour sauvegarder l'énergie, la radio peut être éteinte pendant la durée du NAV.

II.14-4 IEEE 802.15.4 ++(ZIGbee)

La norme IEEE 802.15.4 a été spécialement définie en fonction des caractéristiques des réseaux de capteurs, un faible débit et une faible consommation. Elle est caractérisée par une portée maximum de quelques centaines de mètres et un débit faible (250kbit/s max). Elle décrit le fonctionnement de la couche physique et de la couche MAC.

Dans cette dernière, deux topologies sont supportées, la topologie en étoile et la topologie point à point. Dans la première topologie, les communications s'établissent directement entre un nœud central (coordinateur) et des capteurs. Le coordinateur est le nœud qui initie et gère les communications dans le réseau. La topologie point à point permet à un nœud du réseau de communiquer avec n'importe quel autre nœud ce qui permet de réaliser des réseaux ayant une architecture beaucoup plus complexe.

I.15 Conclusion

Ce chapitre est un récapitulatif sur les points essentiels qui définissent l'état de l'art des réseaux de capteurs « leurs caractéristiques de base, leurs contraintes et leurs domaines d'applications ». Nous avons ensuite présenté quelques protocoles de couche MAC et de routage les plus connus.

Contrairement aux réseaux traditionnels qui se préoccupent de garantir une bonne qualité de service, les réseaux de capteurs doivent, en plus, prendre en compte la conservation d'énergie. Ils doivent intégrer des mécanismes qui permettent aux utilisateurs de prolonger la durée de vie du réseau en

entier, car chaque nœud est alimenté par une source d'énergie limitée et généralement irremplaçable. Dans un nœud capteur, l'énergie est consommée en assurant les fonctions suivantes : la capture, le calcul (*traitement*) et la communication. Cette dernière représente une grande portion de l'énergie totale consommée. De ce fait, la communauté de recherche est en train de développer et de raffiner plusieurs techniques de conservation d'énergie que nous allons développer dans le chapitre suivant.

Sommaire

II.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.
Elle 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 [**29**]

Différents mécanismes de dissipation d'énergie existent grâce à diverses recherches concernant ce domaine, depuis l'avènement des réseaux de capteurs sans fil des problématiques sur l'alimentation et la gestion d'énergie en vu le jour, dans les travaux de recherche en cours nous avons rencontré plusieurs études sur les techniques de conservation d'énergie suivies. Elles se classifient en trois catégories **(1)** techniques de duty-cycling, **(2)** techniques orienté données, **(3)** techniques de mobilité. Résultats de ces travaux ont aboutis à des protocoles économes en énergie appliqués sur différentes couches,

Nous allons dans ce chapitre parler sur les contraintes de consommation d'énergie, les diverses réalisations en se basant sur plusieurs techniques, l'étude de et quelques protocoles économes en énergie appliquées sur différentes couches.

II.2 La durée de vie d'un réseau

Quand le réseau de capteur est en vie il peut accomplir sa mission, mais cette durée de vie est toujours limitée, donc la fonctionnalité du réseau est limitée.
La vie d'un nœud d'un réseau de capteur dépend des deux facteurs :

- De l'énergie qu'il consomme en fonction du temps
- De la qualité d'énergie dont il dispose

La définition de la durée de vie d'un réseau est :

1- Le réseau est en vie jusqu'à ce que le premier nœud épuise son énergie [**30**].

2- dés que le premier *clusterhead (tête de grappe)* épuise toute son énergie la vie du réseau s'achève [**31**].

3- La durée jusqu'à ce qu'il reste au plus une certaine fraction ***B*** de nœuds survivants dans le réseau [**32]**.

4- Demi-vie du réseau : la durée jusqu'à ce que 50% des nœuds épuisent leurs batteries et s'arrêtent de fonctionner [**30**].

5- La durée jusqu'à ce que tous les capteurs épuisent leur énergie [**33**].

6- La durée jusqu'à ce que le réseau soit partitionné : apparition de la première division du réseau en deux (ou plus). Cela peut correspondre aussi à la mort du premier nœud (si celui-ci tient une position centrale) ou plus tard si la topologie du réseau est plus robuste [**30**].

7- k-couverture : la durée pendant laquelle la zone d'intérêt est couverte par au moins k nœuds [**34**].

8- 100%-couverture

- La durée pendant laquelle chaque cible est couverte par au moins un noeud. [**35**]

- La durée pendant laquelle l'ensemble de la zone est couverte par au moins un nœud [**36**]

9- Alpha-couverture

- La durée cumulée, au bout de laquelle au moins une portion *Alpha* de la région est couverte par au moins un nœud [**37**]
- La durée pendant laquelle la couverture tombe en-dessous d'un seuil prédéfini [**38**].
- La durée de fonctionnement continu du système avant que la couverture ou la proportion de paquets reçus (PDR pour Packet Delivery Ratio) tombent en-dessous d'un seuil prédéfini [**39**].

10- La durée pendant laquelle un pourcentage donné de nœuds possèdent un chemin vers la Station de Base [**39**].

11- L'espérance de l'intervalle complet pendant lequel la probabilité de garantir simultanément une connectivité et une k-couverture est au moins Alpha *[**40**]*

12- La durée jusqu'à la perte de la connectivité ou de la couverture**.***[**41**]*

13- La durée jusqu'à ce que le réseau ne fournisse plus un taux acceptable de détection d'événements**.***[**33**]*

14- La durée pendant laquelle le réseau satisfait continuellement les besoins de l'application.*[**42**]*

La durée de vie d'un réseau est le temps de son utilisation dans l'application qui couvre et la différence entre toutes ces définitions n'est que résultat de différence :

- de l'environnement d'utilisation de ce réseau.
- de la topologie utilisée.
- de la criticité du contexte (applications temps réel).
- du type de nœuds.
- de la densité et la couverture des zones.

Le fonctionnement du réseau est proportionnel à sa durée de vie, c'est-à-dire plus la durée de vie est langue, meilleur est le fonctionnement du réseau.

Le calcul de durée de vie se fait on utilisant des probabilités sur le temps de survie d'un capteur, ensuite du réseau, plusieurs acteurs interviennent dans la variation de cette consommation, comme la densité des nœuds dans une zone ou bien le routage de ces derniers, ce qui donne parfois une corrélation positive ou négative.

Toutes ces métriques peuvent bien sûr être évaluées avec un ensemble d'hypothèses sur les caractéristiques d'un nœud donné en terme de *consommation d'énergie*, sur la « charge » courante que le réseau est appelé à traiter (par exemple, où et quand les événements se produisent) et aussi sur le comportement du *canal radio*.

II.3 La consommation d'énergie

Comme les noeuds capteurs sont des composant micro-électroniques, ils ne peuvent être équipés que par des sources limitées d'énergie (<0.5 Ampère-heure, 1.2 V). De plus, dans certaines

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

Sachant que les réseaux de capteurs sont basés sur la communication multi-sauts, chaque nœud joue à la fois un rôle d'initiateur de données et de routeur également, le mal fonctionnement d'un certain nombre de nœud entraîne un changement significatif sur la topologie globale du réseau, et peut nécessiter un routage de paquets différent et une réorganisation totale du réseau.

C'est pour cela que le facteur de consommation d'énergie est d'une importance primordiale dans les réseaux de capteurs. La majorité des travaux de recherche menés actuellement se concentrent sur ce problème afin de concevoir des algorithmes et protocoles spécifiques à ce genre de réseau qui consomment le minimum d'énergie.

En effet, dans les réseaux ad hoc classiques, la consommation d'énergie est un facteur important mais ne constitue pas la première considération pour les concepteurs, car les batteries sont supposées toujours remplaçable par l'utilisateur, les chercheurs ont cependant concentré leurs efforts sur les facteurs de qualité de service dans ce type de réseau, tel que le débit de transmission et la tolérance aux panne.

Par contre, Dans les réseaux de capteurs, l'efficacité en consommation d'énergie représente une métrique de performance significative, qui influence directement sur la durée de vie du réseau en entier. Pour cela, les concepteurs peuvent au moment du développement de protocoles négliger les autres métriques de performance telle que la durée de transmission et le débit, au détriment du facteur de consommation d'énergie.

La consommation d'énergie s'effectue essentiellement au niveau de la capture du traitement (microprocesseur) et de la transmission des données *(module radio*). Les études concernant les réseaux de capteurs ont montré que la phase de communication est la plus consommatrice en énergie. Vu que l'énergie des capteurs n'est pas renouvelable, les opérations de capture, de traitement et de routage doivent être correctement évaluées afin d'éviter l'épuisement prématuré de cette énergie et de prolonger la durée de vie du réseau **[5]**. Les techniques de conservation d'énergie lors de la communication et le calcul sont donc d'une importance majeure dans un réseau de capteurs sans fil**.**

II.3-1 Formes de dissipation d'énergie dans les réseaux de capteurs

Les nœuds-capteurs sont alimentés principalement par des batteries. Ils doivent donc fonctionner avec un bilan énergétique frugal. En outre, ils doivent le plus souvent avoir une durée de vie de l'ordre de plusieurs mois, voire de quelques années, puisque le remplacement des batteries n'est pas une option envisageable pour des réseaux avec des milliers de nœuds [**3**].

En effet la conservation d'énergie peut être réalisée, avec plusieurs méthodes qui nous mènent à des solutions efficaces, il est extrêmement important de faire d'abord une analyse des différents facteurs provoquant la dissipation de l'énergie d'un nœud-capteur.

II.3-1 A) La radio

La radio opère dans quatre modes de fonctionnement : émission, réception, « idle », et sommeil. Une observation importante dans le cas de la plupart des radios est que le mode « idle » induit une consommation d'énergie significative, presque égale à la consommation en mode réception. Ainsi, il est plus judicieux d'éteindre complètement la radio plutôt que de passer en mode "idle" quand l'on a ni à émettre ni à recevoir de données.

Un autre facteur déterminant est que, le passage de la radio d'un mode à un autre engendre une dissipation d'énergie importante due à l'activité des circuits électroniques. Par exemple, quand la radio passe du mode sommeil au mode émission pour envoyer un paquet, une importante quantité d'énergie est consommée pour le démarrage de l'émetteur lui-même. Un autre point important est que les données des constructeurs sous-estiment assez régulièrement ces différentes consommations, en particulier concernant la consommation dans le mode « idle ».

L'émission d'un signal est caractérisée par sa puissance. Quand la puissance d'émission est élevée, le signal aura une grande portée et l'énergie consommée sera plus élevée.

I.13-1 B) Le capteur (détecteur)

Cette phase inclut l'échantillonnage des signaux physiques, le traitement du signal et la conversion analogique/numérique. 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. [**5**]

Étant donné la diversité des capteurs, il n'y a pas de valeurs typiques de l'énergie consommée. En revanche, les capteurs passifs (température, sismiques, ...) consomment le plus souvent peu d'énergie par rapport aux autres composants du nœud-capteur. Notons, les capteurs actifs tels que les sonars, les capteurs d'images, etc. peuvent consommer beaucoup d'énergie.[**3**]

II.3-1 C) Le MCU (unité du microcontrôleur « Micro Controller Unit »)

Les MCUs possèdent divers modes de fonctionnement : actif, « *idle* », et sommeil, à des fins de gestion d'énergie. Chaque mode est caractérisé par une quantité différente de consommation d'énergie. Par exemple, le **MSP430** consomme 3 *m*W en mode actif, 98 *m*W dans le mode « *idle* » et seulement 15 *m*W dans le mode sommeil. Toutefois, la transition entre les modes de fonctionnement implique un surplus d'énergie et de latence. Ainsi, les niveaux de consommation d'énergie des différents modes, les coûts de transition entre les modes mais encore le temps passé par le **MCU** dans chaque mode ont une incidence importante sur la consommation totale d'énergie d'un nœud-capteur.[**3**]

L'énergie de commutation (*actif*) 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 (*idle*) correspond à l'énergie consommée lorsque l'unité de calcul n'effectue aucun traitement. [**11**]

C'est parce qu'il existe d'autres formes de dissipation d'énergie comme : *les lectures et les écritures mémoire l'auto-décharge de la batterie (la batterie perd sa capacité au fil du temps)* qu'il est difficile d'apporter une étude quantitative et comparative précise de la consommation de chaque composant d'un nœud-capteur en raison du grand nombre de plates-formes commerciales existantes.

Cependant, des expérimentations ont montré que c'est la transmission de données qui est la plus consommatrice en énergie. Le coût d'une transmission d'un bit d'information est approximativement le même que le coût nécessaire au calcul d'un millier d'opérations. La consommation du module de détection dépend du type spécifique du nœud-capteur.

II.3-2 Facteurs intervenant dans la consommation d'énergie

II.3-2 A) Etat du module radio

Un capteur consommant beaucoup d'énergie est souvent alimenté par sa propre source énergétique. Nous considérons ici que la couche **MAC** est concernée uniquement par l'utilisation du module radio et du microprocesseur. Dans un réseau de capteurs, la portée est de l'ordre d'une dizaine de mètres dans un milieu clos avec une puissance d'´emission de 0 dBm (1 mW). Avec ce niveau de

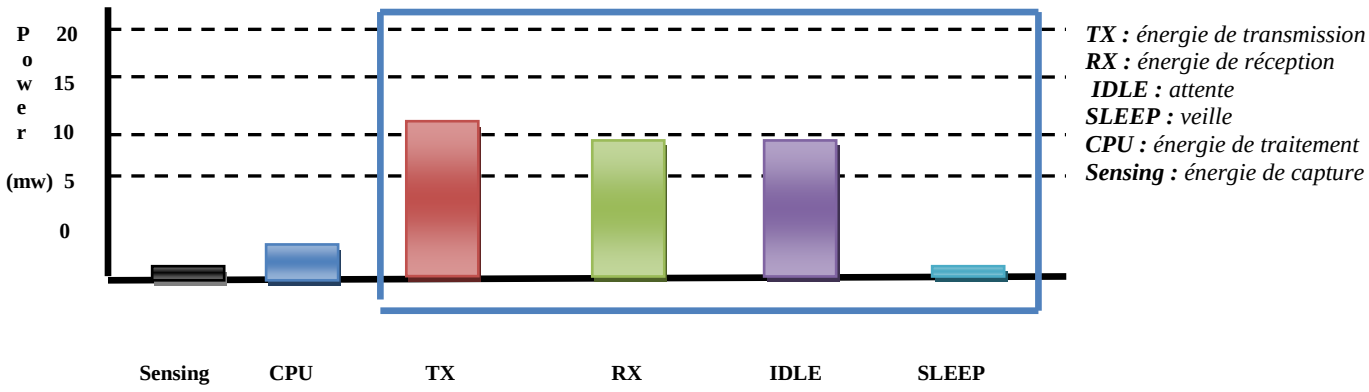
puissance d'émission, l'énergie consommée pour la réception et celle consommée pour l'émission sont quasiment égales. Le tableau suivant

montre un exemple d'énergie consommée pour chaque état du module radio d'une carte B2400ZB-tiny[**21**]

Etat	Energie consommé
Emission	26 *m*A
Réception on écoute	29 *m*A
Veille	15 µA
Sommeil	3 µA

-Tableau I-2 : Energie consommée par état par le composant radio d'une carte B2400ZB-tiny-

Le passage de l'état actif à l'état sommeil peut avoir comme conséquence une consommation d'énergie plus importante que de laisser le module radio en mode actif. Ceci est dû à la puissance nécessaire pour la mise sous tension du module radio. Cette énergie est appelé l'énergie de transition. Il est ainsi souhaitable d'arrêter complètement la radio plutôt que de transiter dans le mode sommeil. Le changement d'état du module radio doit être géré par un protocole de la couche MAC



-Figure II-1 : énergie consommée par les sous-systèmes d'un capteur-[5]

II.3-2 B) Accès au medium de transmission

La surconsommation d'énergie est toute consommation inutile qui peut être évité afin de conserver l'énergie d'un nœud-capteur. Les sources de cette surconsommation sont nombreuses, elles peuvent être engendrées lors de la détection lorsque celle-ci est mal gérée (par exemple par une fréquence d'échantillonnage mal contrôlée).

Un protocole MAC économe en énergie essaie d'utiliser le moins souvent possible le module radio, vu que c'est la partie du capteur qui consomme le plus d'énergie. En effet les modules radio peuvent avoir plusieurs niveaux de consommation quand ils ne sont pas en mode émission ou réception, moins le nœud consomme moins il est réactif, c'est pour cela que les différents états de sommeil existent pour assurer une flexibilité selon le degré de réactivité demandé par la couche MAC.

La surconsommation concerne également la partie communication. En effet, cette dernière est sujette à plusieurs phénomènes qui surconsomment de l'énergie surtout au niveau MAC où se déroule le contrôle d'accès au support sans fil. Certains de ces phénomènes sont les causes majeures de la perte d'énergie et ont été recensés dans :

II.3-2 B) a. La taille des paquets

La taille des messages échangés dans le réseau a un effet sur la consommation d'énergie des noeuds 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) générés augmente l'*overhead*. Dans le cas contraire, une grande puissance de transmission est nécessaire pour des paquets de grande taille. [**11**]

Il y a aussi L'overhead des paquets de contrôle **(**l'envoi, la réception, l'écoute) de ces derniers consomme de l'énergie. Comme les paquets de contrôle ne transportent pas directement des données, ils réduisent également le débit utile effectif.

II.3-2 B) b. Les collisions

Les collisions sont à la fois une source de dégradation des performances du réseau et de perte d'énergie. Les pertes de trames à cause des collisions forcent les nœuds à retransmettre le même paquet plusieurs fois et donc à rester actif pour le répéter et vérifier qu'il est bien reçu par la destination.

A noter que les retransmissions ne se font que pour les trames envoyées en mode unicast (à un seul destinataire) et avec une demande d'acquittement.[**21**] Quand deux trames sont émises en même temps et se heurtent, elles deviennent inexploitables et doivent être abandonnées. Les retransmettre par la suite, consomme de l'énergie. Tous les protocoles MAC essayent à leur manière d'éviter les collisions. Les collisions concernent plutôt les protocoles MAC avec contention.

II.3-2 B) c. L'écoute active

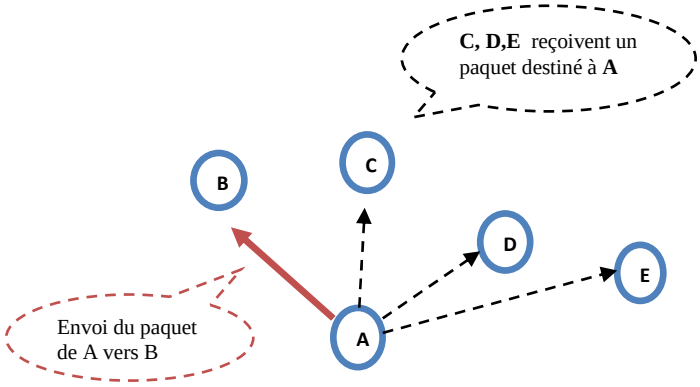
L'écoute active (*idle listening*) du canal pour une éventuelle réception de paquet qui ne sera pas reçu peut engendrer une perte importante de la capacité des nœuds en énergie.

Pour éviter ce problème, il faut basculer les nœuds dans le mode sommeil le plus longtemps possible.[**11**] Mais le coût de la transition entre les modes consomme également de l'énergie, la fréquence de cette transition doit alors rester raisonnable[**3**] .

II.3-2 B) d. L'écoute abusive (overhearing)

Le phénomène de surécoute (*overhearing*) se produit quand un nœud reçoit des paquets qui ne lui sont pas destinés (figure). La surécoute **Figure II-2** conduit à une perte d'énergie additionnelle à cause de l'implication des autres capteurs dans la réception des données.[**11**]

Le coût de l'écoute abusive peut être un facteur dominant de la perte d'énergie quand la charge de trafic est élevée et la densité des nœuds grande, particulièrement dans les réseaux mostly-on.[**3**]



- Figure II-2 : La surécoute dans une transmission-

II.3-2 B) e. La surémission (Envois infructueux)

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. [**11**]

Par exemple quand le nœud récepteur est en mode sommeil (ou hors de portée). Le nœud émetteur est en attente d'un acquittement, et il retransmet donc la même trame plusieurs fois. Il consomme de l'´energie en le faisant du fait qu'il soit resté en mode transmission et en mode réception pour l'´eventuel acquittement. [**21**]

II.3-2 B) f. 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.

II.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 $1/d^n$, où d est la distance entre l'émetteur et le récepteur, n un exposant de perte d'un chemin (Exemple : n=2 dans le vide, de 4 à 6 dans un immeuble). [**11**]

I I.3-4 Routage des données

Le routage dans les RCSFs consiste à acheminer les paquets de la source vers la destination (station de base). En effet, le principal objectif du routage est de trouver un chemin optimal minimisant la consommation d'énergie augmentant ainsi la durée de vie du réseau. De ce fait, les protocoles élaborés doivent assurer une consommation minimale d'énergie tout en maintenant le bon fonctionnement du réseau et sans dégrader ses performances [**5**]. Notant qu'une mauvaise politique de routage peut avoir des conséquences graves sur la durée de vie du réseau.

II.4 Evaluation de la consommation d'énergie dans un réseau de capteurs

Pour évaluer l'énergie consommée, il suffit de connaître le nombre de transmissions et de réceptions dans le réseau de capteurs.

Soient **NTx(t)** le nombre de transmissions et **NRx(t)** le nombre de réceptions à l'instant ***t***, et soient ETx(t) l'énergie consommée par les transmissions et **ERx(t)** l'énergie consommée par les réceptions, l'énergie totale **E(t)** à l'instant ***t*** est donc :

$$E(t) = ETx(t) + ERx(t)$$

Supposons que ETr et ERe représentent respectivement l'énergie nécessaire à la transmission et à la réception d'un paquet de taille *k* bits, et que tous les paquets ont la même taille. D'où :

$$E(t) = NTx(t) * ETr + NRx(t) * ERe. \text{ [15]}$$

II.5 Mécanismes de conservation de l'énergie

Nous avons déjà souligné que le module de communication est la partie la plus consommatrice en énergie dans un nœud capteur. En effet cette caractéristique conjuguée à l'objectif de maximisation de la durée de vie du réseau a suscité de nombreux travaux de recherche. Voila quelques mécanismes de base :

II.5-1 Mode d'économie d'énergie

Ce mode est possible quelle que soit la couche MAC adoptée. Cela consiste à éteindre le module de communication dès que possible. Par exemple, des protocoles

MAC fondés sur la méthode TDMA *(Time Division Multiple Access*) offrent une solution implicite puisqu'un nœud n'échange des messages que dans les intervalles de temps qui lui sont attribués [**3**]. Pour bénéficier de cette technique le capteur se met en veille pendant les autres slots, et il faut tout de même s'assurer que le gain d'énergie obtenu en mettant en veille le module radio ne soit pas inférieur au surcoût engendré par le redémarrage de ce dernier.

II.5-2 Traitement local

L'idée de cette technique est que la source peut se censurer. Ainsi une programmation événementielle semble bien adaptée aux réseaux de capteurs. Seuls les changements significatifs de

l'environnement devraient provoquer un envoi de paquets dans le réseau. Dans le même état d'esprit, une grande collaboration est attendue entre les capteurs d'une même région en raison de leur forte densité et dans la mesure où les observations ne varient presque pas entre des voisins très proches. Ainsi les données pourront être confrontées localement et agrégées au sein d'un seul et unique message. Cette stratégie de traitement local permet de réduire sensiblement le trafic.

II.5-3 Organisation des échanges

Ce procédé revient à limiter les problèmes de retransmission dus aux collisions. La solution extrême consiste à utiliser la technique d'accès au medium TDMA[**22**]

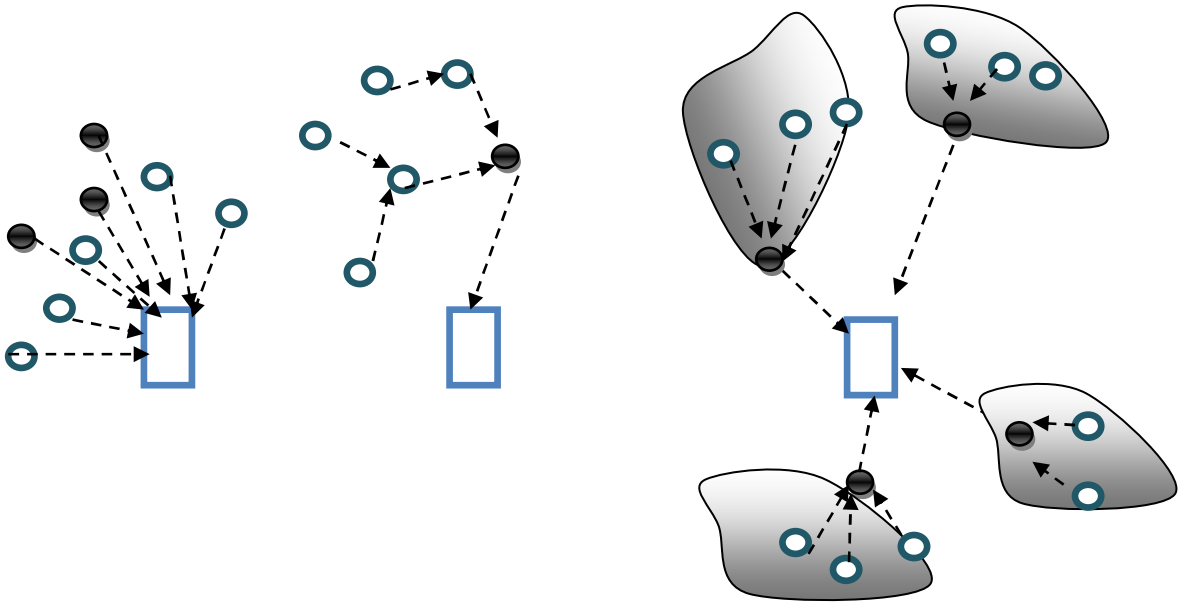
Les collisions sont ainsi fortement réduites. Cette solution présente l'inconvénient d'être peu flexible et de demander une synchronisation fine des capteurs. Des solutions intermédiaires ont vu le jour, par exemple S-MAC *(Sensor MAC)* qui est une méthode d'accès au canal de type CSMA-CA avec le mécanisme RTS/CTS *(Request to Send, Clear to Send)* qui permet d'éviter les collisions et le problème de la station cachée.

La principale innovation, apportée par ce protocole, est d'avoir *un mécanisme de mise en veille* distribué sur chaque nœud du réseau dans le but de réduire la consommation d'énergie. La principale difficulté de S-MAC est également de synchroniser les nœuds entre eux pour que la communication soit toujours possible.[**3**]

II.5-4 Limitation des accusés de réception

Répartition de la consommation d'énergie : la formation de « *clusters* » permet d'envisager des réseaux comportant un très grand nombre de capteurs. Elle favorise une meilleure répartition de la consommation d'énergie. En effet, dans le cas d'une transmission directe vers l'observateur **(I),** les capteurs éloignés vont plus rapidement manquer d'énergie et les autres nœuds peuvent être sujets au phénomène d'*overhearing* dans le cas des réseaux « *Mostly-On* ». Au contraire, dans le cas d'une transmission par saut **(II),** les nœuds proches de l'observateur vont être vite en rupture de batterie car ils seront plus sollicités pour relayer les messages des autres. La solution consiste à hiérarchiser les échanges en divisant la zone d'observation en clusters **(III)** Un « *clusterhead* » est élu pour chaque cluster. Il s'occupe de récupérer les informations auprès des capteurs de son cluster et de les transmettre directement à l'observateur. En changeant régulièrement de *clusterhead*, on obtient un réseau dans lequel aucun capteur n'est prédisposé à arriver en rupture de batterie avant les autres.

Mettre en place des clusters va également permettre de cloisonner le réseau et ceci dans l'objectif de réduire les interférences. On améliore ainsi la qualité du lien radio et par conséquent, on limite les retransmissions liées aux reprises sur erreur. L'exemple phare d'une solution avec des clusters est le protocole LEACH [**23**]

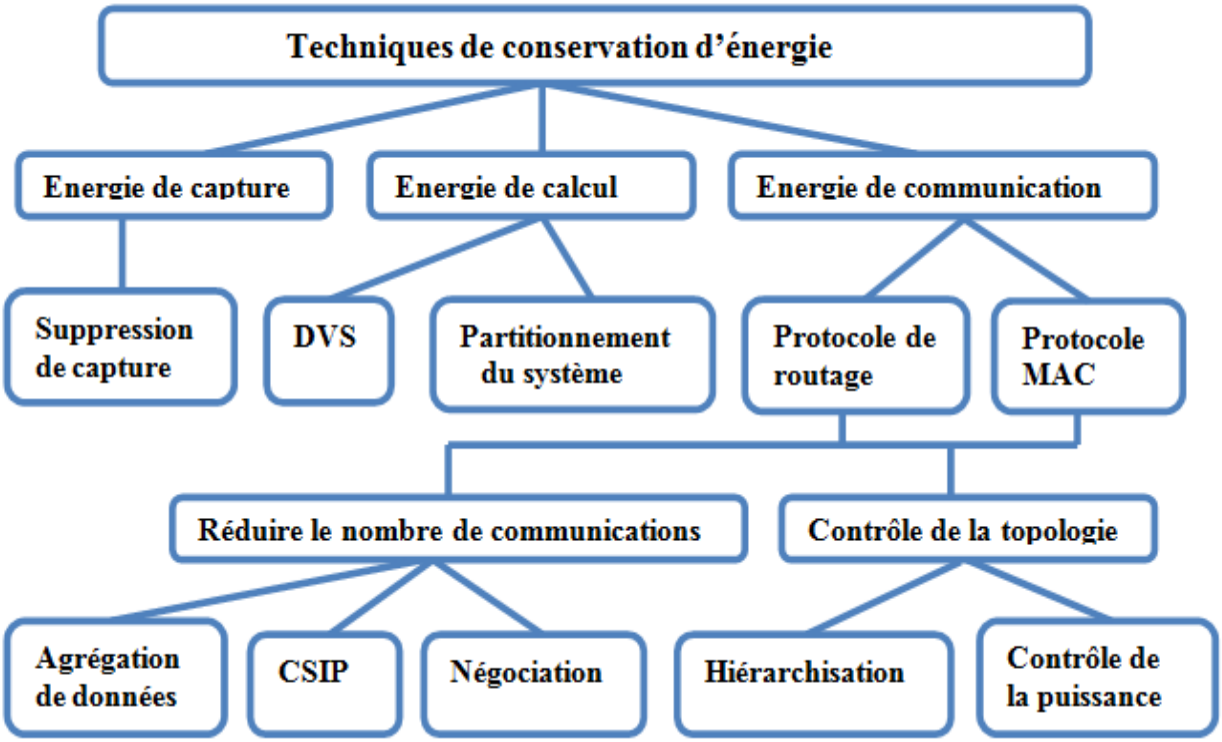


-Figure II-3 :l'hiérarchisation des réseaux de capteurs -

Par ailleurs, il existe dans la littérature d'autres mécanismes de conservation d'énergie, telles les techniques de compression, d'agrégation et de fusion de données, d'autres techniques de routage etc. Le paragraphe suivant illustre les différents niveaux de conservation d'énergie et donne quelques solutions pour accomplir cette partie.

II.5-5 Schéma de classification des techniques de minimisation de la consommation d'énergie

Après avoir fait une description des principales causes de consommation d'énergie dans les RCSF, nous allons présenter dans cette section (**figure II-4**) une vue globale de différentes techniques utilisées pour minimiser cette consommation.



- Figure II-4 : Techniques de conservation d'énergie - *[11]*

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*), qui 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, qui 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.

c- La minimisation de la consommation d'énergie pendant la communication est étroitement liée aux protocoles développés pour la couche réseau et la 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 plusieurs domaines:

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 pour 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).

- 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). Ainsi, il peut conserver l'énergie à deux niveaux : explicitement par l'application de puissances faibles d'émissions et implicitement en réduisant la contention avec d'autres nœuds transmetteurs. Le module de contrôle de la puissance est souvent intégré dans les protocoles soit de la couche réseau soit de la couche MAC.
- 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 chefs de groupe (*cluster head*) et des nœuds membres.
- Une autre technique a été proposée, elle profite de la densité élevée des capteurs déployés pour se permettre d'endormir certains d'entre eux, afin que tous les capteurs ne soient pas actifs en même temps et c'est la technique que nous allons analyser dans les prochains chapitres.

II.6 Les techniques de conservation d'énergie existantes

Pour simplifier la classification des techniques suivis pour la gestion d'énergie dans les réseaux de capteurs nous allons motionner les trois critères de classification :

- Quand on transmet et on traite un ***paquet de données de taille X*** on consomme une somme d'énergie considérable; donc les données transmises forment une problématique de consommation.

- On peut utiliser des ***dispositifs mobiles*** (capteurs) pour recevoir les informations capter par les nœuds cela permet de réduire la consommation d'énergie du à l'acheminement de données (routage).
- Un nœud capteur passe la plupart de son temps à attendre un événement, cela signifie que sa ***radio*** en ***mode Idle (écoute active)*** reste active sans recevoir, ou même recevoir des données inutiles ***(sur-écoute)*** ; ce qui consomme de l'énergie.

De ce fait la classification des techniques se base sur ces trois points : mobilité ; données et état de la radio.

II.6-1 Techniques orientées données

Puisque l'objectif principal d'un réseau de capteur est de capter des informations, des événements, des mesures, des données ; donc effectivement l'énergie consommé est due au captage et transmission de ces données. Nous avons donc dans deux aspects : *le matériel (réduire la quantité d'échationnage)* et la *redondance des données (échantillons inutiles).*

- **Des échantillons inutiles :** les données échantillonnées ont souvent de fortes corrélations spatiales et/ou temporelle, il est donc inutile de communiquer les informations redondantes à la Station de Base. Un échantillonnage inutile implique une consommation d'énergie à son tour inutile. En effet, même si le coût de l'échantillonnage est négligeable, cela induit aussi des communications tout au long du chemin qu'emprunte le message.

- **La consommation électrique du module de détection :** réduire la communication ne suffit pas lorsque le capteur est lui-même très consommateur. Des techniques orientées données sont conçues pour réduire la quantité d'échantillonnage de données en garantissant un niveau de précision acceptable dans la détection pour l'application.

Pour réduire la consommation d'énergie dans ce domaine on utilise la réduction de données :

1) Par agrégation de données.
2) Par compression de données

- **Acquisition de données efficace en énergie :** l'énergie consommée par le module de détection peut être plus grande que l'énergie consommée par le module radio ou autres à cause des facteurs suivants :

1) *transducteur gourmand en énergie.*
2) *convertisseur A/D gourmands*
3) *capteurs actifs*
4) *temps d'acquisition long*

La classification des approches d'acquisition de données efficaces en énergie présentée comme suit :

(1) Comme les échantillons mesurés peuvent être corrélés, les techniques d'échantillonnage adaptatif exploitent de telles similitudes pour réduire la quantité de données à acquérir par le transducteur. Par exemple, les données intéressantes peuvent changer lentement en fonction du temps. Dans ce cas, des corrélations temporelles peuvent être exploitées pour réduire le nombre d'acquisitions.

Une approche semblable peut être appliquée lorsque le phénomène étudié ne change pas brusquement entre les régions couvertes par des nœuds voisins. L'énergie due au prélèvement (et à la communication) peut être alors réduite en profitant des corrélations spatiales entre les données prélevées. Clairement, des corrélations temporelles et spatiales peuvent être conjointement exploitées pour réduire sensiblement la quantité de données à acquérir.

(2) L'approche d'échantillonnage hiérarchique suppose que les nœuds sont équipés de sondes (ou détecteurs) de différents types. Alors que chaque sonde est caractérisée par une résolution donnée et sa consommation d'énergie associée, cette technique choisit dynamiquement la classe à activer, afin d'obtenir un compromis entre la précision et l'économie d'énergie.

(3) Enfin, l'échantillonnage actif fondé sur un modèle adopte une approche semblable à la prévision de données. Un modèle du phénomène mesuré est établi lors des prélèvements de données, de telle sorte que les valeurs futures puissent être prévues avec une certaine précision. Cette approche exploite le modèle obtenu pour réduire le nombre d'échantillons de données, et également la quantité de données à transmettre à la Station de Base bien que ce ne soit pas leur objectif principal.

II.6-2 Techniques de mobilité

Dans certains cas où les nœuds sont mobiles, la mobilité peut être utilisée comme outil pour réduire la consommation d'énergie *(au-delà du duty-cycling et des techniques orientées données).* Dans un réseau de capteurs statiques, les paquets provenant des nœuds suivent des chemins multi-sauts vers la station de base.

Ainsi, certains chemins peuvent être chargés *(sollicités plus que d'autres),* et les nœuds proches de la Station de Base relayent plus de paquets et sont plus sujets à l'épuisement prématuré de leurs batteries *(funneling effet).* Si certains nœuds *(éventuellement, la station de base*) sont mobiles, le trafic peut être modifié si les nœuds mobiles sont chargés de collecter des données directement à partir de nœuds statiques. *[***3***]*

Comme un facteur qui passe par chaque maison récupérer leurs courrier; les nœuds ordinaires attendent le passage d'un ***dispositif mobile*** pour lui envoyer *leurs messages* de telle sorte que la communication ait lieu à proximité (directement ou au plus avec un nombre limité de sauts en évitant le routage de données). Par conséquent, les nœuds ordinaires peuvent économiser de l'énergie parce que la longueur du chemin, la contention et les overheads de diffusion sont ainsi réduits.

En outre, le dispositif mobile peut visiter le réseau afin de répartir uniformément la consommation d'énergie due à la communication. Lorsque le coût de la mobilité des nœuds de capteurs est prohibitif, l'approche classique consiste à attacher un capteur à des entités qui seront en itinérance dans le champ de détection, comme des autobus ou des animaux.

- **Les stratégies reposant sur la mobilité peuvent être classées en deux ensembles**

Les stratégies avec ***un Sink (puits) mobile*** et les stratégies avec ***des relais mobiles***, selon le type de l'entité mobile. Il est important de souligner ici que, lorsque nous examinons des systèmes mobiles, un problème important est le type de contrôle de la mobilité des nœuds qu'intègre la conception du réseau.

- **Les nœuds mobiles peuvent être divisés en deux catégories**

Ils peuvent être spécifiquement conçus comme partie de l'infrastructure du réseau, ou faire partie de l'environnement.

(1) Quand ils font partie de l'infrastructure, leur mobilité peut être entièrement contrôlée dans la mesure où ils sont, généralement, robotisés.

(2) Lorsque les nœuds mobiles font partie de l'environnement, ils pourraient ne pas être contrôlables. S'ils suivent un horaire strict, ils ont une mobilité complètement prévisible (par exemple, une navette pour les transports publics). Sinon, ils peuvent avoir un comportement aléatoire de sorte qu'aucune hypothèse ne puisse être faite sur leur mobilité. Enfin, ils peuvent suivre un schéma de mobilité, qui n'est ni prévisible, ni totalement aléatoire.

Par exemple, c'est le cas d'un bus se déplaçant dans une ville, dont la vitesse est soumise à d'importantes variations en raison des conditions de circulation. Dans un tel cas, les schémas de mobilité peuvent être tirés en se fondant sur des observations et des estimations d'une certaine précision.

II.6-3 Technique du Duty-cycling

Cette technique est principalement utilisée dans l'activité réseau. Le moyen le plus efficace pour conserver l'énergie est de mettre la radio de l'émetteur en ***mode veille*** *(low-power)* à chaque fois que la communication n'est pas nécessaire. Idéalement, la radio doit être éteinte dès qu'il n'y a plus de données à envoyer et ou à recevoir, et devrait être prête dès qu'un nouveau paquet de données doit être envoyé ou reçu. Ainsi, les nœuds alternent entre périodes actives et sommeil en fonction de l'activité du réseau. Ce comportement est généralement dénommé Duty-cycling. Un Duty-cycle est défini comme étant la fraction de temps où les nœuds sont actifs.[**3**]

Notons qu'il doit y avoir un algorithme d'ordonnancement pour gérer le passage d'un état à un autre pour chaque nœud ***(sommeil/réveil).*** Il s'agit généralement d'un algorithme distribué qui surveille le temps où chaque nœud décide de se mettre en veille ainsi il assure l'interaction et l'échange de paquets possible.

Nous nous intéressons à cette technique qui utilise ***la mise en veille*** des nœuds ; comme nous l'avons remarqué dans l'étude bibliographique les mots « ***mise en veille ; éteindre le module radio ; sleeping ; sommeil d'un nœud*** » sont souvent utilisé ; cela nous pousse à dire que dans les réseaux de capteurs sans fil cette technique est indispensable voir la plus utilisable pour préserver et économiser la consommation d'énergie, d'où augmenter la durée de vie du réseau.

Avant d'aborder le chapitre de mise en veille nous allons voir quelques protocoles économes en énergie dont la technique de ***mise en veille*** est la plus utilisé.

II.7 Mécanismes de conservation d'énergie aux niveaux de différentes couches

Comme nous l'avons mentionné dans le chapitre précédant ; l'énergie dissipée est répartie entre les différentes couches (couches physique, liaison, réseau et transport) entre traitement, capture et communication, nous allons décrire dans ce qui suit quelques mécanismes de conservation de l'énergie au niveau de ces couches.

Selon l'étude bibliographique de la thèse *[**8**]* nous avons le résumé des résultats de différentes études sur les mécanismes de conservation d'énergie dans les différentes couches des réseaux WSN.

II.7-1 Au niveau physique

(a) La compression, ou codage de source *[**43**]* ; une technique qui permet de réduire le taux moyen de données transmises par des capteurs dont les lectures sont corrélées selon une fonction de corrélation (l'idée se base sur les travaux de *[**44**])* et *[**45**]*.

(b) *[**46**]* proposent un algorithme de codage réparti, implanté sur un réseau de capteurs, qui permet aux nœuds de compresser d'une façon "aveugle" (i.e., sans connaître les données transmises par les nœuds voisins ni l'expression de la corrélation entre leurs données et celles de leurs voisins) leurs données à un taux spécifié par le CT. L'algorithme comprend une composante qui estime d'une façon adaptative la corrélation entre les nœuds du réseau. La solution proposée par *[**46**]* est adaptée aux RCSF car les nœuds, qui codent les données, sont munis d'un encodeur à faible complexité qui leur permet de consommer une énergie relativement faible dans le codage.

(c) Dyamic Voltage Scheduling (DVS) proposé par *[**47**]*.Une famille de techniques qui consistent en une planification dynamique du voltage qui alimente le processeur, en fonction des besoins de calcul de ce dernier. Il s'agit de prévoir le comportement du processeur et d'adapter sa fréquence et son courant en entrée en fonction des tâches qu'il a à accomplir. Pour cela, des algorithmes

complexes appelés "planificateurs de voltage" (Voltage Schedulers (VS)) sont requis pour évaluer la charge future du processeur et adapter son courant d'alimentation et sa vitesse.

II.7-2 Couche liaison

- Passe par deux techniques :

1- La limitation du temps d'accès au canal (*mise en veille des capteurs*).
2- Une gestion plus efficace des transmissions afin de limiter les collisions

Les protocoles d'accès efficaces au canal se divisent en deux catégories principales *(nous allons détailler dans les paragraphes qui suivent quelques protocoles)* :

- accès sans collision TDMA (Time Division Multiple Access)
- accès avec contention
- accès avec combinaison des deux types

TDMA : l'axe du temps est divisé en trames temporelles (time slots).

- **TRAMA** : *[***48***]* proposent un protocole qui divise chaque trame de temps en deux périodes; l'une à accès aléatoire (avec contention) l'autre à accès planifié (TDMA) où chaque capteur à accès au canal dans la période qu'il a réservée auparavant.

Avec cette méthode les capteurs peuvent savoir le temps dont ils doivent êtres actifs ou se mettre en veille.

- **B-MAC** : un protocole d'accès avec contention proposé par *[***49***]* intégré au système d'exploitation TinyOS. Dans B-MAC, les capteurs effectuent, entre chaque deux période de veille successives, des écoutes périodiques du canal pendant un intervalle de temps fixe pour vérifier si des communications ont lieu. La fréquence des réveils pour écoute s'adapte aux besoins de l'application.

- **ASCENT** : proposé par *[***50***]*, c'est un mécanisme réparti qui permet aux nœuds d'adapter leurs états aux conditions du réseau, cela en informant ces voisins qui décident de passer de *l'état en veille* à *l'état actif* de participer au routage.

- **S-MAC** : proposé par *[***51***]*, c'est un protocole adaptatif d'accès au canal qui introduit une modification sur la couche MAC afin de l'adapter aux contraintes énergétiques des WSN. Il favorise l'accès au canal aux nœuds ayant beaucoup de trafic à transmettre. S-MAC permet aussi aux nœuds de passer en mode en ***veille (sleep)*** périodiquement lorsqu'il n'y pas de trafic à acheminer pour éviter l'écoute excessive en l'absence de trafic. Pour traduire les délais et les trames de contrôle engendrés par le réveil provoqué de certains nœuds suite à leur ***veille périodique***, S-MAC utilise l'écoute adaptative qui consiste à provoquer un changement d'état des nœuds, passant de ***l'état en veille*** **à** ***l'écoute***, selon le type et l'importance du trafic.

- *[***52***]* proposent un mécanisme de conservation de l'énergie, il consiste à réduire les transmissions afin d'éviter les collisions.

II.7-3 Couche réseau

- **SPIN** : par *[***53***]* intègre la conservation de l'énergie grâce à la négociation. Dans ce cas le nœud diffuse d'abord dans tout le réseau une métadonnée décrivant les données à transmettre, seuls les nœuds qui manifestent leur intérêt pour cette métadonnée recevront les données en entier, ce qui sauvegarde considérablement la bande passante et l'énergie des nœuds.

- *[***54***]* proposent un protocole orientée-diffusion (Directed Diffusion), la conservation de l'énergie est réalisée grâce à l'agrégation de données.

- *[***55***]* proposent de répartir la charge de routage sur un ensemble de chemins sous-optimaux pour ne pas épuiser l'énergie des mêmes nœuds faisant partie du chemin optimal et ainsi prolonger la durée de vie du réseau.

Le choix de ces chemins se fait via une fonction de probabilité qui exprime la consommation énergétique de chacun d'eux. La survie du réseau est la métrique principale dans cette approche. Le protocole suppose que chaque nœud possède une adresse. Il a trois phases (initialisation, communication et maintenance).

- *[***56***]* proposent le protocole **Geographical Adaptive Fidelity (GAF)** qui conserve l'énergie du réseau en éteignant les capteurs qui ne sont pas nécessaires pour le fonctionnement du réseau.

La position du capteur est déterminée par un GPS et s'associe à un point d'une grille virtuelle regroupant des capteurs géographiques proches. Les capteurs associés au même point de la grille sont considérés équivalents et ne sont donc pas requis de fonctionner simultanément.

Certains nœuds peuvent donc se *mettre en veille*, conservant ainsi leur énergie résiduelle.

II.7-4 Couche transport

- **RMST (Reliable Multi-Segment Transport Protocol)** proposé par *[***57***]*, c'est un protocole qui tire profit de la corrélation des données transmises et utilise certains nœuds du réseau comme mémoire cache pour garantir l'acheminement de l'information des capteurs vers le nœud puits.

  L'arrivée des données au nœud puits ne suppose pas l'arrivée de tous les paquets transmis mais plutôt de l'information globale qui est extraite de la corrélation et des données en cache, chaque fois que des paquets sont perdus. Bien que ce mécanisme contribue à conserver l'énergie en minimisant les retransmissions il est peu adapté aux WSN, car il a besoin de la mémoire cache et une certaine capacité de calcul au niveau de certains nœuds pour le calcul de la corrélation.

- **ESRT (Event-to-Sink Reliable Transport protocol),** c'est un protocole proposé par *[***58***]*, il assure un transport fiable des données des capteurs vers le nœud puits sans avoir recours à une mémoire cache au sein du réseau, ce qui minimise l'énergie dépensée par rapport à RMST. Il est adapté aux WSN car il ne repose pas sur l'identification des nœuds mais plutôt sur celle de l'événement détecté, quelle que soit sa source.

  Comme il permet aussi de contrôler le flux des données allant des nœuds vers le nœud puits et les algorithmes de contrôle de flux sont implantés au nivaux du nœud puits qui ordonne aux nœuds de diminuer leurs flux respectifs dès qu'il détecte l'événement souhaité à partir des différentes lectures corrélées, sans garantir les arrivées individuelles des données envoyées par chacun des capteurs.

- **PSFQ (Pump Slowly, Fetch quickly)** proposé par *[***59***]*; c'est un protocole de transport basé sur une garantie locale d'arrivée des données. Chaque nœud intermédiaire doit conserver une copie du paquet transmis en cache jusqu'à s'assurer de l'arrivé de ce dernier à la destination, cela facilite la correction de pertes localement.

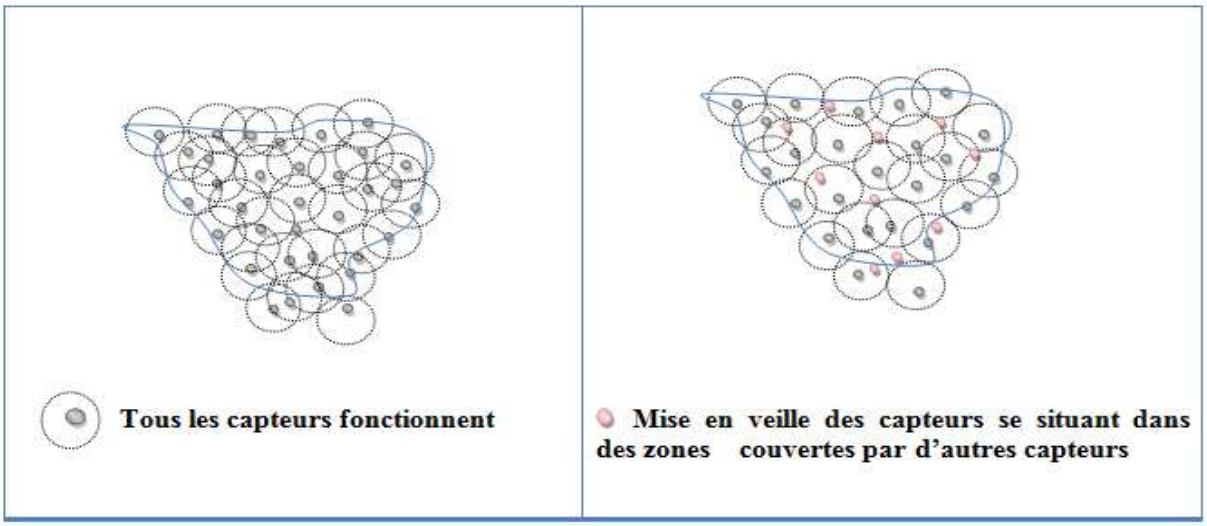
II.8 La conservation d'énergie sous différentes contraintes

II.8-1 conservation de l'énergie sous contraintes de couverture

- **Couverture de surface**

La surface couverte par le réseau est l'union des différentes zones de couverture de chaque objet. Les protocoles de couverture de surface visent à diminuer le nombre de nœuds actifs tout en conservant une couverture équivalente à celle offerte par l'ensemble des objets. La difficulté est d'assurer une couverture de zone totale à partir de décisions locales simples, tout en minimisant le nombre de nœuds actifs et l'énergie totale consommée. Un mécanisme de *mise en veille* alternative assurant une couverture totale est proposée dans *[***33***]*

Après une première phase de découverte du voisinage, chaque nœud décide d'un temps d'attente aléatoire au bout duquel il calcule la couverture fournie par ses voisins. En cas de couverture totale, il décide d'être inactif et envoie un message de retrait à ses voisins. Ceux n'ayant pas encore décidé mettent à jour leur table de voisinage. S'il décide de rester actif, aucun message n'est envoyé. *[***60***]* proposent une solution économe en nombre de messages échangé**.**



-Figure II-5 : conservation d'énergie sous contraintes de couverture-

Donc ce principe se base sur l'alternance des états des capteurs entre un état d'activité et un autre de ***mise en veille***.

1) *[***61***]:* ils ont fait une analyse géométrique de la relation entre la couverture de la zone surveillée et la connectivité du réseau, le protocole proposé s'appel **CCP (Coverage Configuration Protocol)** qui assure une configuration dynamique du réseau.

Sont principe de fonctionnement consiste à ce que chaque capteur choisisse son état « *actif ou en* ***veille*** », en fonction du degré de couverture des points d'intersection entre son cercle de couverture et celui de ces voisins.

2) *[***62***]:* présentent l'algorithme nommé **SCOM (Scalable COverage Maintenance)** ; c'est un protocole localisé pour le maintien de couverture.

Son principe de fonctionnement est : les capteurs utilisent une politique de redondance pour décider s'ils doivent s'allumer ou ***se mettre en veille*** ; les capteurs ayant une énergie résiduelle faible décident de leur état avant ceux qui on un résiduelle élevée, favorisant ainsi la prolongation de durée de vie du réseau.

3) *[***63***]* **:** proposent une heuristique centralisée qui calcule dynamiquement un ensemble presque-optimal de capteurs qui garantit un taux de couverture prédéfini, tout en assurant la connectivité du rayon de couverture.

4) *[***64***]:* proposent une planification des états « *actif/* ***en veille*** » des capteurs, pour assurer la couverture de chaque point de la zone.

Son principe de fonctionnement est : les capteurs voisins échangent un temps de référence aléatoire **T**ref à chaque période **T**, et prennent la décision de ce mettre ou non à l'état activé durant la prochaine période **T**.

Le problème de cette proposition est qu'elle ne prenne pas en considération l'énergie résiduelle de chaque capteur, ce qui rend les nœuds avec une énergie résiduelle faible plus enclins à disparaitre, d'où le disfonctionnement du réseau.

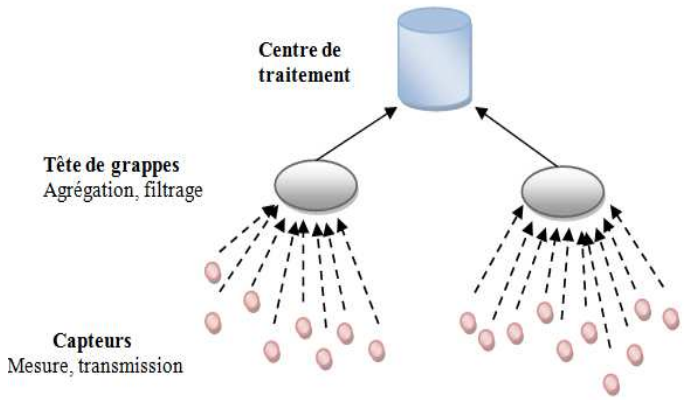
II.8-2 Conservation de l'énergie par la formation de grappes (clustering)

- **La notion de grappes (clustering)**

Cette technique consiste à diviser l'ensemble des nœuds en sous ensembles , dans chacun on définit un nœud en tête pour communiquer avec le reste des têtes des autres grappes et avec le puits *(sink)* et le restes des capteurs pour transmettre l'information capté à la tête de grappe qui les achemine jusqu'au centre de traitement , soit directement, soit en multi-sauts via des passerelles ou d'autres tête de grappe voisines.

Notant que les têtes de grappe allègent la quantité de données transmises par certaines opérations comme le filtrage et l'agrégation des données collectées par conséquent la minimisation de la consommation d'énergie des nœuds collecteurs.

L'organisation en grappes permet de mieux coordonner et ordonnancer les tâches des capteurs dans un RCSF. En effet, les tête de grappe peuvent ordonnancer l'activité des capteurs membres de leur grappe en ordonnant à certains capteurs de ***se mettre en veille*** ou de limiter leur puissance de transmission, par exemple dans le cas où il y a une redondance des événements rapportés ou des zones couvertes. Les têtes de grappe peuvent également organiser les instants de transmission des capteurs de leurs grappes respectives afin d'éviter les collisions et par conséquent les retransmissions. Cela dit, comme les tête de grappe consomment plus d'énergie que les capteurs réguliers, il est important de disposer d'un mécanisme efficace de rotation des tête de grappe.***[8]***



-Figure II-6 : Structure en grappes dans un WSN [8]-

1) LEACH : proposé par *[***53***];* c'est l'un des premiers protocoles de formation de grappes proposées pour les RCSF, il est réparti, proactif et dynamique.
La formation de grappes se base sur la puissance du signal reçu et les têtes de grappes sont les seuls nœuds qui routent les données.

2) Power-Efficient Gathering in Sensor Information Systems (PEGASIS): il est Proposé par *[***65***]*, sa variante Hierarchical-PEGASIS, sont deux améliorations de LEACH.
Au lieu d'avoir plusieurs grappes, on forme une chaîne de capteurs voisins directs, un seul de ces derniers est sélectionné pour envoyer directement les données au centre de traitement.

3) Weighted Clustering Algorithm(WCA) : proposé par *[***66***]* ; c'est un algorithme réactif de formation de grappes le choix de la tête de grappe est basé sur l'évaluation, pour chaque capteur on a une fonction qui s'appel « pondération combinée » (combined weight). Cette fonction score est une combinaison linéaire pondérée de degré, du niveau de mobilité, de la puissance de transmission et de l'énergie résiduelle d'un nœud.
Chaque capteur diffuse sa pondération combinée à ces voisins et le nœud ayant le score le plus bas est élu tête de grappe.

4) Hybrid Energy-Efficient Distributed Clustering(HEED): proposé par*[* **67***]*; c'est un autre protocole réparti de formation de grappes qui utilise une combinaison hybride de l'énergie résiduelle, et du coût de communication intra-grappe comme critère de sélection d'une tête de grappe pour garantir la connectivité du graphe formé par les têtes de grappes. Il favorise un uniforme pour la désignation d'une tête de grappe afin de garantir la connectivité du graphe formé par les têtes de grappes.

Dans la phase d'initialisation de HEED, chaque capteur *i* calcule sa probabilité *PTG i* de devenir TG. Cette probabilité est proportionnelle à son énergie résiduelle et à un taux prédéterminé de têtes de grappes dans le réseau. Ensuite, durant une phase de répétition, chaque capteur cherche la meilleure TG dans son voisinage, pour se connecter à elle. Si un capteur ne trouve aucune TG dans son voisinage, il double sa probabilité *PTG i* et la diffuse de nouveau à ses voisins. Puis, il cherche la meilleure TG dans son voisinage et ainsi de suite. Le capteur *i* arrête le processus soit lorsque *PTG i* = 1 (le capteur se promeut lui-même TG), soit lorsqu'il trouve une TG dans son voisinage. *« TG : tête de grappe »*

5) Energy-Efficient Clustering(EEHC) : proposé par *[**68**]* ; c'est un autre algorithme réparti de formation de grappes à un ou plusieurs niveaux de hiérarchie.

Chaque capteur s'avance à ses voisins éloignés de K sauts, comme tête de grappe potentielle, avec une probabilité **P**. Ensuite, chaque capteur se connecte à la tête la plus proche.

Les capteurs qui ne reçoivent aucune annonce et qui ne sont pas têtes de grappes ils le deviennent alors forcément.

6) Energy-Efficient Strong Head Clustering(EESH): proposé par ***[69]***, est un autre protocole de formation de grappes dans lequel les têtes de grappes sont élues en fonction de leurs coûts. Elles sont calculées à partir de leurs énergies résiduelles respectives, leurs degrés respectifs, les distances les séparant de leurs voisins et les énergies résiduelles de ces derniers. Le capteur ayant le coût le plus élevé est élu TG. Le processus d'élection se termine lorsque chaque capteur est soit élu TG, soit connecté à une TG voisine.

II.8.3 conservation de l'énergie par l'ajustement optimisé des puissances de transmission

Dans les réseaux de capteur, la transmission de données est la plus consommatrice en énergie. Nous avons dans ce tableau une comparaison en utilisant les capteurs MEDUSA-II et WINS.

La conservation de l'énergie dissipée est donc dépendante d'une optimisation de la puissance de transmission utilisée, tout en respectant certaines contraintes du réseau comme, par exemple, la contrainte de connectivité de chaque capteur à une TG, ce qui implique l'utilisation d'une puissance minimale permettant d'atteindre au moins une TG voisine.

-Tableau II-1: Consommation énergétique, aux différents états d'opération, des capteurs WINS de Rockwell- *[8]*

Etat du capteur	Etat di module radio	Puissance consommé (mW)
	Tx (36.3 mW)	1080.5
	Tx (19.1 mW)	9.86
	Tx (13.8 mW)	9426

Actif	Tx (3.47 mW)	815.5
	Tx (2.51 mW)	807.5
	Tx (0.96 mW)	787.5
	Tx (0.30 mW)	773.9
	Tx (0.12 mW)	771.1
Actif	Rx	751.6
Actif	Ecoute passive	727.5
Actif	Eteint	383.3
Ecoute passive	Eteint	64

-Tableau II-2 : Consommation énergétique des différents états d'opération des capteurs MEDUSA II-*[8]*

Etat du capteur	**Etat du module radio**	**Schéma de modulation**	**Taux de transmission (kb/s)**	**Puissance consommée (mW)**
Actif	Tx (0.7368 mW)	OOK	2.4	24.58
	Tx (0.0979 mW)	OOK	2.4	19.24
	Tx (0.7368 mW)	OOK	19.2	25.37
	Tx (0.0979 mW)	OOK	19.2	20.05
	Tx (0.7368 mW)	ASK	2.4	26.55
	Tx (0.0979 mW)	ASK	2.4	21.26
	Tx (0.7368 mW)	ASK	19.2	27.46
	Tx (0.0979 mW)	ASK	19.2	22.06
Actif	Rx	Tout	Tout	22.20
Actif	Ecoute passive	Tout	Tout	22.06
Actif	Eteint	Tout	Tout	9.72
Eteint	Éteint	Tout	Tout	0.02

1) *[**70**]:* considèrent un RCSF hétérogène où un ensemble prédéfini de capteurs avec des ressources « illimitées » est assigné le rôle de TG et où chacun des autres capteurs doit être connecté à une TG moyennant un lien direct ou multi-sauts.

Principe de fonctionnement : un rayon de transmission optimal est calculé pour chaque ensemble de capteurs situés à égale distance de la tête de grappe la plus proche. Plus les capteurs sont proches de la TG, plus ils vont dissiper de l'énergie dans le routage des données provenant des nœuds plus lointains de la TG, mais moins ils vont dissiper de l'énergie dans la transmission. La dissipation de l'énergie dans le routage est compensée par l'ajustement de la puissance de transmission.

Le disque centré en la TG et contenant tous les capteurs connectés à cette TG est divisé en anneaux de largeur variable. Le rayon optimal de chaque anneau est calculé de manière à minimiser l'énergie totale dépensée par le réseau. Toutefois, le fait que l'ensemble des TG est prédéfini et fixé à l'avance a tendance à produire des configurations sous-optimales.

2) *[**71**]* **:** proposent un protocole réparti qui ajuste dynamiquement la puissance d'énergie de transmission de chaque capteur pour contrôler la taille de grappes, indépendamment de la distribution des nœuds.

Principe de fonctionnement : chaque capteur ajuste sa puissance de transmission en fonction du nombre de voisins, ensuite un ensemble de TG est élu selon l'énergie résiduelle ; se qui assure une performance élevé en terme de durée de vie mais la contrainte de couverture n'est pas prise en charge. L'autre faille est que le choix de têtes de grappes se fait intuitivement se basant uniquement sur l'énergie résiduelle.

3) *[**72**]* **:** présentent deux algorithmes répartis pour l'ajustement des puissances de transmission.

Principe de fonctionnement :

Le premier se base sur un seuil minimal du nombre de voisins, le second sur un seuil du nombre moyen des voisins.

4) *[73]* **:** évaluent analytiquement la puissance de manière expérimentale (sans simulations) dans le cas d'une topologie aléatoire, ils font une étude de la corrélation entre la puissance de transmission optimale, le débit de données et la densité spatiale des nœuds.

II.8-4 Conservation de l'énergie par l'ajustement optimisé des états des capteurs

Le but de cette branche de recherche qui s'intéresse à la planification optimale des états des capteurs et l'ordonnancement optimal des activités, est de minimiser l'énergie dissipée par le réseau.
C'est l'internonce entre les états d'un capteur, quand il est activé alors il consomme un certain niveau d'énergie, et un autre où il est éteint et donc consomme une quantité d'énergie négligeable.

1) *[74]* **:** ont fait une comparaison entre les consommations d'énergie respectives de deux types de capteurs MEDUSA II (développés au laboratoire de réseautique et des systèmes embarqués de l'université de Californie à Los Angeles(UCLA), et WINS, développés par la compagnie Rokwelle *(voire les tableaux II-1 et II-2)* on remarque alors une différence de consommation d'énergie entre ***l'état actif*** et ***en veille***.

2) *[75]* **:** proposent un algorithme réparti pour trouver un sous-ensemble optimal de couverture ou si ce n'est pas possible, un ensemble de couvertures offrant le minimum de points géographiques sont couverts.

Principe de fonctionnement : Dans cet algorithme, si la portée entière d'un capteur est couverte par un sous-ensemble de ses voisins directs alors il ***se met en veille***.
Pour que deux capteurs ne se mettent pas en veille simultanément, un temps ***(back-off)*** aléatoire est défini pour éviter se problème ;

Cependant, ce protocole n'est pas optimal à cause de la redondance dans la couverture due à l'aspect localisé des décisions prises par ces derniers.

3) *[76]:* proposent un mécanisme d'allocation des états aux capteurs dans un réseau hiérarchisé en grappes, avec garantie de couverture, ce mécanisme réparti d'auto-organisation se base sur un réseau dense et affecte les états (***actif, en veille***, TG) aux capteurs, diviser les capteurs en grappes, ensuite les sous ensembles de capteurs qui peuvent être mis à l'état actif simultanément pour assurer la couverture totale de la zone, un seul de ces ensembles sera activé

jusqu'à se qu'il épuise son énergie, tandis que les autres se mettent en marche après l'épuisement du dernier cluster.

II-9 Conclusion

Dans ce chapitre nous avons abordé l'état de l'art des techniques de gestion d'énergie dans les réseaux de capteurs en passant par différentes études rencontrés dans la littérature, nous avons parlé des différentes techniques d'économie d'énergie pour les RCSF, particulièrement la mise en veille, en première partie. Ensuite dans la deuxième partie nous avons abordé quelques protocoles économes en énergie. Nous avons remarqué que la plupart de ces protocoles utilisent la mise en veille du module radio.

Le chapitre suivant contient quelques détails sur la technique de mise en veille, l'une des solutions utilisées pour économiser l'énergie d'un réseau de capteurs sans fil, qui permettent de maximiser la durée de vie de ce dernier afin qu'il puisse être fonctionnel le plus longtemps possible.

Sommaire

III.1 Introduction

Mettre en veille un capteur c'est mettre hors tension certains composant d'un capteur particulièrement (le module radio qui est le composant le plus consommateur en énergie) ; c'est pour ça que l'idée de « ***mise en veille*** de capteurs quand ils ne sont pas utilisés » a motivé beaucoup de recherches dans les réseaux de capteurs sans fil *[**49**][**50**][**51**][**61**][**62**][**74**][**75**][**76**][**33**]*.

Pour bénéficier de cette technique de conservation d'énergie, plusieurs contraintes s'imposent, donc il ne s'agit pas d'une seule règle à appliquer pour avoir un bon résultat, dans ce chapitre nous allons expliquer cette technique ainsi que le bénéfice qu'elle apporte en ce basant sur les résultats obtenus dans la recherche (exemples d'utilisation de la ***mise en veille*** dans quelques protocoles).

Les protocoles au niveau des différentes couches peuvent ***mettre en veille*** les capteurs :

1) La sélection des nœuds sources qui devraient transmettre les données au puits faite à la couche application, permet aux nœuds de sources redondantes de *dormir* dans l'ordre pour économiser l'énergie pour une utilisation ultérieure.

2) Le contrôle de topologie, qui est habituellement effectué à la couche réseau, crée un chemin de routage pour la livraison de données de sorte que le reste des nœuds Non-routeur peuvent *dormir*. ***La mise en veille*** des nœuds de routage qui ne sont pas directement impliqués dans la livraison des données peut être faite par le protocole de routage.

3) le *duty cycling* est souvent employé par le protocole MAC, permettant aux capteurs de dormir périodiquement afin de réduire leur *idle lisening*, ce qui est énergivore.

III.2 Définition de la mise en veille dans les WSN

Un élément qui fonctionne sur batterie ne peut avoir une durée de vie indépendante de l'énergie dédié à celle là, y compris le nœud capteur qui est un élément important dans une infrastructure (réseau). Pour un nœud tout seul, la problématique est simple :

Eteindre le nœud (les dispositifs concernés radio, capteur et CPU quand ils ne sont pas utilisés).

Mais, parce que ce nœud fait partie d'un réseau qui enchaine les données captés vers une station de base ; plusieurs contraintes doivent être prises en compte, et pour que ***la mise en veille*** soit plus bénéfique que compliquée, plusieurs protocoles adaptent cette technique au profit de l'énergie, donc le bénéfice se généralise d'une augmentation d'une durée de vie dans un capteur au réseau entier. Dans le chapitre précédent nous avons vu que « ***la mise en veille*** » est souvent impliqué dans les différents protocoles conçus pour WSN.

Selon les couches protocolaires du réseau WSN :

- Dans les études faites au niveau de la couche liaison tel que TRAMA [**48**], TDMA, BMAC [**49**], ASCENT [**50**] et SMAC[**51**] présentés précédemment ;
- Aussi au niveau de la couche réseau tel que le travail de [**56**], à titre d'exemple ;

Selon la zone de couverture :

- tel que les travaux présentés par [**33**], [**61**], [**62**] et [**64**].

Selon l'ajustement optimisé des états de capteurs :

- tel que [**74**] [**75**] et aussi [**76**].

Donc la définition globale de ***la mise en veille*** dans les WSN est la suivante :

Mettre les modules les plus consommateurs en énergie dans les nœuds d'un réseau de capteurs (EX : radio) en mode ***veille*** (hors tension) afin de préserver l'énergie de leurs batteries quand ils ne sont pas en fonction (dans le cas du tranceiver « radio » émission/réception de données) ;(dans le cas du capteur ; phase de captage d'un événement) ; (dans le cas de la CPU,

existence de données à traiter on peut citer la compression de données par exemple avant de les envoyer[**91**] pour un autre nœud). pour cela il faut respecter les contraintes du réseaux comme le domaine d'utilisation du réseau (par exemple pas pour un réseaux WSN temps réel ou le nœud doit être tout le temps éveillé pour recevoir ou envoyer les données) ; la densité des nœuds, les techniques de routage….etc. donc pour tirer profit de cette technique les recherches faites pour les WSN l'ont adapté d'une manière intelligente et réfléchie pour qu'elle soit basé sur plusieurs protocoles MAC et de routage.

III.3 Composition technologique

Maximiser la durée de vie des RCSF réside dans la nécessité de réduire au minimum la consommation d'un nœud, tout en garantissant un maximum de performance aux utilisateurs.

En effet pour étudier l'énergie d'un réseau de capteurs sans fil, il faut s'approcher du composant essentiel qui est le nœud capteur.

Cette approche basse consommation impose des composants peu gourmands en énergie. D'apparence triviale, la démarche se révèle souvent complexe. Premier paramètre visé **: la consommation**, en temps normal**, du processeur**, **du capteur**, **de l'émetteur récepteur radio** et d'autres composants, comme **la mémoire externe** et **les périphériques**.

Opter pour des ressources basse consommation oblige à des compromis sur la performance. D'ordinaire, un processeur de faible puissance tourne à une fréquence d'horloge réduite en ayant moins de fonctionnalités intégrées que ses homologues plus consommateurs en énergie. L'astuce consiste à choisir des éléments tout justes assez performants pour remplir leurs missions.

III.4 Minimiser la consommation des composants

Il importe de minimiser la consommation en ***mode veille*** ; pour cela, il est souvent possible de couper l'alimentation du capteur et de l'émetteur récepteur. Le processeur, par contre, exigera une autre forme de ***mise en sommeil***, désactivable en temps utile. Une basse consommation dans ce

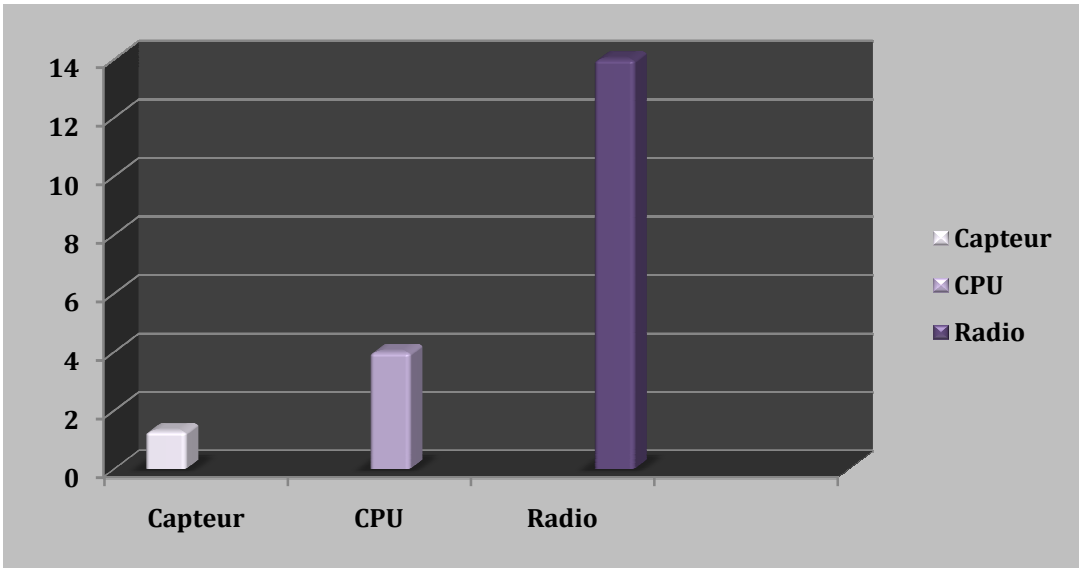
mode est déterminante pour la facture énergétique globale. Signalons un aspect souvent négligé : le temps nécessaire pour ***allumer*** et ***éteindre*** ces éléments.

L'émetteur/récepteur (radio ou tranceiver), par exemple, aura besoin d'un délai minimal pour stabiliser ses oscillateurs. Entre-temps, *émetteur/récepteur* et processeur consomment de l'énergie : il faut donc minimiser ce gaspillage. Cela est aussi vrai pour alimenter ces deux composants. Enfin, il faut s'assurer que tous les éléments nécessaires sont sous contrôle du processeur qui, en véritable coordonnateur du système, maîtrise la totalité de ses blocs fonctionnels.

III.5 La problématique « système »

Souvent, le protocole de communication est imposé. Le but est d'utiliser les ressources disponibles dans les limites acceptables sans jamais rien avoir à alimenter de superflu. Cela revient à allumer et à éteindre des unités comme *le capteur*, *le processeur* et *l'émetteur/récepteur (radio).*

Ces trois éléments s'avèrent les plus fonctionnels dans une mote (un nœud capteur), comme nous le remarquons dans la **Figure III-1** le taux de consommation d'énergie vari entre ces trois comme le montre la Figure III-1 :



-Figure III-1 : Consommation d'énergie par les modules (Capteur, CPU, Radio)-

1. Pour la gestion de la radio

Le module radio (tranceiver) est le composant le plus consommateur en énergie, pour préserver l'énergie dans le temps ou ce dernier n'est pas en réception ou émission de données, le système de gestion d'énergie effectue des événements suivants :

- pour mettre en veille la radio le système déclenche ***«* ACTION_power_down_radio»** et pour la remettre à l'écoute le système déclenche l'événement suivant :
  « ACTION_power_up_radio ».

2. ***Pour la gestion de la CPU***

La consommation d'énergie dans la CPU est due aux applications et algorithmes de traitement des données acquises après capture ou réception, quand il n'y a plus de données à traiter, on peut éteindre ce module, pour le faire, le système déclenche l'événement :

- Pour mettre en veille la CPU ***«*ACTION_power_down_CPU »**et pour la redémarrer **« ACTION_power_CPU ».**

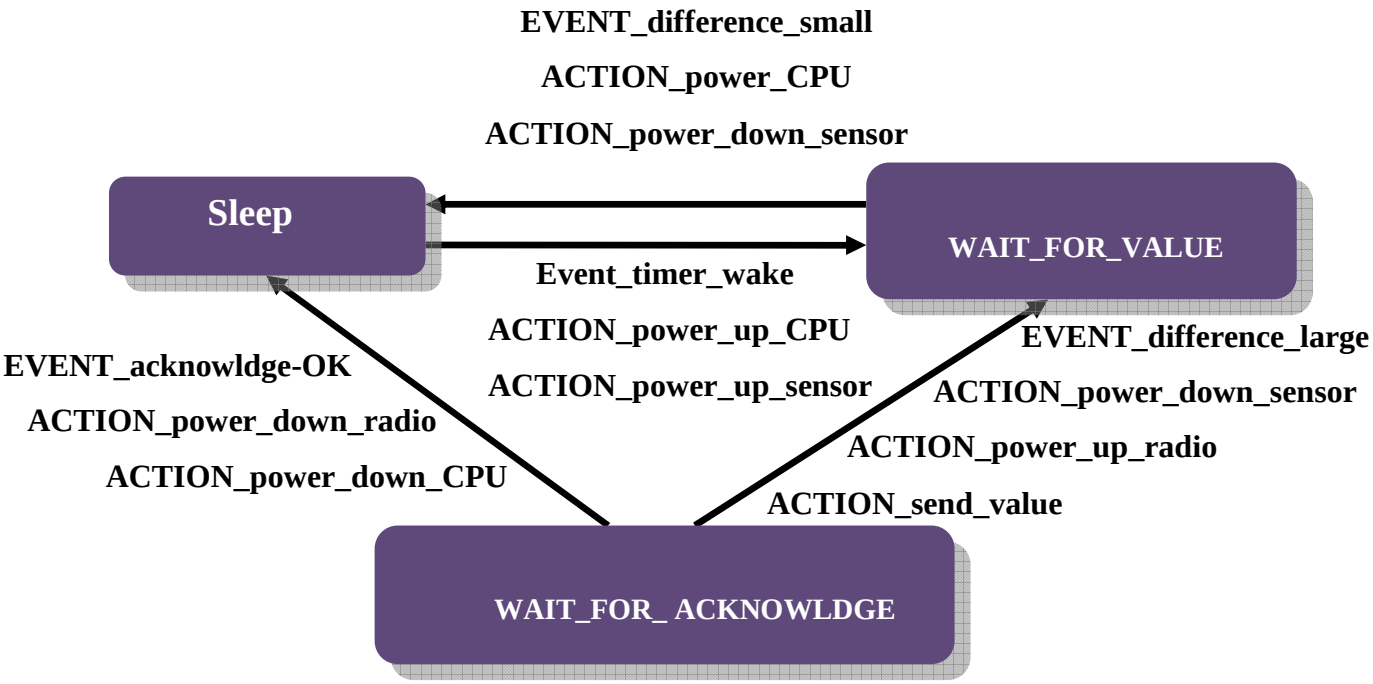
3. ***Pour la gestion des capteurs***

Les capteurs (sensors) sont des dispositifs sensibles à des phénomènes naturels qu'ils transforment en signaux analogiques puis valeurs numériques pour pouvoir les traiter; pour la plupart des applications (orientés événement par exemple) le capteur reste actif sans capter un événement ; pendant cette période il peut s'éteindre afin d'éviter la consommation d'énergie.

- Pour mettre en veille le capteur, le système déclenche l'événement suivant :
  « ACTION_power_down_radio » et pour le redémarrer il déclenche l'événement
  « ACTION_power_up_sensor ».

Parce qu'éteindre et allumer nécessite une harmonie entre les différents états des modules et le fonctionnement du logiciel et du système, le schéma de la **Figure III-2** nous résume les

différentes transitions entre les ***modes veille***, *attente d'une valeur* ou *d'un acquittement* des trois modules ; quand est ce que le système peut éteindre et allumer chacun d'eux.



-Figure III-2 : Evénements et actions entraînant un changement d'état du logiciel-

Prenons l'exemple d'un ***nœud dormant*** qui doit ***se réveiller*** à intervalles réguliers pour transmettre sa mesure, mais seulement lorsque celle-ci s'écarte du précédent relevé de plus d'une certaine valeur. Une fois la mesure transmise sur le ***canal radio***, l'unité attend un message d'acquit lui confirmant la bonne réception du paquet de données. Le comportement souhaité du logiciel est mieux expliqué sous la forme d'un diagramme schématisant l'état présent du logiciel **Figure III-2** les événements pouvant entraîner son changement d'état et les actions associées à chaque transition.

III-6 Estimation de la durée de vie d'un nœud avec et sans mise en veille

Operation	Telos	Mica2	MicaZ
Minimum Voltage	1.8V	2.7V	2.7V
Mote Standby (RTC on)	5.1 μA	19.0 μA	27.0 μA
MCU Idle (DCO on)	54.5 μA	3.2 mA	3.2 mA
MCU Active	1.8 mA	8.0 mA	8.0 mA
MCU + Radio RX	21.8 mA	15.1 mA	23.3 mA
MCU + Radio TX (0dBm)	19.5 mA	25.4 mA	21.0 mA
MCU + Flash Read	4.1 mA	9.4 mA	9.4 mA
MCU + Flash Write	15.1 mA	21.6 mA	21.6 mA
MCU Wakeup	6 μs	180 μs	180 μs
Radio Wakeup	580 μs	1800 μs	860 μs

-**Tableau III-1 :** consommation d'énergie dans les différents modes pour Telos, Mica2, MicaZ[**86**] -

Cette estimation de durée de vie d'un nœud est réalisée par [86] ; la mise en veille du module radio montre un énorme bénéfice dans la durée de vie d'un nœud. Il doit rester à l'écoute du réseau en permanence pour pouvoir accomplir sa fonction de relai.

Sur 1 seconde on a :
– Calcul et écoute : 2ms à 1,2 mA
– Réception : 250/19200 s = 13 ms à 4.5 mA
– Emission : (250+50)*8/19200 s = 15,6 ms à 12 mA
– Le reste (969,4 ms) est en écoute à 12 mA
Consommation : 29.7 mA
A ce régime, avec sa batterie de 2300mAh, il pourra fonctionner pendant 3 jours et 5 heures et 26 minutes.

S'il n'écoutait pas le réseau tout le temps mais passait en mode veille quand il est inactif :
Sur 1 seconde on aurait :
– Calcul : 2ms à 1.2 mA
– Réception : 250/19200 s = 13 ms à 4.5 mA
– Emission : (250+50)*8/19200 s = 15,6 ms à 12 mA
– Le reste (969,4 ms) est en veille à 8 µA
Consommation : 17.708 mA
A ce régime, avec sa batterie de 2300mAh, il pourra fonctionner pendant 5 jours, 9 heures et 53 minutes

Nous remarquons que le gain obtenu dans cette expérience est égal à ***40 %*** de l'énergie consommé sans mise en veille.

III.7 Questions de protocole

Au-delà de cette double contrainte ***d'électronique basse puissance*** et de ***mise en veille/réveil*** « *intelligents* » des unités, le protocole de communication joue un rôle majeur dans le bilan énergétique du système.

Des détails du protocole fixent les seuils inférieurs de la consommation électrique.

Certains protocoles sont réputés pour leur piètre efficacité ; dans ce cas, aucune programmation de logiciel embarqué au monde, aussi fine soit-elle, ne saura ramener la consommation à un niveau acceptable.

A l'inverse, d'autres protocoles sont d'emblée conçus dans une optique de faible consommation, sans dégrader les performances de la transmission. Par exemple:

WISA1 (*Wireless Interface to Sensors and Actuators*)

Son haut niveau de performances s'appuie sur deux techniques : ***le saut unique et le multiplexage temporel***. La première gomme les retards de transmission au niveau des nœuds intermédiaires ; la seconde garantit l'occupation du canal par un seul nœud, écartant tout risque de collision.

ZigBee [87] et son protocole normalisé 802.15.4

Sont plus généralistes, mais moins performants. Dans un réseau multi saut, un message peut transiter par plusieurs relais radio avant d'atteindre sa cible. Les nœuds n'ont pas d'allocation de créneaux temporels pour transmettre mais doivent se partager le canal en accès multiple ; si davantage d'utilisateurs peuvent ainsi accéder au support, cette méthode a le défaut d'ajouter son lot d'incertitudes tout en augmentant le retard et la consommation lorsqu'un noeud doit attendre son tour. Plus embêtant, les noeuds intermédiaires ne savent pas quand relayer la transmission.

III.8 Les réseaux de capteurs « Mostly-off et Mostly-on »

Le comportement global de la communication dans un réseau de capteurs sans fil est axé sur l'application. Par conséquent, nous trouvons plusieurs types de réseaux en fonction du modèle de délivrance de données qu'impose l'application (continu, initié par la Station de Base ou hybride). Ce modèle influe sur l'activité radio des nœuds, ce qui fait que plusieurs types de réseaux existent, en l'occurrence les réseaux *« Mostly-off »* et les réseaux *« Mostly-on »*.

Les réseaux de capteurs *« mostly-off »* sont évoqués dans plusieurs documents notamment dans c'est une catégorie de réseaux de capteurs, où les nœuds gardent leurs radios éteintes pendant de très longues périodes. Par la suite, ces nouds se réveillent pour envoyer leurs données à un collecteur.

Le problème qui revient souvent dans ce type de réseau est le problème de la synchronisation entre les nœuds dû à la dérive d'horloge.par contre *« mostly-on »* sont les réseaux de nœuds dont les radios sont la plupart du temps allumées.*[**3**]*

III.9 Classification des protocoles Sleep/wake up

Comme nous venons de le mentionner un régime ***sleep/wakeup*** peut être défini pour un composant donné (i.e. le module Radio) du nœud-capteur. On peut relever les principaux plans ***sleep/wakeup*** implantés sous forme de protocoles indépendants au-dessus du protocole **MAC** (*au niveau de la couche réseau ou de la couche application*). Les protocoles ***sleep/wakeup*** sont divisés en trois grandes catégories : à la demande, rendez-vous programmés, régimes asynchrones [**3**].

1) Les protocoles à la demande utilisent l'approche la plus intuitive pour la gestion d'énergie. L'idée de base est qu'un nœud devrait se réveiller seulement quand un autre nœud veut communiquer avec lui. Le problème principal associé aux régimes à la demande est de savoir comment informer un nœud en ***sommeil*** qu'un autre nœud est disposé à communiquer avec lui. À cet effet, ces systèmes utilisent généralement plusieurs radios avec différents compromis entre énergie et performances (*une radio à faible débit et à faible consommation pour la signalisation, et*

une radio à « haut » débit mais à plus forte consommation pour la communication de données). Le protocole **STEM** (Sparse Topology and Energy Management), par exemple, utilise deux radios.

2) Une autre solution consiste à utiliser une approche de rendez-vous programmés. L'idée est que chaque nœud doit se réveiller en même temps que ses voisins. Typiquement, les nœuds se réveillent suivant un ordonnancement de réveil et restent actifs pendant un court intervalle de temps pour communiquer avec leurs voisins. Ensuite, ils se rendorment jusqu'au prochain rendez-vous.

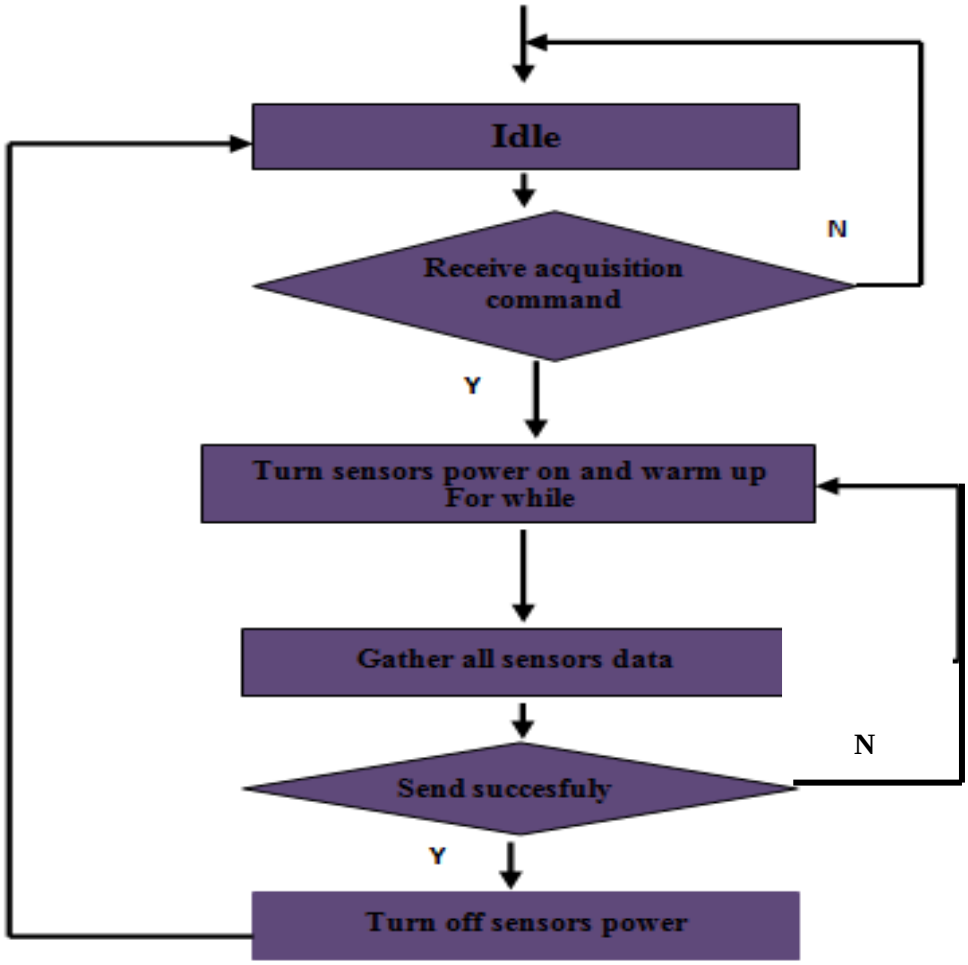
3) un protocole ***sleep/wakeup*** asynchrone peut être utilisé. Avec les protocoles asynchrones, un nœud peut se réveiller quand il veut et tant qu'il est capable de communiquer avec ses voisins. Ce but est atteint par des propriétés impliquées dans le régime ***sleep/wakeup***, aucun échange d'informations n'est alors nécessaire entre les nœuds. Quelques régimes ***sleep/wakeup*** asynchrones sont proposés dans.

III.9-1 Application de la mise en veille dans un protocole

L'éteinte d'un module (tranceiver, CPU, ou sensor) est la définition basique de la mise en veille, pour que cette technique soit efficace et bénéfique dans la préservation d'énergie d'un nœud et plus d'un réseau de capteurs, les différentes contraintes soumises aux WSNs doivent être respectées.

Voici un exemple **Figure III-3** qui illustre l'utilisation de cette technique.

Modèle de mise en veille



-Figure III-3: SDEM (Sensor Dynamic Energy Management)-

Le principe de cette technique est ***d'allumer les nœuds*** quand ils reçoivent une commande d'acquisition, et ***éteindre la radio*** quand les nœuds sont en mode ***Idle***. Ce protocole est utilisé dans les réseaux de capteurs conçus pour l'agriculture ; chaque nœud possède deux radios, l'une moins consommatrice en énergie, elle s'occupe d'envoi ou réception des demandes, ensuite la radio la plus consommatrice se réveille pour envoyer un paquet de données, ensuite ce remettre en veille après l'envoi.

III.10 Bénéfices d'utilisation de mise en veille

Pour sauvegarder le plus d'énergie, il est intuitif de ***mettre en veille*** des nœuds capteurs (tranceiver) à chaque fois que possible. Cela implique que :

(1) les nœuds à source redondants peuvent dormir lorsque leurs données ne sont pas requises par le puits.

(2) des nœuds de routage redondants peuvent dormir quand ils n'ont pas de données à router.

(3) les nœuds de source et les nœuds de routage qui sont impliqués dans la livraison des données peuvent également dormir quand ce n'est pas leur tour de transmettre ou recevoir des données.

En conséquence, les systèmes de ***mise en veille*** existants sont mis en œuvre dans de telle manière que les différentes couches effectuent leurs tâches.

Spécifiquement, la couche application sélectionne et active uniquement certains nœuds de la source pour réduire la fuite d'énergie tout en préservant la qualité de service désirée [**88**]. Par exemple, dans l'application de [**88**] suivi de cible, tous les capteurs au à une certaine distance de l'objectif pourraient être des nœuds source, mais seulement quelques-uns sont finalement activés par la couche application pour obtenir une faible distorsion et réduire la consommation d'énergie.

D'autre part, les nœuds de routage, sont généralement sélectionnés et activés par la couche réseau de deux manières :

- ***La première*** est appelée contrôle de topologie, qui vise à établir un chemin de routage pour garantir la connectivité de réseau, par conséquent les nœuds non routés peuvent ***dormir***. Le contrôle de topologie met également à jour les chemins de routage périodiquement en fonction de l'énergie restante de chaque capteur pour équilibrer la consommation d'énergie.

  - **Par exemple**, GAF [**56**] est un protocole de contrôle de topologie, qui divise un réseau en zones virtuelles avec un seul capteur dans chaque Grille activé à tout moment, constituant un

réseau dense pour acheminer toutes les données. Le contrôle de topologie est souvent utilisé à grande échelle, des réseaux denses où de nombreux nœuds fournissent des voies redondantes. Donc les capteurs ***mis en veille*** peuvent augmenter la durée de vie du réseau.

- ***La seconde*** manière de ***faire dormir les capteurs*** à la couche réseau est par protocoles de routage. Les protocoles de routage [**34**] [**76**] mettent à jour régulièrement les routes, et sauvegardent l'énergie en ***mettant en veille*** les nœuds de routage qui ne sont pas impliqués dans la livraison des données.

  - **Par exemple**, la technique décrite par [**85**] ***met en veille*** les capteurs en surveillant le routage des messages de contrôle et les transmissions de données de sorte que seuls les routeurs utiles sont maintenus en activité. La technique décrite par [**76**] a également fait appel au routage par décision afin que les nœuds de capteurs ne ***se réveillent*** pas quand ils ne font pas partie d'un chemin de routage.

Par rapport à la topologie de contrôle, ces protocoles de routage avec ***le sommeil*** ne gaspillent par l'énergie non-utilisé des nœuds routés, et ils sont également convenables pour des réseaux à petite échelle où le contrôle de topologie fonctionne mal.

Une fois les nœuds source et les chemins de routage sont déterminés, la couche MAC peut placer des capteurs dans le cycle de ***sommeil-réveil*** [**49**] de sorte que l'Idle listening peut être encore réduite en raison du fait que le trafic de charge dans les réseaux de capteurs sans fil n'est généralement pas très élevé.

Par exemple les protocoles suivants déjà invoqués dans le chapitre précédent

- SMAC [**51**] permet à chaque capteur de diffuser son ***calendrier sommeil-réveil*** de temps en temps, afin que ses capteurs voisins puissent l'entendre et synchroniser leurs calendriers avec lui. Une fois que les capteurs sont synchronisés, ils utilisent les *séquences RTS / CTS / DATA / ACK* pour communiquer pendant la période de ***réveil*** de chaque cycle.

- BMAC [**49**], utilise l'écoute périodique à faible consommation, il force un capteur à ***se mettre en veille*** si rien n'est détecté sur les médias pendant la durée de réveil du capteur. Cependant, le bénéfice d'une plus langue durée de vie fournit par la couche MAC accompagné de la ***mise en veille*** provoque une grande latence.

Donc la ***mise en veille*** peut être appliquée sur la couche application, la couche réseau et la Couche MAC individuellement, et même plus elle peut aussi être appliquée à plusieurs couches. Dans le but de gagner plus d'énergie.

III.11 Mise en veille au niveau de la couche de routage

Une étude à permet de montrer les a tous empotés par la mise en veille au niveau de la couche de routage, l'essai à été appliqué au protocole Directed Diffusion l'un des protocoles de routage.

III.11-1 Directed Diffusion

Comme nous l'avons déjà invoqué dans le chapitre I; Directed Diffusion comprend deux phases, une étude exploratoire et une phase de renforcement, qui, ensemble, permettent à un nœud puits d'obtenir les données souhaitables à partir des nœuds source. La phase exploratoire est utilisée pour découvrir les sources de données à un faible débit, tandis que la phase de renforcement est utilisée pour baisser les données souhaitables à un taux élevé de données. Le fonctionnement de ce protocole est expliqué dans le chapitre précédant dans la partie des protocoles de routage.

Directed Diffusion ne permet pas aux capteurs de ***se mettre en veille***. Toutefois, les nœuds de routage ne sont pas tous impliqués dans la livraison des données tout le temps, car seul le chemin de routage à moindre retard est un renforcement positif. Ainsi, en permettant aux nœuds de routage redondants de ***dormir***, on peut sauver d'autres chemins de routage et contribuer à une plus longue vie de l'application on minimisant la consommation d'énergie.

III.11-2 Sleeping Directed Diffusion

Pour ***mettre en veille*** les nœuds de routage quand ils ne sont pas impliqués dans la livraison des données, il est nécessaire de noter l'importance de diffusion du paquet d'intérêt et les données exploratoire, et calculer les séquences d'établissement du temps et l'entretien de chemin dans Directed Diffusion. Comme mentionné ci-dessus, dans Directed Diffusion il y a deux diffusions critiques dans l'ensemble du réseau. L'une est la diffusion périodique d'un paquet d'intérêt, initiée par le nœud puits.

Tous les nœuds de routage doivent être éveillés pour transmettre le paquet afin que les nœuds source puissent le recevoir, et une table de routage peut être mise en place pour acheminer les paquets de données qui vont suivre. Les autres inondations sont des données exploratoires, périodiquement initiées par un nœud source. Tous les nœuds de routage doivent être éveillés pour transmettre ces paquets de telle sorte que le nœud puits puisse les recevoir et commencer le renforcement positif en fonction de leur arrivée.

On a deux timers à chaque nœud, un timer d'intérêt et un timer de données, pour un transfert *intérêt* et un transfert de Données exploratoire respectivement. Les deux timers sont programmées et lancés périodiquement. Spécifiquement, le *timer d'intérêt* est prévue après la fin d'inondation du paquet d'*intérê*t, et se déclenche avant la prochaine inondation *intérêt*, tandis que le timer de données se déclenche après qu'une diffusion de données exploratoires se termine.
Un capteur peut ***se mettre en veille*** quand les deux timers sont en attente, mais il faut se réveiller une fois qu'un timer se déclenche, et rester éveillé jusqu'à ce que le timer soit changé.

Il ya un intervalle entre le moment où un timer se déclenche et quand il est reportée. L'écart de temps est utilisé pour attendre la réponse de l'inondation correspondante afin d'établir un chemin renforcé à partir du noeud puits pour le noeud source. L'établissement du chemin commence par la diffusion du message d'information, suivie par la diffusion des données exploratoires, qui est ensuite suivie par le renforcement d'unicasting positif, et par conséquent suivie par un taux élevé données unicasting.

Ainsi, l'écart de temps du *timer* des paquets d'information est utilisé pour attendre la diffusion des données exploratoires à chaque découverte d'une nouvelle source de nœuds, tandis que l'écart de temps du *timer* de données est utilisé pour attendre les paquets de renforcement positif afin que l'un des nœuds source puisse être renforcé et commencer à fournir des données haut débit.

Si un nœud ne reçoit aucun paquet de données exploratoires au cours de l'écart de temps du *timer* du paquet d'information en raison d'absence de nœud source disponible ou de perte de paquets, le *timer* exploratoire de données n'est pas initié. Ainsi, le nœud suit le *timer* des paquets d'information pour ***se mettre en veille*** (quand il est en instance) et de se réveiller (quand il expire).

Sinon, le *timer* de données exploratoires est lancé, et il est ordonnancé périodiquement en fonction du débit exploratoire. Si le nœud reçoit un paquet de renforcement positif pendant l'intervalle de temps du *timer* de données, le nœud se trouve sur le chemin de renforcement (impliqués dans la livraison des données), et donc ne peut pas ***se mettre en veille*** jusqu'à ce qu'il soit mal renforcé ou qu'il n'a plus d'énergie. Si le nœud ne reçoit aucun paquet de renforcement positif lors de la période du *timer* de données, le nœud va ***se mettre en veille*** lorsque les deux *timers* sont en attente (pas de diffusion), et se réveille jusqu'à ce que au moins l'un des *timers* expire (à la dernière diffusion faite).

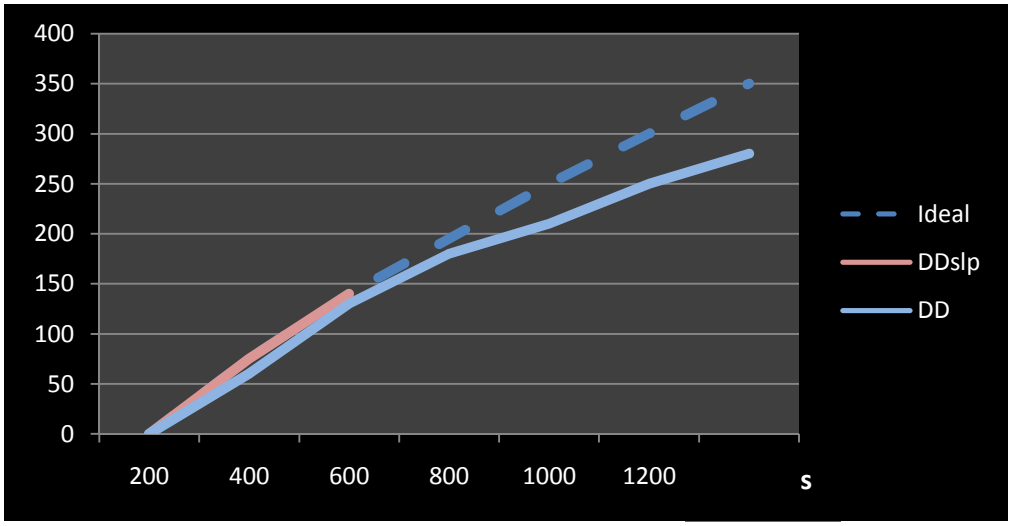


Figure III-4 : Performance de sleeping Directed Diffusion par rapport a Directed Diffusion[89]-

III.12 Mise en veille au niveau de la couche MAC

Pour montrer les atouts de l'utilisation de la *mise en veille* au niveau de la couche MAC une comparaison entre un protocole MAC utilisant la technique de mise en veille SMAC et un autre protocole sans *mise en veille* à été réalisé par [**89**].

III.12-1 SMAC

Est un protocole MAC explicitement proposé pour les réseaux de capteurs sans fil, avec réduction de la consommation d'énergie qui est son principal objectif. Il introduit le *sommeil/réveil* périodique pour chaque nœud pour éliminer l'idle-listening. Utilisé pour la transmission et réception des données, la période de *mise en veille* peut avoir une taille fixe, qui est déterminé par les paramètres de la couche physique et MAC.

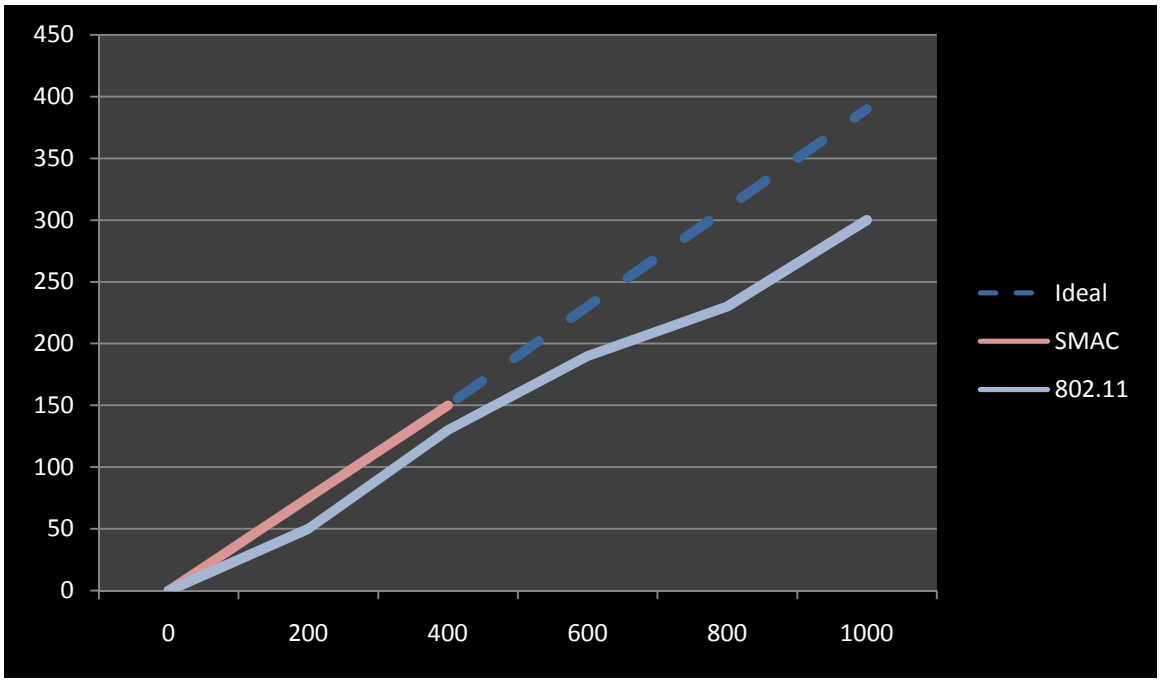
La période de *mise en veille* d'un cycle peut être grande ou petite, ce qui influence la consommation d'énergie de SMAC, ainsi que la latence encourus par *la mise en veille*. Par conséquent, les variations dans le duty cycle, qui est définie comme le rapport de la période d'*éveil* à un *sommeil* complet et cycle de réveille, conduit à des variations correspondantes dans la performance de SMAC.

Pour la commodité de la communication, et de la réduction du contrôle de l'overhead, SMAC essaie de synchroniser tous les nœuds, de sorte que les nœuds peuvent *dormir* et *se réveiller* simultanément. Pour y parvenir, chaque nœud diffuse périodiquement son calendrier dans un Paquet SYNC, de sorte que les ordonnancements des autres voisins peuvent être écoutés. Le début de paquet SYNC suivant le même ordonnanceur. Toutefois, certains nœuds ne peuvent pas entendre les paquets SYNC de leurs voisins parce qu'ils ont déjà exécutés différents ordonnancements. Ainsi, chaque nœud doit rester éveillé pour un intervalle de temps d'un ensemble des paquets SYNC, de sorte que les différents ordonnancements de ses voisins peuvent être entendus. Un nœud qui a différents ordonnancements de ses voisins peut suivre deux calendriers en même temps.

Pendant le temps éveillé, SMAC est similaire à la norme IEEE 802.11, qui :

(1) utilise RTS / CTS pour résoudre le problème de terminal caché.
(2) utilise le support physique de détection et porteuse virtuelle de détection pour éviter la collision.
(3) utilise RTS / CTS / DATA / ACK séquences de garantir des transmissions en monodiffusion succès.

Si un nœud ne parvient pas à accéder aux médias, il se rendort jusqu'à la prochaine période d'éveille. Si, d'autre part, un nœud accède aux médias avec succès, il ne dort pas jusqu'à ce qu'il finisse la transmission courante.



-Figure III-5 : **Performance de SMAC par rapport a IEEE 802.11[89]-**

Ces schémas Figure III-4 et Figure III-5 nous montre les performances de la technique de mise en veille par rapport aux protocoles, la conclusion tirée sur ces exemples peut également être appliquée à d'autres protocoles de routage, les protocoles qui *mettent en veille* les capteurs quand ils ne sont pas impliqués dans la transmission de données, et d'autres protocoles MAC qui utilisent des cycles pour économiser de l'énergie.

Avant de conclure ce chapitre sur la mise en veille et son utilisation dans les différentes couches et protocoles WSN, nous citons que :

- ***la mise en veille*** dans les protocoles de routage couche de routage est plus adapté pour les réseaux à haute redondance ou une forte contention.

- ***la mise en veille*** dans la couche MAC est plus sensible aux contentions, et est donc un bon choix pour les applications à trafic léger et les réseaux à petite échelle.

Par conséquent, une décision intelligente peut être faite par un gestionnaire d'énergie pour choisir dynamiquement le *schéma de mise en veille* dans la couche de routage ou bien dans la couche MAC ou les deux avec la coordination d'inter couche(corss-layer).

III.13 Conclusions

Dans ce chapitre nous avons défini le principe de *mise en veille* qui est l'une des méthodes les plus adaptées pour la préservation d'énergie dans les WSN ; nous avons également classifié *la mise en veille* selon les différents cas d'utilisation et aussi selon les différentes couches.

Nous avons pris quelques résultats des différents cas d'utilisation de *la mise en veille* pour permettre de voir la performance aboutie.

Notre dernier chapitre est l'évaluation du gain d (énergie de la mise en veille à travers deux applications, l'une utilise la mise en veille et l'autre sans mise en veille. La simulation est effectuée sur le simulateur TOSSIM de TinyOS.

Sommaire

IV.1 Introduction

Pour étudier le comportement d'un réseau soumis à des contraintes précises selon l'environnement concerné ; une plate forme de simulation qui imite l'environnement réel et produit les mêmes contraintes à été développé pour rendre l'étude des réseaux de capteurs plus efficace et moins couteuse.

Nous allons utiliser dans notre exemple la simulation d'un réseau de capteur de température ; qui nous permettra d'évaluer les résultats de ***mise en veille*** sur la consommation d'énergie ; qui concerne la problématique posée dans notre sujet.

Comme nous l'avons vu dans les chapitres précédents les réseaux de capteurs ont plusieurs contraintes, dont l'énergie est l'un des points que les différentes recherches et études rencontrés dans la littérature invoquent et asseyent de minimiser la consommation afin de maximiser la durée de vie des WSN.

L'utilisation de la ***mise en veille*** des capteurs est l'une des techniques la plus adapté que nous avons rencontré dans les diverses solutions que la recherche a réussi à proposer pour ce domaine, cependant l'efficacité de cette techniques doit être étudié selon les différentes contraintes et types de réseaux ; elle peut être appliqué a divers protocoles de différentes couches.

Nous allons dans ce qui suit, comparer les résultats de consommation d'énergie d'une application de capture de température, sans et avec mise en veille, aussi nous allons présenter des résultats et des proposer des perspectives.

IV.2 Environnement d'étude

IV.2-1 Evaluation des performances

Il existe trois techniques d'évaluation de performances d'un système : la méthode analytique, par simulation et par mesure sur un système réel existant. Ces trois méthodes diffèrent par le coût, la durée du temps nécessaire et l'exactitude des résultats [93]

A) Analytique

Il s'agit de réduire le système en un modèle mathématique et de l'analyser numériquement. L'approche analytique est parfois rapide à réaliser, mais présente le souci de la représentation fidèle du système. Il est parfois très complexe, voire impossible, de modéliser le comportement réel du système mathématiquement. Généralement, on pose des hypothèses qui simplifient l'étape de modélisation du système et rendent l'évaluation numérique faisable. Ces hypothèses simplificatrices peuvent toucher à la fidélité de représentation du système, mais permettent toutefois de traduire son comportement approché.

B) Mesure

Il s'agit de faire des mesures et de les analyser directement sur un système réel. Cette technique permet de comprendre le vrai comportement du système. Faire des mesures sur des systèmes réels n'est pas toujours possible, car ça pourrait gêner le fonctionnement du système ou aussi pour des problèmes de coûts (système non encore existant, instruments de mesure complexes, etc.). Les résultats issus de la mesure ne sont pas génériques et ne reflètent qu'une seule trajectoire du système.

C) Simulation

Il s'agit d'implanter un modèle simplifié du système à l'aide d'un programme de simulation adéquat. C'est une technique largement utilisée pour l'évaluation des performances.
Elle présente l'avantage par rapport aux méthodes analytiques de traduire d'une manière plus réaliste le comportement du système à évaluer. On procède généralement à la simulation pour évaluer un nouveau système ou bien pour des raisons de coûts d'évaluation par des mesures réelles (simulation spatiale ou aéronautique). En plus, la simulation permet de visualiser les résultats sous forme de graphes faciles à analyser et à interpréter.

IV.2 -2 Caractéristiques des environnements de simulation de réseaux sans fil

Un simulateur réseau ou à événements discrets se caractérise par le fait que les changements d'états dans le réseau simulé (événements) se produisent à des instants répartis de manière discrète sur l'axe de temps. Classiquement, une simulation consiste à reproduire le comportement du système complet dans un environnement synthétique où le temps est simulé par une horloge à événements discrets. La simulation demande une modélisation complète, depuis les applications jusqu'au réseau physique, ce qui peut se révéler à la fois complexe et coûteux en terme de temps d'exécution et d'utilisation mémoire dans le cas de réseaux de taille importante. En général, un simulateur réseau doit avoir les caractéristiques suivantes : [92]

A) Génération de la topologie du réseau

Chaque simulateur a son mécanisme pour générer la topologie du réseau. Plusieurs possibilités existent: un scripte ou un langage de configuration spécifique au simulateur, des fichiers textes ou via des interfaces graphiques. Le simulateur doit aussi permettre la création de topologies hiérarchiques et la génération automatique de topologies aléatoires.

B) Génération et la gestion du trafic réseau

Afin de générer le trafic de données circulant sur l'environnement de simulation, il est nécessaire d'utiliser des générateurs de données suivant une distribution qui doit être la plus représentative du trafic réel et qui permettra d'établir les statistiques nécessaires pour tirer des conclusions.

C) Contrôle du réseau

Pendant l'exécution, il est très utile de contrôler dynamiquement le réseau soit par flux de données ou par noeuds via des interfaces graphiques. Les traces et les résultats du contrôle peuvent êtres sauvegardés dans des fichiers afin de faire des comparaisons ou ré exécuter la simulation avec un paramétrage plus affiné.

E) Modèle protocolaire, de mobilité et de propagation radio

Un bon simulateur dans son modèle protocolaire doit disposer de plusieurs protocoles dans les différentes couches de communication. Dans certains cas pour les réseaux sans fil, nous avons

besoin de gérer la mobilité des noeuds. Cela se fait généralement par l'intégration d'un modèle de mobilité dans l'environnement de simulation. De plus, le modèle de la propagation radio pour les réseaux sans fil a un grand impact sur les résultats de la simulation, donc le simulateur doit fournir plusieurs modèles de propagation radio qui doivent être les plus proches possible du cas réel.

F) Flexibilité et extensibilité

Le simulateur doit donner la possibilité de contrôler facilement les composants dans différents niveaux (changer leurs états par exemples). De plus, les modèles disponibles peuvent êtres remplacés, modifiés ou enrichis par l'ajout de nouveaux modèles sans perturber le bon fonctionnement du simulateur. D'une manière générale, l'architecture du simulateur doit être modulaire et ouverte.

G) Temps d'exécution

Le temps d'exécution est un facteur qui doit être pris en considération dans le choix du simulateur et surtout s'il s'agit de simuler des réseaux de grandes tailles comme c'est le cas pour les

H) Visualisation des résultats et statistiques

L'environnement de simulation doit être le plus ergonomique que possible et doit faciliter les tâches de configuration et de suivie des différentes étapes de la simulation via des interfaces graphique. De plus, une représentation graphique des résultats et statistiques à travers des diagrammes, graphes ou sur des cartes géographiques est très importante afin de permettre une meilleure interprétation des résultats.

IV.2-3 Les systèmes d'exploitation dédiés aux réseaux de capteurs

La technologie des réseaux de capteurs fait partie de celle des systèmes embarqués, a cause de leurs caractéristiques : mobilité, taille réduite et source limité ; les OS (Operating System) qu'on leurs dédie sont réduits et consomment moins d'espace mémoire et d'énergie.
Parmi ces OS nous citons :

A) FreeRTOS : est un noyau de système d'exploitation temps réel pour système embarqué et portable sur plusieurs microcontrôleurs.

B) TinyOS : est le premier système d'exploitation spécialement dédié aux réseaux de capteurs. C'est un OS *open source* initialement conçu à l'Université de Berkeley en Californie et qui connait une notoriété dans le domaine des réseaux de capteurs.

C) NutOS : c'est un OS temps réel open source conçu pour système embarqué. Il est multitâche et offre une pile TCP/IP. Il a constitué une base pour ETHERNUT ou encore pour l'OS BTNut utilisé dans les BTnodes et où une pile Bluetooth open source a été rajoutée.

D) Contiki : est un système d'exploitation *open source*, léger, multitâche, générique développé pour tout système embarqué ayant des contraintes mémoire. La portabilité de ce système va de l'ordinateur ayant une architecture 8 bits aux systèmes embarqués sur des microcontrôleurs comme certaines plateformes pour RdC.

E) Mantis OS: est aussi un OS pour réseaux de capteurs entièrement écrit en C et développé par une équipe de l'université du Colorado. Ce système utilise une approche différente des autres OS pour réseaux de capteurs sans fil qui est *thread driven execution model* différent de TinyOS qui utilise un *event driven model*.

F) SOS: est un OS pour réseaux de capteurs utilisant le même modèle, *non-preemptive event driven scheduler*, que TinyOS mais écrit en C. Il a été développé à l'université de Californie, Los Angeles (UCLA).

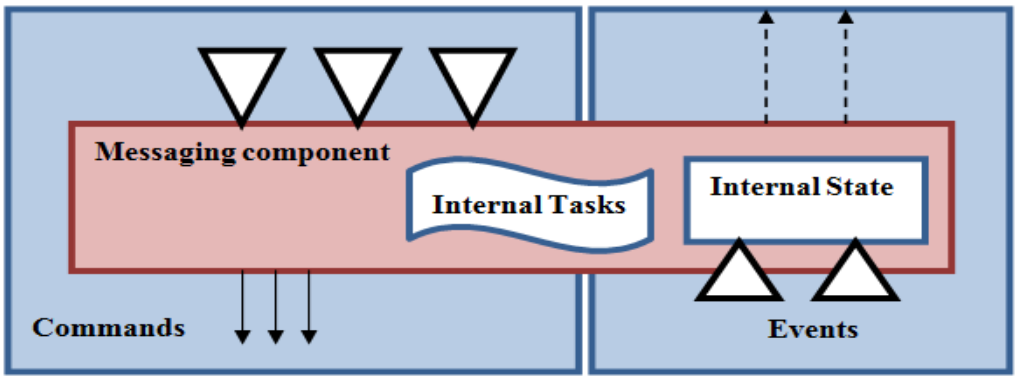
IV.2-4 Le système utilisé pour cette simulation (Tinyos)

A) Présentation

TinyOS[94] [96] un système d'exploitation open-source conçu pour des réseaux de capteurs sans fil. Il respecte une architecture basée sur une association de composants, réduisant la taille du

code nécessaire à sa mise en place. Cela s'inscrit dans le respect des contraintes de mémoires qu'observent les réseaux de capteurs. Pour autant, la bibliothèque de composant de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. L'ensemble de ces composants peut être utilisé tel quel, il peut aussi être adapté à une application précise.

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



- Figure IV-1 : Schématisation d'un composant TinyOS-

B) Caractéristiques

- **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. Ainsi, l'ensemble des sources sont disponibles pour de nombreuses cibles matérielles.

- **Event-driven**

  Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des évènements se produisant. 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 évènementiel *(event-driven)* s'oppose au fonctionnement dit temporel *(time-driven)* où les actions du système sont gérées par une horloge donnée.

- **Langage**

Comme nous l'avons évoqué plus haut, TinyOS a été programmé en ***langage NesC*** que nous allons détailler plus loin.

- **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 peut stopper l'exécution d'une tâche.

- **Temps réel**

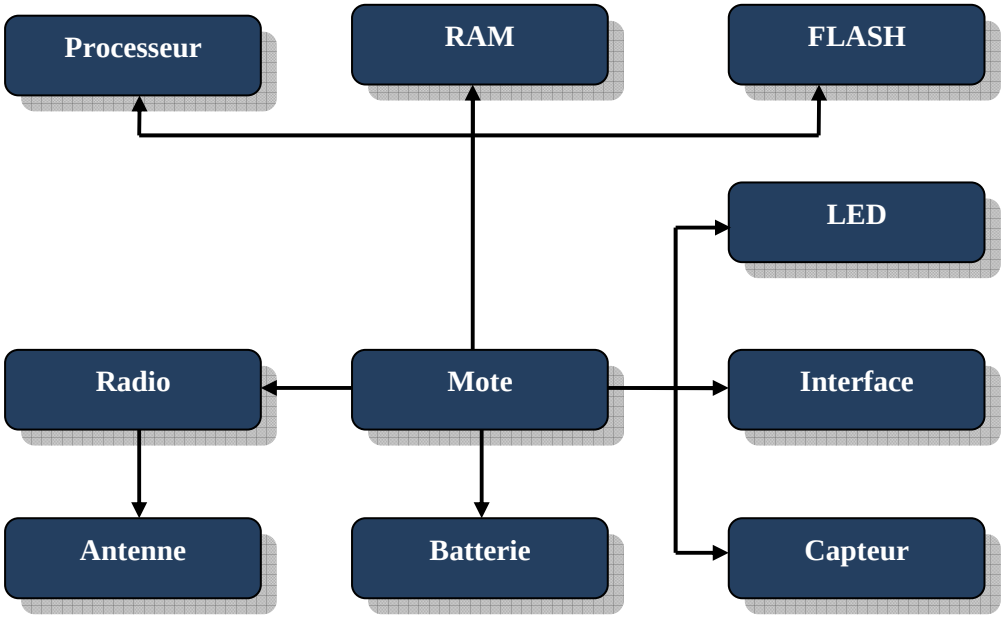
  Lorsqu'un système est dit « *temps réel* » celui-ci gère des niveaux de priorité dans ses tâches permettant de respecter des échéances données par son environnement. Dans le cas d'un système strict, aucune échéance ne tolère de dépassement contrairement à un système temps réel mou. TinyOS se situe au-delà de ce second type car il n'est pas prévu pour avoir un fonctionnement temps réel.

- **Consommation**

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

- **Cibles de TinyOS**

  Il existe de nombreuses cibles possibles pour ce système d'exploitation embarqué. Malgré leurs différences, elles respectent toutes globalement la même architecture basée sur un noyau central autour duquel s'articulent les différentes interfaces d'entrée-sortie, de communication et d'alimentation. Voici un schéma représentant cette architecture :



- Figure IV-2: Architecture de Tinyos-

1. ***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.

2. Radio et antenne

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

3. LED, interface, capteur

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

4. Batterie

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

C) Concepts de base de TinyOS

TinyOS[97], est construit autour des différents concepts décrits ci-dessous :

Les composants : constitués de :

- **Frame :** est un espace mémoire de taille fixe, permettant au composant de stocker les variables globales, et les données, qu'il utilise. Il n'en existe qu'un seul par composant ;
- **Tâches :** contiennent l'implémentation des fonctions. Elles sont décomposées en deux catégories : les commandes et les évènements ;
- **Les interfaces :** représentent le descriptif des fonctions définies dans les tâches ; TinyOS, est un ensemble de components logiciels, qui peuvent être reliés ensemble en un seul exécutable sur un senseur, comme illustré dans la figure.

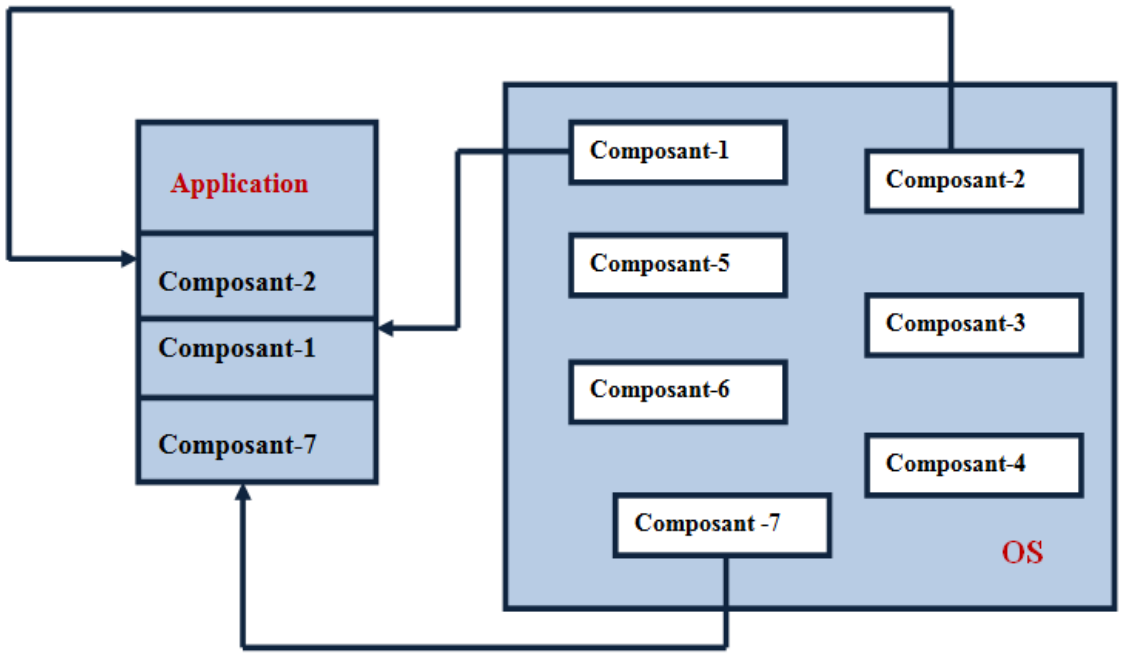


Figure VI-3 : TinyOS un ensemble de composants logiciels-

D) Allocation de la mémoire

Il est important de préciser de quelle façon un système d'exploitation aborde la gestion de la mémoire, d'autant plus lorsque ce système travaille dans un environnement aussi restreint. sa

distribution minimale. En plus de cela, il est nécessaire d'avoir 4 Ko de mémoire libre qui se répartissent de la façon suivante :

· ***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 « crashs » et corruptions de la mémoire.

E) L'ordonnanceur TinyOS

Le choix d'un ordonnanceur déterminera le fonctionnement global du système et le dotera de propriétés précises telles que la capacité à fonctionner en évènementiel.
L'ordonnanceur TinyOS comporte :

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

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

F) Plateformes compatibles avec TinyOS

TinyOS est prévu pour fonctionner sur diverses plateformes. En effet, TinyOS peut aussi bien être installé sur Windows, Linux, Mac OS ou sur un capteur. Nous avons procédé à l'installation sur les deux plateformes à notre disposition, c'est à dire Windows xp et Linux (distribution Ubuntu). Procédure d'installation :

- ***Windows :*** un guide propose l'installation de tous les principaux outils nécessaires au bon fonctionnement du système, notamment Cygwin (couche d'émulation de l'API Linux) qui permet d'avoir une interface Unix sous Windows. Il est à noter que le JDK Java 1.4 de Sun est nécessaire afin d'effectuer la procédure d'installation.[voir annexe]

- ***Linux :*** L'installation de TinyOS s'est révélée relativement longue et laborieuse sous Ubuntu, la version de Linux que nous avions déjà installée sur nos machines.[***voir annexe***]

IV.2-5 Le langage nesC

A) 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).

Les composants NesC présentent des similarités avec des objets. Les états sont encapsulés et on peut y accéder par des interfaces. En NesC, l'ensemble des composants et leurs interactions sont fixés à la compilation pour plus d'efficacité. Ce type de compilation permet d'optimiser l'application pour une exécution plus performante. En langage objet, cette phase est réalisée lors de l'exécution ce qui rend celle ci plus lente. [**98**]

IV.2-5 B) Développement

Dans la pratique, NesC permet de déclarer 2 types de fichiers: ***les modules et les configurations.*** Le fichier configuration est la définition du ou des composants qui seront utilisés

par l'application déployée sur le capteur. Par exemple, l'application ci dessous utilise le composant *LedsC*, ce qui permet de manipuler les *leds du capteur*. Les interfaces sont des fichiers décrivant les commandes et évènements proposés par le composant qui les implémente. L'utilisation des mots clefs « Use » et « Provide » au début d'un composant permet de savoir respectivement si celui ci fait appel à une fonction de l'interface ou redéfini son code. Pour suivre l'exemple précédent, il est possible de redéfinir dans ce fichier les fonctions de manipulation des leds.

Les concepts de base du nesC sont :

1. ***Séparation de construction et de composition :*** les programmes sont construits à partir de composants, qui sont assemblés ("câblés") pour former des programmes complets. Les composants sont divisés en 2 blocs, un pour leur spécification (contenant les instances des interfaces), et un pour leur implémentation. Les composants sont en concurrence interne sous la forme de taches. Les threads de commande peuvent passer d'un composant à un autre au travers des interfaces. Ces threads peuvent être appelés soit dans une tache soit par interruption matérielle.
2. ***Spécifications du comportement du composant en termes de groupe d'interfaces :*** Il existe 2 groupes d'interfaces :

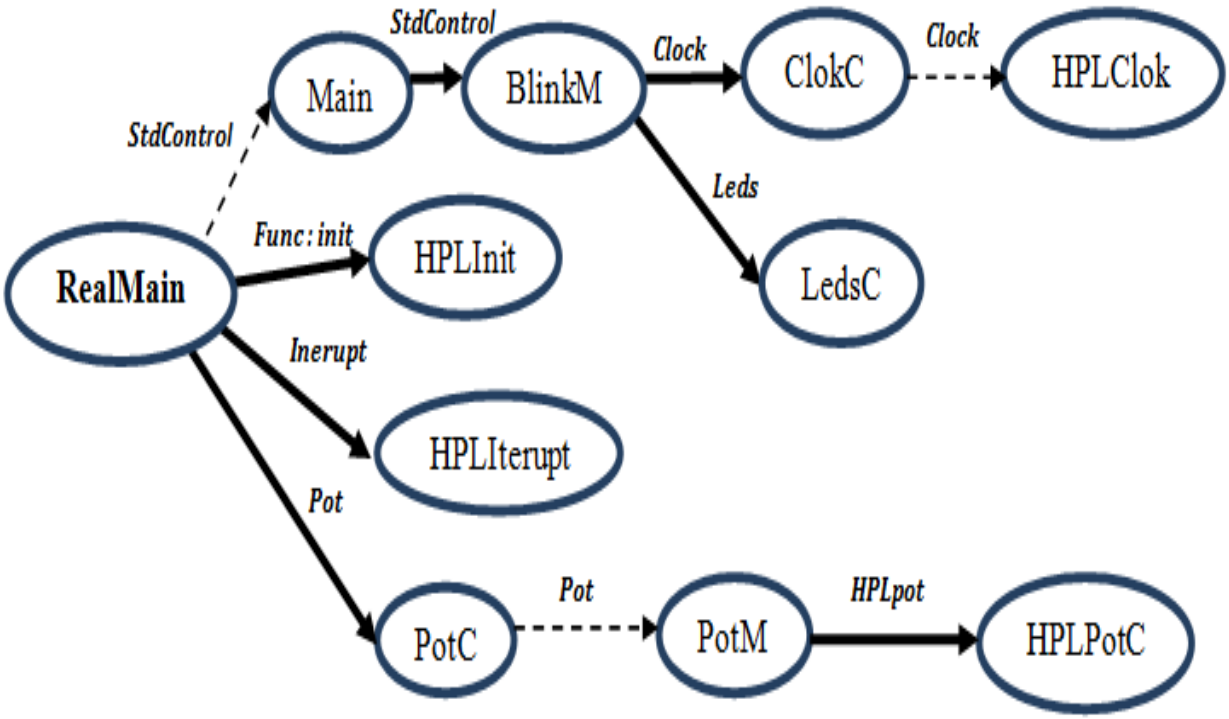
Les interfaces fournies ou les interfaces utilisées par le composant. Les interfaces fournies sont prévues pour représenter la fonctionnalité que le composant fournit à son utilisateur, les interfaces utilisées représentent la fonctionnalité que le composant a besoin pour réaliser son travail.

3. ***Les interfaces sont bidirectionnelles :*** les interfaces spécifient un ensemble de fonctions à implémenter par le fournisseur de l'interface (*commandes*) et un ensemble à implémenter par l'utilisateur de l'interface (*événements*). Ceci permet à une simple interface de représenter une interaction complexe entre les composants. C'est crucial car toutes les très longues commandes dans TinyOS sont non bloquantes ; leur accomplissement est signalé par un événement. Donc un composant qui fait appel à une commande doit implémenter une interface qui reçoit l'événement d'accomplissement de la commande appelée. Typiquement les commandes appellent vers les niveaux bas, c.-à-d., depuis des composants d'application vers ceux plus près du matériel, alors que les événements appellent vers les couches supérieures. Certains événements proviennent d'interruptions matérielles.

4. ***Des composants sont statiquement liés entre eux par l'intermédiaire de leurs interfaces.***
   Ceci augmente la vitesse d'exécution, encourage une conception robuste, et tient compte d'une meilleure analyse statique des programmes.

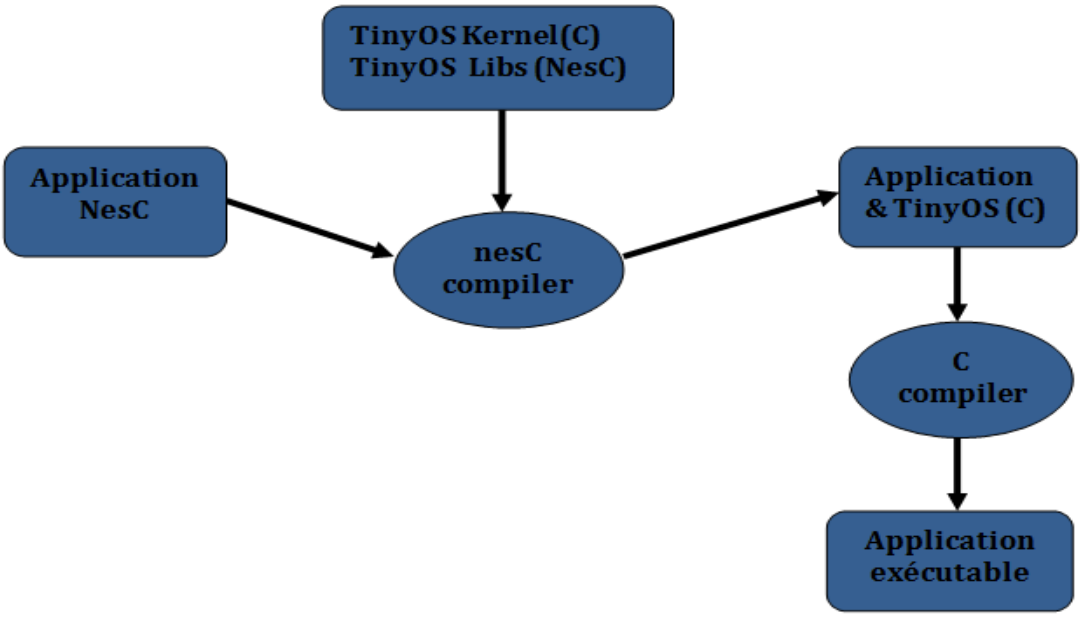
Le nesC est conçu dans l'optique que le code sera produit par des compilateurs de programme complet. Ceci tient compte d'une meilleure génération et analyse du code.

L'image ci-dessous montre la structure (les composants et leur câblage) d'une application simple qui allume une LED chaque seconde :



-Figure VI-4 : Structure de l'application Blink-

C) Compilation



-Figure VI-5 : Processus de compilation-

Les capteurs possèdent en général la même architecture et sont programmables via le langage ***NesC***. Cependant plusieurs firmes fabriquent de tels capteurs, ce qui implique quelques différences qui sont palliées par le compilateur de ***NesC*** appelé ***ncc***. En effet celui ci propose une compatibilité pour les plates formes de capteurs les plus répandues.

Pour effectuer la compilation, les deux fichiers vus dans la partie précédente, doivent se situer dans le même répertoire contenant aussi un ***makefile*** de la forme :

```
COMPONENT=Blink  //nom de l'application
include ../Makerules
```

Ce ***Makefile*** permet de compiler le composant ***Blink*** en spécifiant en paramètre la plateforme sur laquelle doit fonctionner l'application. Par exemple, pour un capteur de type ***mica2***, la commande permettant de compiler l'application sera : ***make mica2***. Le compilateur ***ncc*** offre aussi la possibilité de pouvoir compiler l'application pour l'utiliser sur TOSSIM, le simulateur de TinyOS. Dans ce cas là, la commande sera : ***make pc.*** Cette commande génère un exécutable « ***main.exe*** » dans l'arborescence **/repertoire_courant/build/pc.**

IV .2-6 Les outils de simulation tinyOS

TinyOs offre des outils de Simulation *(TinyViz/Tossim/PowerTossim)*

A) Le simulateur dédié à TinyOS : « TOSSIM »

Pour arriver à simuler le comportement des senseurs, au sein d'un WSN, un outil très puissant, a été développé, et proposé pour TinyOS, baptisé TOSSIM

- ***TOSSIM** (**T**iny**OS SIM**ulator)*

Est un simulateur discret, basé sur la programmation par événements, et qui a été conçu, et désigné, pour simuler les réseaux de senseurs, qui utilise la plateforme TinyOS. Le principale but de *TOSSIM*, est de créer une simulation très proche, de ce qui ce passe, dans ces réseaux, dans le monde réel (*envoie/réception)* de messages via les ondes radios, traitement de l'information, etc. au sein d'un réseau de senseurs). Au lieu de compiler, l'application directement dans le senseur, l'utilisateur, peut le faire dans le cadre *TOSSIM (programmation niveau réseau),* qui s'exécute sur un PC, permettant ainsi, d'examiner leur code TinyOS à l'aide des débogueurs, et d'autres outils de développement *(test, analyse des algorithmes),* dans un environnement contrôlé, et reproductible. En d'autres mots, *TOSSIM*, permet de tester, valider, et simuler le fonctionnement d'un réseau de senseurs *(envoie/réception de messages via les ondes radios, traitement de signal, etc*.).

TOSSIM, 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. De plus, *TOSSIM* contient un modèle abstrait, de chaque composant matériel d'un noeud. Pour une simulation *TOSSIM*, on utilise le même code, que celui destiné au noeud cible, mais cette fois ci, *TOSSIM* émule le comportement du matériel, en utilisant les modèles des composants.
Simuler exactement le code, qui tournera sur les noeuds, permet de tester l'implémentation finale, des algorithmes. Cette notion d'abstraction du matériel, est tout à fait intéressante.

TOSSIM, fourni deux modèles de radios, pour la communication. Le modèle par défaut, est celui *simple*. Les paquets, sont transmis dans le réseau, avec aucune erreur, et ils sont reçus par chaque noeud. Avec ce modèle, il est ainsi possible, que deux noeuds différents, peuvent envoyer

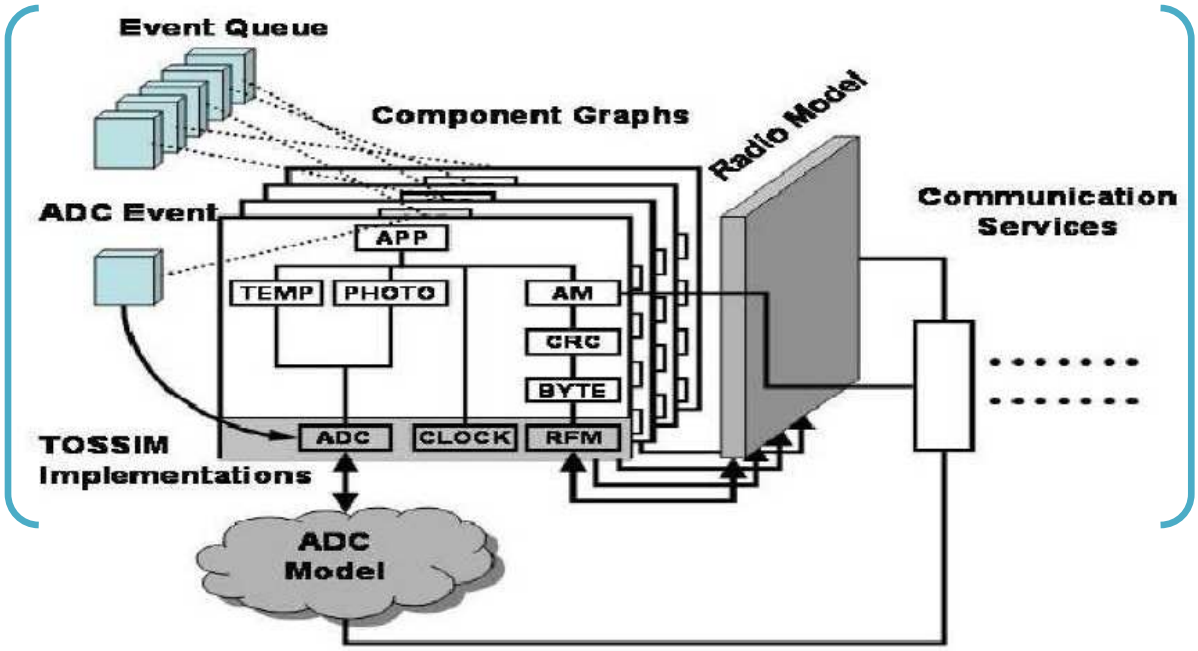
un paquet, en même temps avec la conséquence, que ces deux paquets seront alors détruits, due au chevauchement des signaux. Le deuxième modèle, est le modèle *lossy*.

Dans ce modèle, les noeuds sont placés dans un graphe dirigé, formé d'un couple (a, b) ce qui signifie, qu'un paquet envoyé, par le noeud « a » peut être reçu par le noeud « b » .

Pour une compréhension moins complexe, de l'activité du réseau, *TOSSIM*, utilise une interface graphique, écrite en Java : TinyViz, pour visualiser, de manière intuitive le comportement de chaque senseur, au sein du réseau.

Cette interface graphique, est équipée par plusieurs *APIs plugins*, qui permettent d'ajouter plusieurs fonctions, à notre simulateur, comme par exemple, suivre la dépense d'énergie, en utilisant un autre simulateur, qui s'appelle PowerTOSSIM.

Ils sont opérationnels, dès lors que TinyOS fonctionne. Le schéma suivant illustre l'architecture de TOSSIM :



-Figure VI-6 : Architecture intérieure du simulateur TOSSIM-

Comme toutes les autres technologies, TOSSIM présente des avantages, et des limites, dont les principales:

- **Avantages**

TOSSIM permet de simuler fidèlement, le comportement d'un réseau de capteurs. Il permet également, l'affichage des évènements, et des messages de débogage, pour chaque senseur, mais aussi, simultanément pour l'ensemble des senseurs. A cela, se rajoute l'interface graphique TinyViz, qui permet la visualisation des échanges radios, conjointement aux messages de débogage, ce qui permet, à chaque instant, de la simulation, d'avoir une vue globale de l'activité du réseau étudié. TinyViz, offre la possibilité, de ralentir la simulation, par un délai, afin d'observer le déroulement des évènements, ce qui est très intéressant, lorsque le réseau est surchargé de messages.

TOSSIM permet de simuler fidèlement, le comportement d'un réseau de senseurs, puisque le code est écrit, en respectant l'implémentation de TinyOS. Donc, une fois la simulation est terminée, le code peut être exécuté, directement dans un senseur, utilisant le système TinyOS, avec un petit changement.

TOSSIM prend en charge la couche MAC, l'encodage de données, le timing, et les acquittements de synchronisation.

- **Limites**

TOSSIM fournie une abstraction de certains phénomènes du monde réel.

1. ***La Radio:*** il ne modélise pas la propagation dans la radio, à la place, il fournie une abstraction de l'indépendance directe, de taux d'erreurs entre deux noeuds.

2. ***Puissance/énergie:*** il ajoute des annotations aux composants, qui consomment de l'énergie, pour fournir des informations concernant le changement de l'état énergétique.

3. Il exige l'exécution du même code, pour tous les noeuds. Donc, il ne distingue pas les différents types de senseurs (senseur maître, senseur esclave).

4. Manque de souplesse, et d'extensibilité, puisqu'il est lié à une interface de visualisation conçue séparément.

La plateforme, aussi fidèle soit elle, ne peut prendre en compte tous les phénomènes réels. Par exemple, prenons le cas, d'un réseau de senseurs, qui mesure les températures sur une zone. Il se peut, que les senseurs soient déposés au sol par lâché aérien. Sur le nombre de senseurs, certains seront détruits à l'arrivée sur le sol, d'autre pourront rester coincés, dans un arbre ou d'autres tombés dans une crevasse, ce qui rendra le réseau beaucoup moins performants, car le nombre de senseurs actifs sera diminué.

- **Compilation et exécution d'une simulation**

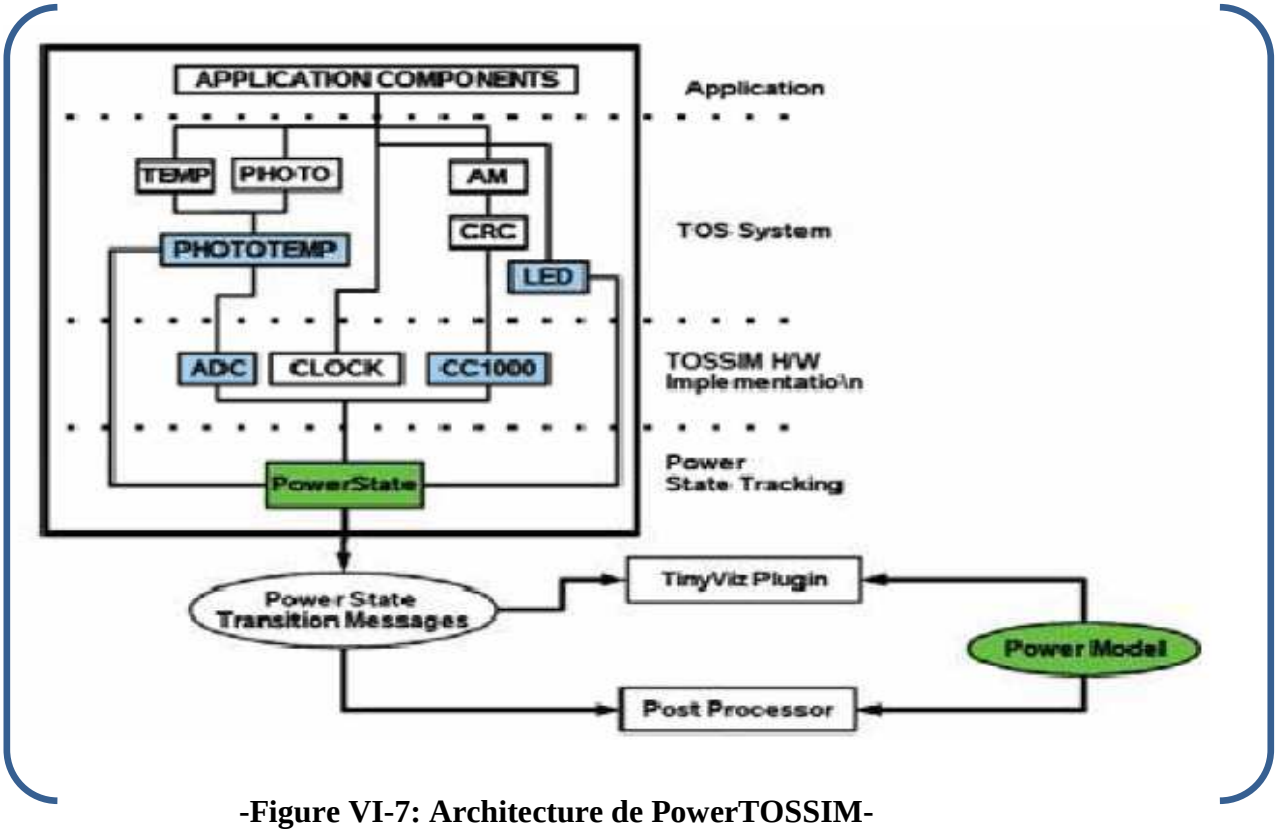
TOSSIM est crée automatiquement, lors de la compilation d'une application. Les applications sont compilées, dans le répertoire d'applications (le répertoire *apps* ; par ex. */apps/blink*), en tapant la commande *make*, ou lors de compilation de la simulation, pour l'application en tapant la commande :

make pc

Il existe plusieurs options de compilation (sont dans le fichier makefile de TinyOS1.1) pour le compilateur *nesC (ncc)*, lors de la compilation de *TOSSIM*, y compris le nombre maximum de noeuds, qui peuvent être simulés. L'exécutable TOSSIM est baptisé ***main.exe***, et est dans le répertoire *build/pc*.

B) 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 WSN 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 générer un fichier de l'extension ***.trace,*** qui enregistre les détails de la simulation, comme l'énergie consommée dans le réseau .



-Figure VI-7: Architecture de PowerTOSSIM-

PowerTOSSIM, est l'extension de TOSSIM, qui contient un modèle de consommation d'énergie. Pour les valeurs de consommation, les auteurs se sont basés sur le ***Mica2*** (nœud développé à l'université de Berkeley). Les auteurs connaissent les consommations des différents composants, de ce noeud suivant leurs états. Il faut donc, connaitre l'état de chaque composant d'un noeud, pendant la simulation. Grace au modèle de simulation, basé sur TinyOS, on connait immédiatement l'état des composants, autres que le microcontrôleur.

Puisque, les changements d'états, correspondent à des évènements dans TinyOS, et donc, dans TOSSIM. Plusieurs composants du noeud, sont parfois abstraits dans TOSSIM, par un seul composant. L'estimation de la consommation du microcontrôleur, est plus délicate : il faut instrumenter le code, pour être capable de compter le nombre d'exécution de chaque bloc d'instruction, et il faut, faire correspondre chaque bloc d'instruction, avec son code en assembleur. Lors de la simulation, on note le nombre de passage, d'exécution, de chaque bloc d'instructions.

Sachant combien d'instructions élémentaires, contient chaque bloc de base, on en déduit le nombre d'instructions effectuées, par le microcontrôleur, et donc sa consommation. Cette approche, est intéressante, mais elle ne permet pas de varier la précision du modèle de consommation. Enfin,

les simulateurs ***TOSSIM,*** et ***PowerTOSSIM*** ne conviennent, que pour des applications écrites en TinyOS.

IV .2-7 Le modèle radio (tranceiver)

Le modèle radio utilisé dans notre simulation est TMote2 ; TOSSIM nous offre une trace du fonctionnement de cette radio tout au long de la durée de simulation ; et PowerTOSSIM nous offre une estimation de consommation d'énergie de ce composant en évaluant le fichier à extension « **.trace** » produit par TOSSIM.

LPM :Low Power Mode	**Durée de transition (ms)**		**L'énergie moyenne de transition(mA)**		**L'énergie moyenne consommée (mA)**	
Capteur	***TMote***	***MICAz***	***TMote***	***MICAz***	***TMote***	***MICAz***
Réception	N.D	N.D	N.D	N.D	21.56	21.97
Transmission	N.D	N.D	N.D	N.D	18.40	19.70
LPM1 :IDLE	4.56	4.38	3.72	3.04	0.627	0.743
LPM2 :Power down	5.15	5.58	2.96	2.94	0.179	0.298
LPM3 :Power off	6.81	5.87	1.88	3.20	0.038	0.190

Tableau VI-1 : Caractéristiques des deux radios TMote et MICAz (ND :Non Disponible, LPM :Low Power Mode)

Le tableau fournit des détails sur les caractéristiques des deux radios *TMote* et *MICAz*. Plusieurs observations peuvent être faites. Premièrement, l'énergie consommée dans *la transmission* ou la *réception* des données est largement plus grande que si on garde la radio en état *en veille*. Donc, la clé pour une gestion effective de l'énergie sera dans l'éteinte de la radio. Deuxièmement, le temps nécessaire pour passer du mode ***"en veille"*** au mode ***"active"*** est considérable. Donc, le nombre de changements de mode doit être minimisé.

IV.3 Implémentation et simulation

Le test que nous allons présenter est une comparaison entre la consommation d'énergie dans le cas général, ou les radios des nœuds capteurs sont toujours allumées et consomment de l'énergie dans leurs trois modes « transmission, réception et Idle» tout au long de la période de simulation.

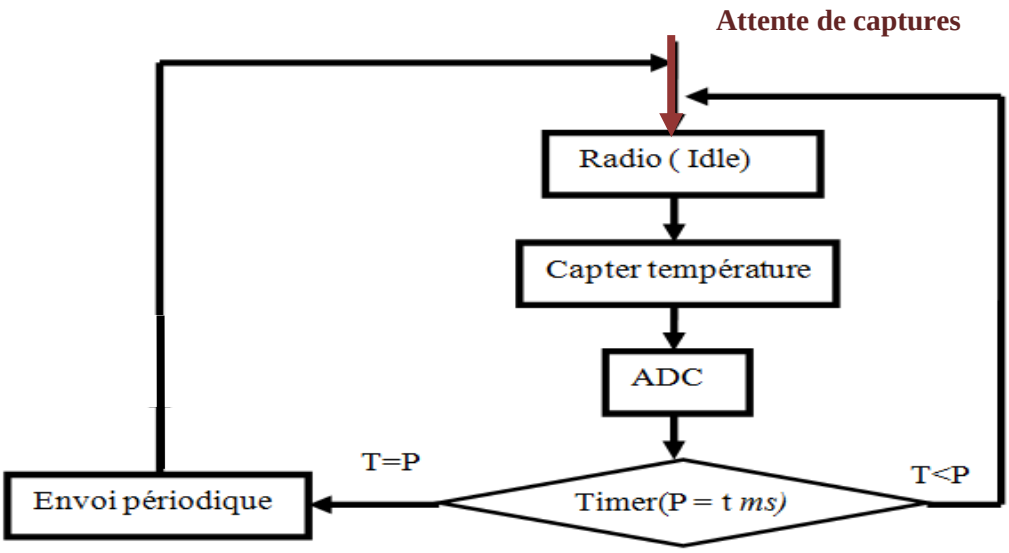
Et le cas où on implémente la technique de préservation choisi : ajouter le mode « sleep » pour les 3 états de la radio ; ainsi pendant la phase Idle qui est consommatrice en énergie ; on éteint le module radio, ce qui nous permet un gain d'énergie. Cependant le choix de l'intervalle de mise en veille n'est pas aléatoire car :

- Avec une longue période de *mise en veille* on risque la perte de données, si un événement se produit pendant la période de *sommeil* ; le nœud senseur ne peut pas capter l'information. Ni recevoir un paquet envoyé par un autre nœud *(donc ne participe pas dans le routage de données)* ce qui provoque parfois la latence dans le cheminement.
- Avec une courte période de *mise en veille* ; on peut éviter la perte de données et aussi le problème de latence lors de l'enchainement vers le nœud puit*(sink),* sauf que la consommation d'énergie à chaque allumage de la radio « pour stabiliser les composants » peut faire perdre plus d'énergie à pendant une période donnée.
- Donc on essaye de trouver une période de *mise en veille* idéale pour éviter les deux contraintes.

Les résultats des tests si après nous portent plus d'éclaircissement.

VI-3.1 Présentation de l'application

Nous avons choisi pour la simulation un exemple d'application (capteurs de températures), nous avons réalisé une application simple qui permet à un nœud d'envoyer ou recevoir des températures transmises en valeurs numériques par l'ADC pour qu'elles soient traitées.la **Figure VI-8** est un schéma qui résume le fonctionnement du nœud pour effectuer la procédure d'envoi d'une température captée.



-Figure VI-8: transmission d'une temperature -

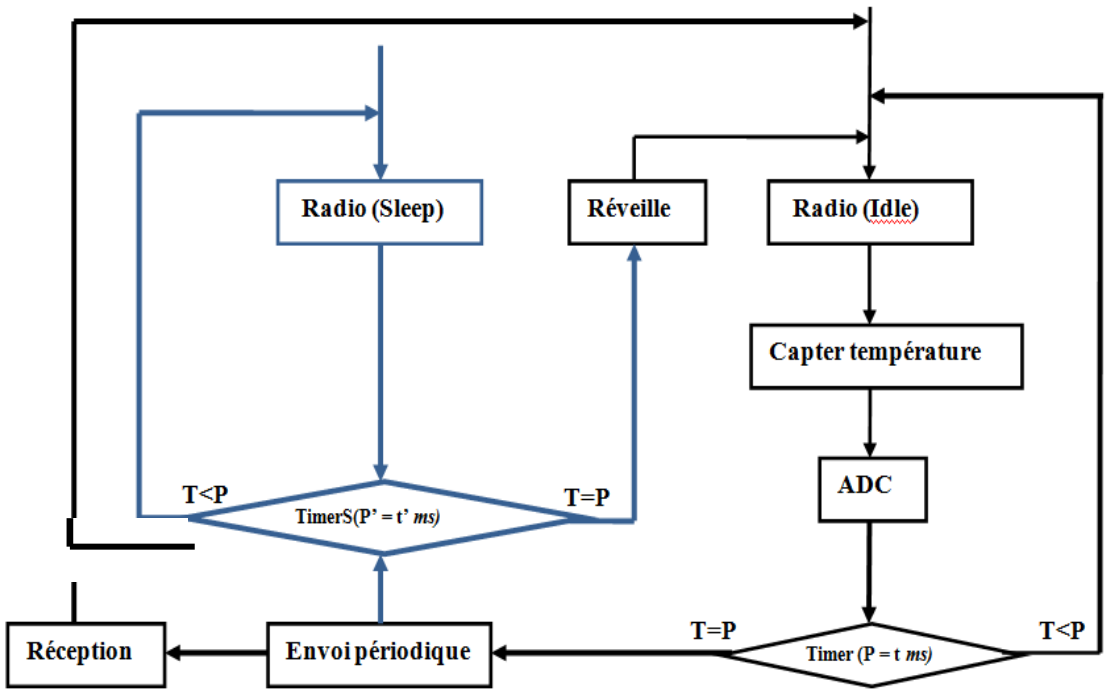
Algorithme : Capture et envoi périodique de la température par un capteur

Begin
capter temperature()
atendre_timer()
if *est_synchronise*() **then**
{t ←relever_temperature()
envoyer_temperature(*t;1*) //le nœud 1 reçoit les températures captées par les autres noeuds
afficher_temperature(t)
}
end if

Recevoir_temperature(t')
End.

VI-3.2 Application avec mise en veille des capteurs

Dans l'application avec mise en veille, nous avons ajouté une procédure qui permet au capteur de dormir après avoir envoyé une température cela pendant une durée de temps T. le capteur se réveille dès que la période sleeping s'achève **Figure VI-9** pour se remettre à nouveau à l'écoute, et refaire la même procédure après chaque envoi d'un paquet.



-Figure VI-9 : Mise en veille du module radio après envoi de paquets-

Algorithme : Capture et envoi périodique de la température par un capteur avec mise en veille

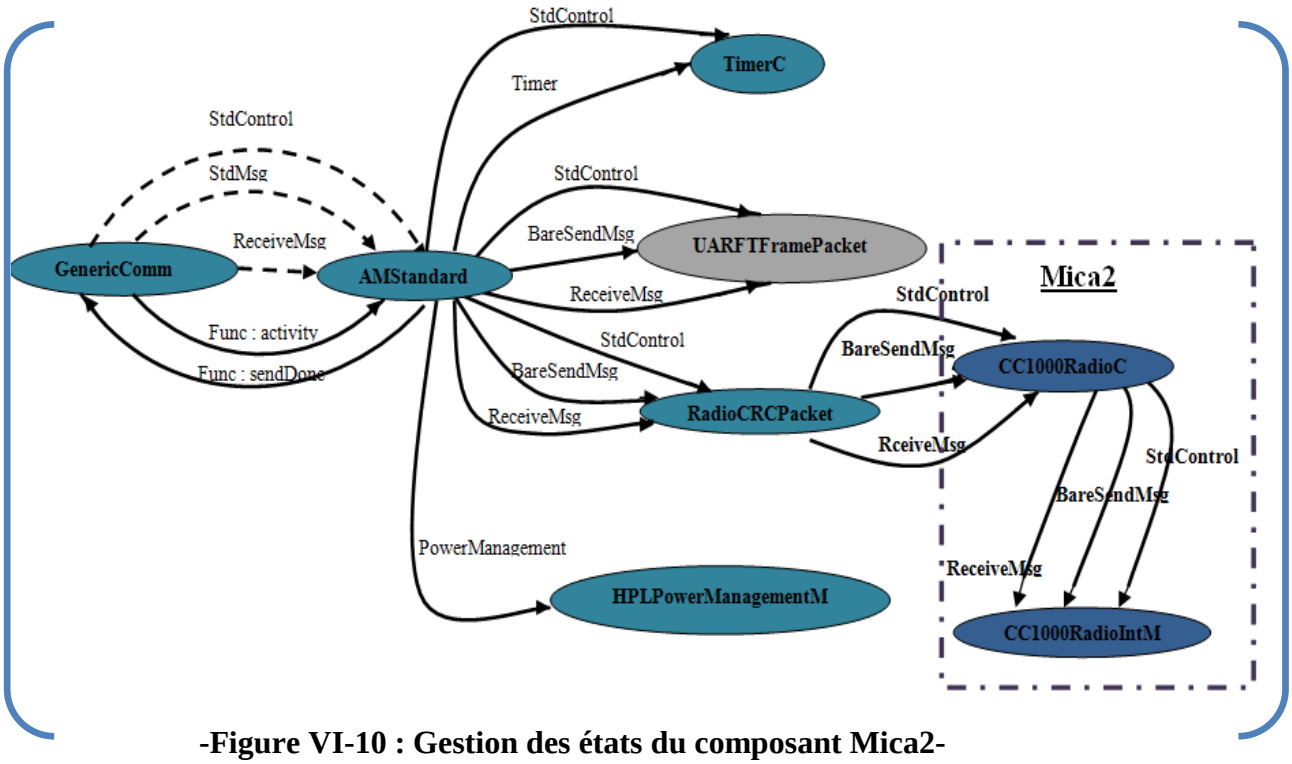
Begin
capter temperature()
attendre_timer() // ***le timer défini pour envoi de données captées***
attendre_timer_controle() // ***le timer défini pour la mise en veille après envoi***

if *est_synchronise*() **then**
{t ←relever_temperature()
envoyer_temperature(*t;1*) // ***le nœud 1 reçoit les températures captées par les autres noeuds***
afficher_temperature(t)

dormir(period) // ***la radio du nœud se met en veille après envoi d'une température***
}
end if
Recevoir_temperature(t')
End.

VI-3.3 Contrôle du module radio

Ce schéma **Figure VI-10** nous montre la programmation de gestion du module radio de Mica2 sous tinyos, ce module est un exemple d'applications éxistantes sous tinyos qui facilitent la programmation grace à la caractéristique de tinyos qui fonctionne sur le modèle de composant.



-Figure VI-10 : Gestion des états du composant Mica2-

VI-3.4 Les composants de l'application en NesC

Cette partie du programme représente le fichier de configuration de l'application de mise en veille que nous avons réalisé. Le composant HPLPowerManagementM est le module qui s'occupe de la gestion d'énergie, nous allons faire appel à ce dernier pour éteindre et allumer le module radio.

```
Configuration Temperatur{
implementation {
 components Main, TemperaturM, LedsC, Temp, TimerC, GenericComm as Comm, HPLPowerManagementM as PM;

 Main.StdControl -> TemperaturM.StdControl;
 Main.StdControl -> TimerC;
 Main.StdControl -> Comm;
 TemperaturM.ADC -> Temp;
 TemperaturM.PowerManagement -> PM;
 TemperaturM.Enable -> PM.Enable;
 TemperaturM.CommControl -> Comm;
 TemperaturM.Timer -> TimerC.Timer[unique("Timer")];
 TemperaturM.Leds -> LedsC;
 TemperaturM.ReceiveMsg -> Comm.ReceiveMsg[AM_COUNTMSG];
 TemperaturM.SendMsg -> Comm.SendMsg[AM_COUNTMSG];
}
```

A) Les interfaces de quelques composants

Pour consommer l'énergie dans cette application « capter la température d'un environnement » la gestion de la radio est la partie qui nous concerne plus ; pour cela nous allons présenter les deux composants participant à la surveillance de cet élément « tranceiver » : le temps (timer) et (les transitions « etteindre/allumer » la radio).

Ce composant permet de définir l'intervalle de temps périodique prévu pour déclencher et gérer les événements et les différentes tâches de l'application.

B) L'interface timer

Le timer permet de gérer les périodes dédiées aux applications, par exemple pour savoir la période de mise en veille, un timer doit s'occuper du lancement ou d'éteindre le capteur .

```
configuration TimerC {
  provides interface Timer[uint8_t id];
  provides interface StdControl;
}

implementation {
  components TimerM, ClockC, NoLeds, HPLPowerManagementM;

  TimerM.Leds -> NoLeds;
  TimerM.Clock -> ClockC;
  TimerM.PowerManagement -> HPLPowerManagementM;

  StdControl = TimerM;
  Timer = TimerM;
}
```

C) L'interface PowerManegement

Ce code permet d'ajuster l'état d'un composant en ce qui concerne la consommation d'énergie, dans notre cas nous allons faire appel à PowerManagement pour allumer et éteindre le composant radio, selon un intervalle de temps déterminé (timer).

```
interface PowerManagement {
  async command uint8_t adjustPower();
  async command result_t enable();
  async command result_t disable();
}
```

initialisation

```
command result_t StdControl.init(){
  ……..
  call Enable();
  call PowerManagement.adjustPower();
  call CommControl.init();
  return SUCCESS;
 }
```

D) Envoi d'un message

```
async event result_t ADC.dataReady(uint16_t temperature){
  *((int16_t *)beacon_packet.data)=temperature;
   if(start == FALSE){
    ……
    call CommControl.start();
    …..
       // une fois le modèle ADC signal l'existence d'un paquet le timer se déclenche.
    call Timer.start(TIMER_REPEAT, 500);
   }
    else if((m_sending==FALSE) && (start == TRUE)) {
     //envoi d'un parquet au noeud 1
    if (call SendMsg.send(1,sizeof(int16_t),&beacon_packet)== SUCCESS)
{
……
  dbg(DBG_USR1, "la temperature envoye est =%d\n", temperature);
}
}
```

Mise en veille du module radio

```
event result_t SendMsg.sendDone(TOS_MsgPtr msg, bool success)
 {
…….
 dbg(DBG_USR1, "mise en veille apres envoi\n");
 call Timer.start(TIMER_REPEAT,2000);  //mettre en veille la radio pour une durée de
2000ms par exemple.
 call CommControl.stop(); // allumer la radio quand la durées'achève.
……..
  return SUCCESS;
}
```

VI-4 Simulation et évaluation des résultats

A) Métriques à évaluer

- Tester la consommation d'énergie d'un nœud capteur au sain du réseau choisi pour la capture de température.
- Choix de l'intervalle de *mise en veille*.
- Le taux de gain d'énergie.
- Le taux de perte de données.
- Evaluation de la consommation d'énergie
- Montrer la différence de consommation d'énergie entre le mode *sans mise en veille* et *avec mise en veille.*
- Comparaison de la consommation d'énergie entre la technique de mise en veille et la technique de compression de données.

B) Caractéristiques de la simulation

Dans l'exemple de simulation nous avons

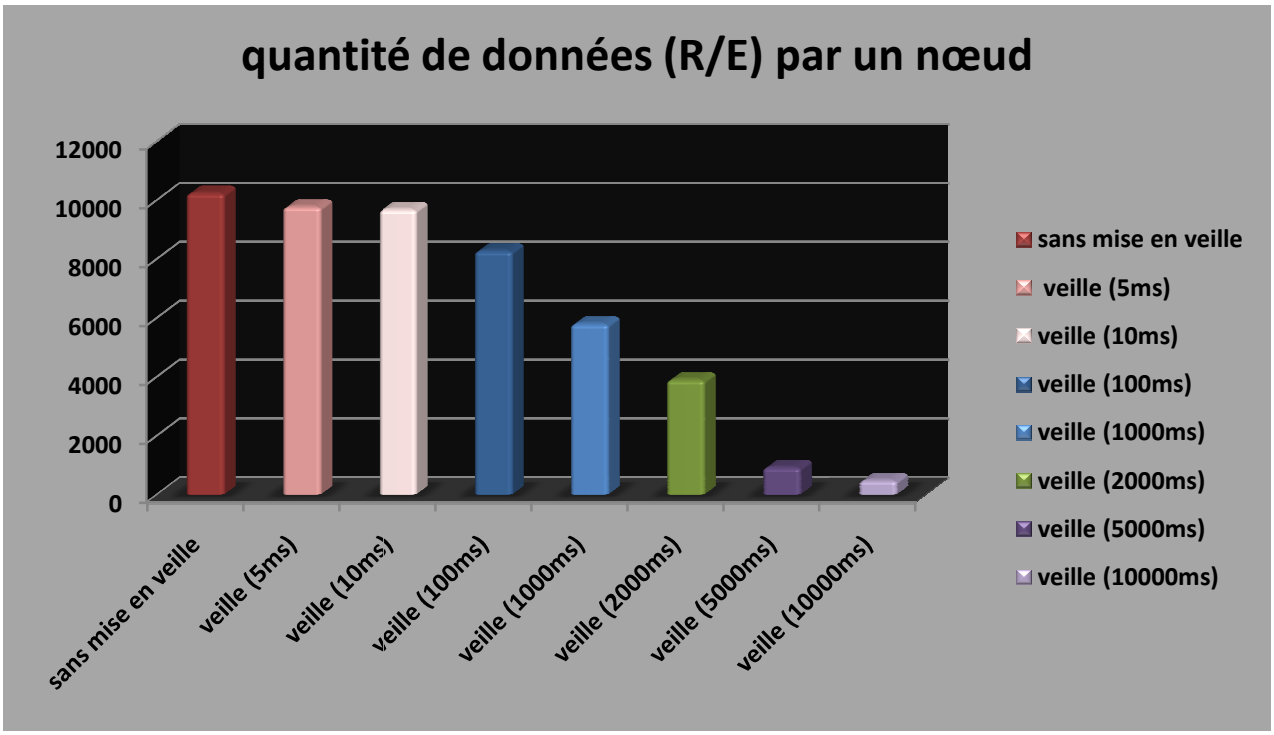
a. Le nombre d'itérations pour une simulation est 5.
b. Le temps de *mise en veille idéale* est *2000ms* après chaque envoi de message
c. Le temps de mise en *veille abusive* est *10ms*
d. Le nombre de nœuds testés *est 2*
e. *Le temps de test pour le choix d'intervalle 5 minutes.*

C) Choix de l'intervalle de mise en veille

Pour avoir une période de mise en veille idéale, qui permet le gain en énergie et la préservation des fonctionnalités du nœud et sa participation dans l'enchainement des données dans le réseau, nous avons pris plusieurs périodes pour tester le nombre de paquets enchainés par un nœud par rapport à l'énergie préservé pendant cette période. Donc nos mesures se focalisent sur :

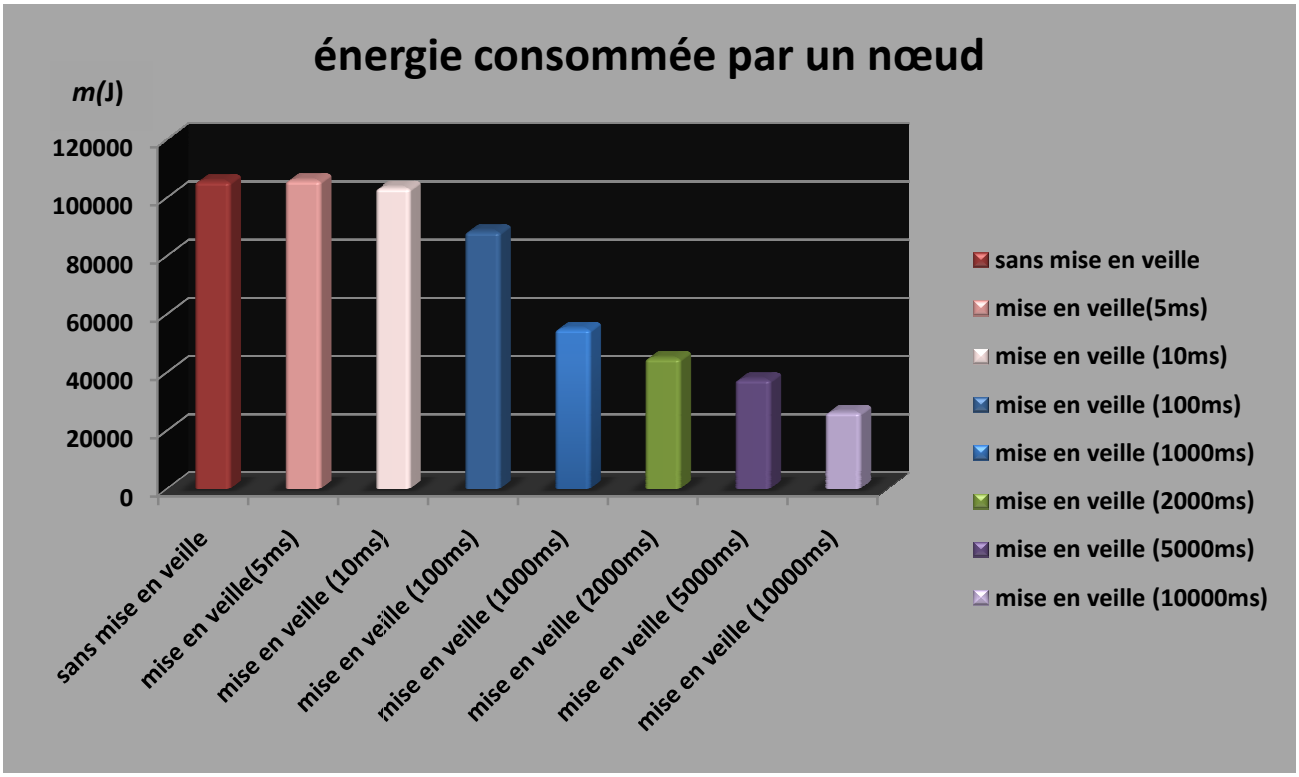
- Le nombre de données « **valeurs de température** » (envoyés/reçu) par un nœud (pendant un temps de simulation) dans le cas sans mise en veille.
- Proposition de 6 périodes de mise en veille : 5ms, 10ms, 100ms, 1000ms, 2000ms, 5000ms et 10000ms.
- Choix de la période selon le nombre de données (valeurs de température) envoyés d'un nœud à une période donnée, et le bénéfice énergétique.

VI-4.1 Choix de l'intervalle



-FigureVI-11 : quantité de données enchainées par un nœud selon la période de mise en veille-

La figure VI-14 nous montre la variation de la quantité de données échangées par un capteur dans l'application sans mise en veille et l'application avec mise en veille, en testant différents intervalles de mise en veille, nous remarquant qu'à force d'augmenter le temps de mise en veille, le nombre de données enchainées par un nœud diminue.



-Figure VI-12 :- la consommation d'énergie selon la période de mise en veille-

La **figure VI-15** nous donne les résultats des deux applications, et les différents intervalles de mise en veille, nous constatons que dans ces résultats l'énergie diminue à force d'augmenter la période de mise en veille d'un nœud.

Les petites périodes 5ms et 10 ms, ne sont pas bénéfiques, car une mise en veille abusive entre de petites périodes oblige le système qui rallume la radio à fournir une quantité d'énergie considérable à chaque réveille. Dans ce cas, il y a autant de perte que de gain d'énergie.

Par contre à force d'augmenter la période, l'énergie perdue dans l'Idle pourra être sauvegardé grâce à cette période de sommeil.

A cause de la consommation d'énergie dans les deux périodes 5ms et 10 ms (le redémarrage de la radio à chaque 5 ms/10ms) nous les avons considéré comme cas de mise en veille abusive.

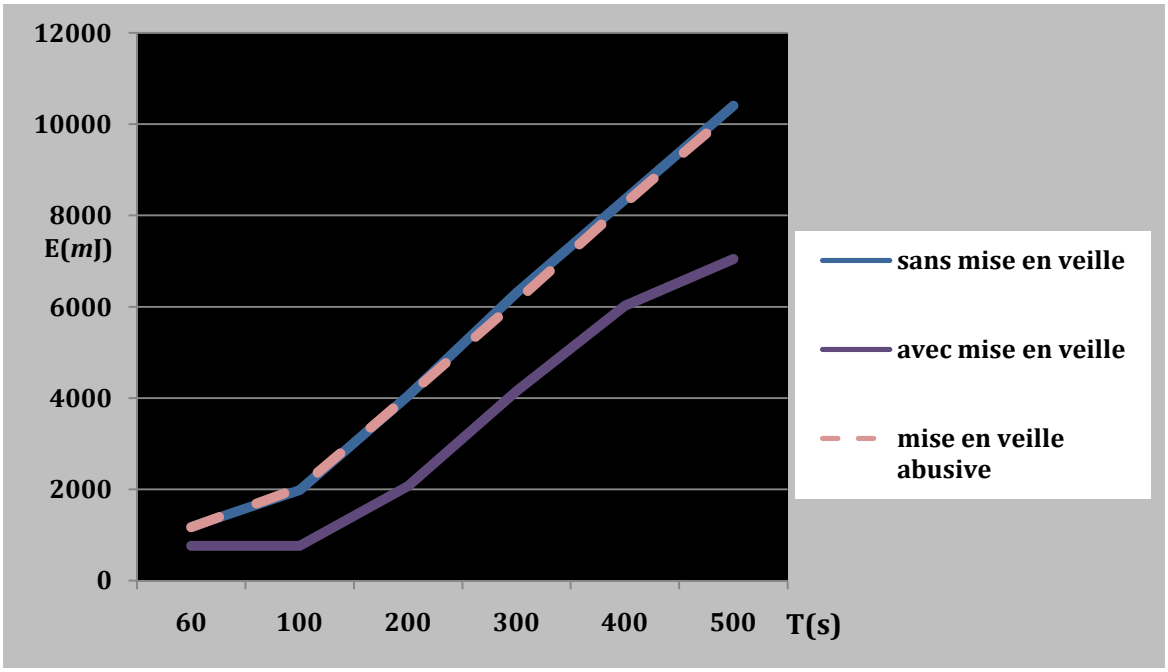
Nous avons choisi comme intervalle (**2000ms**) entre les périodes qui restent afin d'éviter une perte importante de données.

En utilisant cet intervalle ; nous estimons que l'énergie gagné par un nœud est de 38% par rapport à l'énergie consommé par un nœud qui est toujours actif dans l'application sans mise en veille.

Cependant l'estimation de la quantité de perte de donnée est de 67.3% ; c'est une valeur considérable, sauf que le contexte d'utilisation du réseau de capteur peut bénéficier de ce pourcentage. Pour justifier, nous citons par exemple :

- Dans le cas d'un réseau dense si un nœud qui est mis en veille évite 67.3% des données qu'il doit (envoyer et recevoir), nous constatons que ce n'est pas une vraie perte car dans ce type de réseau, la densité provoque la redondance et la mise en veille diminue le taux de redondance.
- Parmi les 67.3% des données perdues d'un nœud (dans l'application avec mise en veille), un pourcentage considérable représente des données qui ne le concernent pas ; dans ce cas, cette technique de mise en veille évite la sur-écoute.

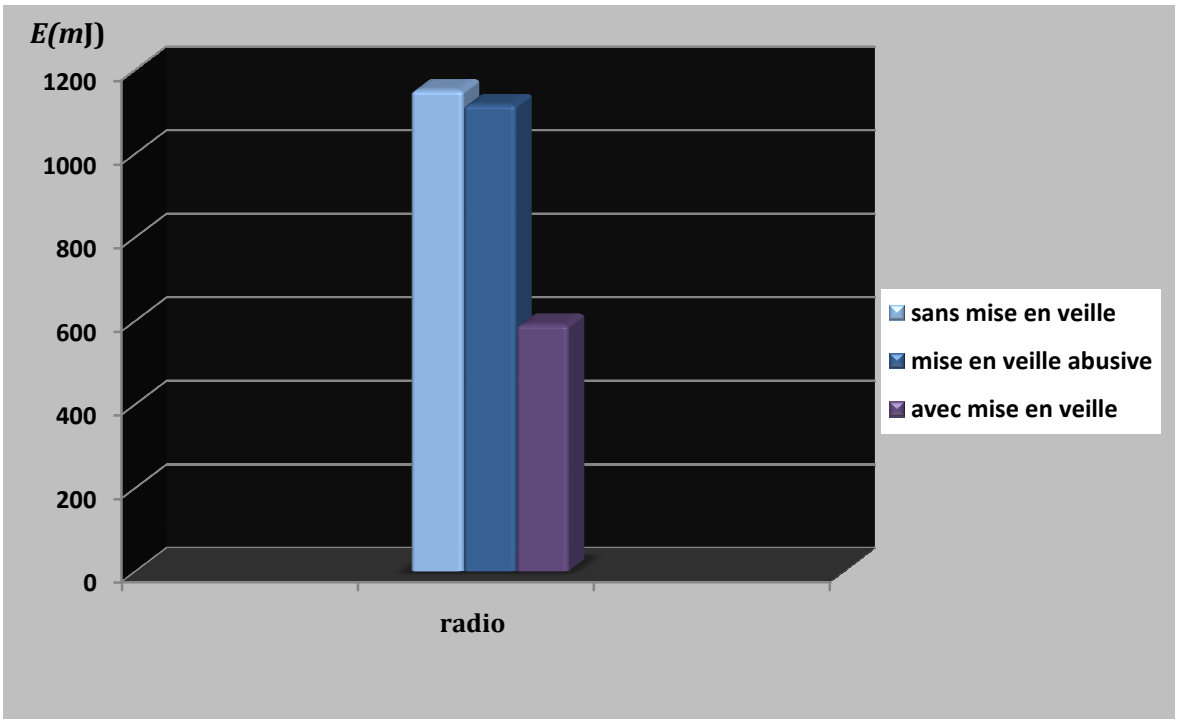
VI-4.2 Evaluation de la consommation pendant un intervalle de temps



-Figue VI-13 : déroulement de la consommation énergétique d'un capteur dans les différents cas –

Ce diagramme montre l'évaluation de la consommation d'énergie au fil du temps.

VI 4-3 Comparaison des résultats de l'application avec et sans mise en veille

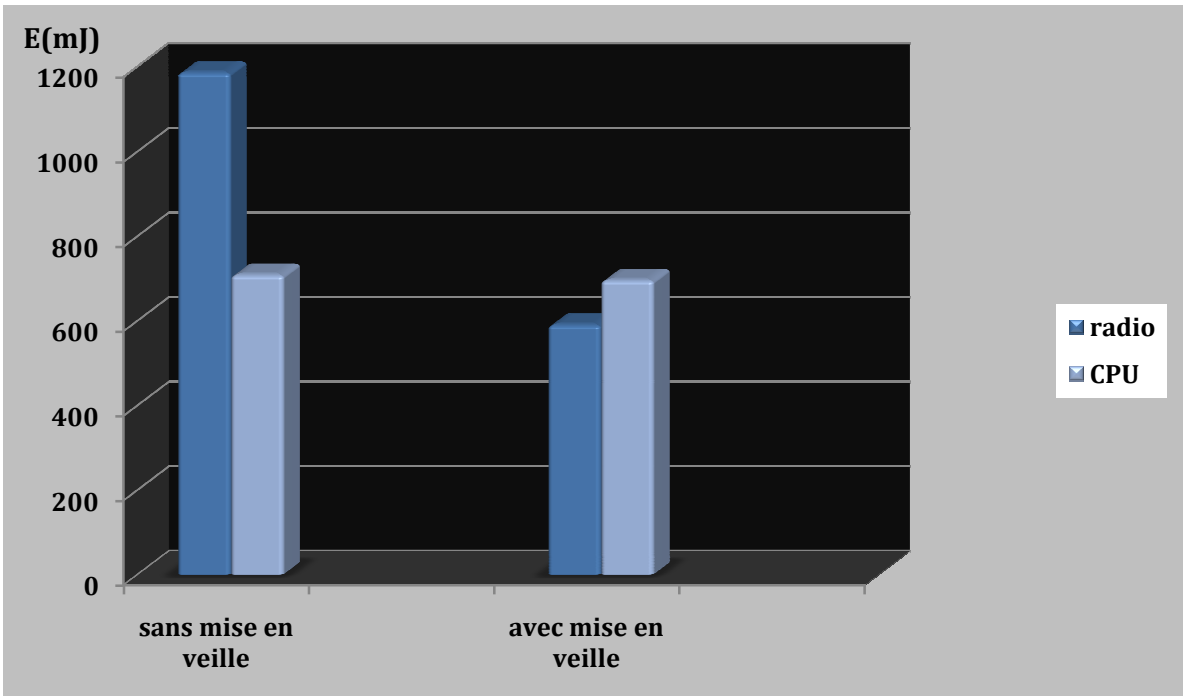


-figure VI-14 : comparaison de la consommation d'énergie d'une radio-

Dans ces résultats nous avons pour une durée de temps de simulation égal à 60s les résultats de consommation d'énergie du modèle radio :

- Sans mise en veille.
- Avec une mise en veille de la radio après envoi d'un paquet de données pour une durée du timer égale à *2000 ms.*
- Avec une mise en veille abusive pour une durée du timer égale à *10ms.*

VI 4-2.4 Consommation de la CPU et modèle radio (transcriver) dans les deux modes



-

Figure VI-15 : Comparaison de la consommation d'énergie avec/sans mise en veille –

Cette figure montre la diminution de la consommation d'énergie de la radio, tandi que la CPU consomme la même quantité, car la mise en veille dans notre cas ne concerne pas la CPU.

VI 4-2.5 Taux de gain d'énergie

Selon les résultats aboutis dans le cas de mise en veille idéale (dans cette simulation T=2000*ms*), nous avons constatés que dans l'application avec mise en veille gagne ***32.93%*** de l'énergie consommé par un nœud de l'application sans mise en veille.

VI 4.2.6 Comparaison des résultats avec les résultats de l'application avec compression de données (paquets transmis par la radio)

- **Gestion d'énergie d'un capteur avec la technique de compressions de données**

La technique de compression de données est l'une des solutions proposées dans la littérature pour la préservation d'énergie dans les réseaux de capteurs (technique invoqué dans le chapitre II).

Comme la mise en veille s'intéresse à l'état IDLE du module radio (l'élément le plus consommateur en énergie) ; la technique de compression de données s'intéresse à l'état

transmission du module radio, on minimisant la taille de données avant de les transmettre. A noter que la le taux de consommation d'énergie le plus élevé est lors de la transmission radio.

Le travail de [**91**] basé sur l'étude de cette technique donne le résultat de la consommation d'énergie en utilisant l'algorithme de compression de donné RLE et KRLE ; le résultat de comparaison entre les simulations de l'application de capture de temperature soumise à ces deux techniques sont mentionnés dans le **Figure VI-91** suivante :

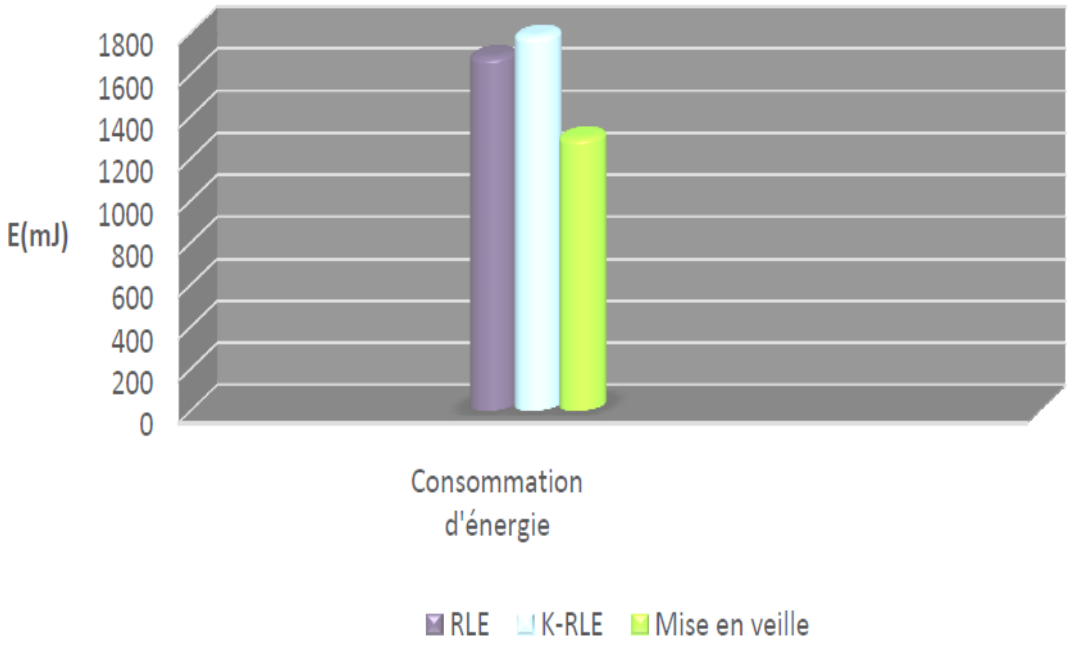


Figure VI-16 : Comparaison de la consommation d'énergie (compession de données/mise en veille)

Nous remarquons que la technique de mise en veille donne plus de bénéfice pour les raisons suivantes :

- Les algorithmes de compression de données consomment de l'énergie (énergie de traitement) plus l'algorithme est complexe plus la CPU consomme d'énergie.

- Contrairement au cas de mise en veille qui permet d'atteindre la radio, dans la technique de compression de données le gain d'énergie se fait par rapport à la taille de la donnée reçu, pas par rapport au temps de fonctionnement de la radio (tout le temps allumé).

VI- 5 Conclusion

La mise en veille du module radio est une technique très populaire adaptée dans diverses recherches pour la gestion d'énergie dans les réseaux de capteurs et les autres types de réseaux. Nous avons vu dans les chapitres précédents plusieurs protocoles et solutions basés sur cet impact.

Nous avons montré dans ce chapitre l'efficacité de cette technique et nous l'avons ensuite comparé avec une autre technique (la compression de données).

Cette technique est efficace en énergie ; cependant son utilisation doit se mettre sous les différentes contraintes :

- Le temps de mise en veille
- La latence de données
- Le choix du contexte : quand ce que l'application de cette technique peut être bénéfique.
  exemples :
  1. déconseillé dans une application à événements temps réel puisque cette technique provoque la latence (enchainement) et la perte de données (capter).
  2. Bénéfique dans les applications à réseaux denses. Dans ce cas on évite la sur-écoute, la redondance et on gagne dans la durée de vie du réseau.

Différents protocoles de tous les niveaux de couches bénéficient de cette technique de préservation d'énergie ; et les recherches se poursuivent en invoquant d'autres techniques à base de la mise en veille.

Conclusion

Les réseaux de capteur 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 calculs et de stockage limitées et surtout de leurs dépendances 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'omniprésent dans le monde de demain. Les recherches dans ce domaine se focalisent donc dans la majorité des cas sur la gestion énergétique.

Pour cela, nous avons mis en avant que la principale problématique à résoudre est la durée de vie des batteries embarquées sur les capteurs. Plusieurs techniques de consommation d'énergie ont été proposées. Dans le cadre de ce travail, nous avons choisi l'étude de la technique de mise en veille dans les réseaux de capteur, qui permet de mettre les nœuds capteurs en veille à des fins énergétique. En effet, d'énormes économies d'énergie soit envisageable si nous appliquons cette technique.

Dans l'étude que nous avons réalisé sur TOSSIM (le simulateur de TinyOS), nous avons pu mesurer l'efficacité de cette technique en terme de gain d'énergie ; cependant nous avons constaté que le choix de l'intervalle de mise en veille est l'une des contraintes qui peuvent assurées la performance du réseau et sa durée de vie.

Nous avons alors fait un choix pour la période de mise en veille, qui se résume ainsi :

- Pour une durée de mise en veille très langue, le gain en énergie est très important. Cependant, nous remarquons que la capacité d'envoi/réception du nœud diminue dans le réseau pendant la durée de simulation, c-à-d le nombre de paquets enchainé ou fournis par ce capteur diminue quand la durée de mise en veille augmente.
- Ce qui est le contraire dans le cas ou l'intervalle de temps de mise en veille est réduit. Sauf que qu'ici la consommation d'énergie d'un nœud capteur risque d'être plus importante qu'un nœud qui ne se met pas en veille, cela est dû au changement d'état du module radio (sleeping-> idle) lors du réveil qui nécessite une quantité d'énergie considérable.

Perspectives

La problématique d'énergie dans les réseaux de capteurs reste toujours un champ ouvert pour les recherches et questions scientifiques, divers études sont en cours de réalisations d'autres déjà réalisées, à chaque solution d'autres performances peuvent améliorer ces résultats.

Pour arriver au plus haut bénéfice dans l'étude de la minimisation de la consommation d'énergie nous proposons de faire une étude complète avec la mise en veille.

C'est-à-dire nous allons appliquer la mise en veille des trois modules consommateurs en énergie (MCU, sensor, radio) dans une seule application, en étudiant leurs périodes d'inactivité pour leurs permettre de sauvegarder l'énergie qui se consume dans l'Idle ou l'attente d'événement ou d'un paquet de données à traiter. Le résultat peut être bénéfique si les contraintes qui s'opposent ne seront pas un obstacle qui diminue la qualité du réseau.

Une autre perspective consiste à appliquer les deux techniques que nous avons comparées précédemment, dans ce cas le gain sera au niveau des états du module radio, qui gagne de la mise en veille l'énergie gaspillé dans « l'Idle », et dans l'état d'activité l'énergie dans « la transmission et réception » de données soumises à la technique de compression afin de réduire la taille des paquets ; d'où la minimisation de la consommation durant leurs enchainement dans le réseau.

Bien que la consommation des algorithmes de compression de données et le redémarrage de la radio, CPU et sensors après mise en veille peuvent être un obstacle sur ces perspectives ; nous pensons qu'une meilleures utilisation, et le choix du domaine et du matériel peuvent aider à apporter des atouts importants pour la problématique de l'énergie dans les réseaux de capteurs sans fil (WSN).

Bibliographie

[**1**] HUGO Duro ; *« Vers l'autonomie énergétique des réseaux de capteurs embarqués : conception et intégration d'un générateur piézoélectrique et d'un micro dispositif de stockage capacitif en technologie silicium »*, LAAS-CNRS, UPR 8001, 7 avenue du Colonel Roche, 31077 Toulouse cedex 4.

[**2**] ROUX Steve PLANCHE Sylvie ; T. E. R. *« Réseaux de capteurs sans fil & Courant Porteur en Ligne Master professionnel Systèmes Informatiques et Réseaux »*, Année 2008/2009.

[**3**] Rahim KACIMI, *« Techniques de conservation d'énergie pour les réseaux de capteurs sans fil, Ecole doctorale : Mathématiques Informatique et Télécommunications* », Toulouse, France Septembre, 2009.

[**4**] YACINE CHALLAL, « *Réseaux de Capteurs Sans Fils »* ; http://creativecommons.org/licenses/by-nc/2.0/fr/;2008.

[**5**] Mounira OULARBI et Soraya KASSAB , *« Elaboration d'un protocole de routage efficace en énergie pour réseaux de capteurs sans fil »* ; École nationale Supérieure d'Informatique (ESI) ; LABORATOIRE LMCS ; SEPTEMBRE 2010.

[**6**] Définition Réseau de capteurs sans-fil - Encyclopédie scientifique en ligne.htm.

[**7**] Ali CHAMAM ; *« Mécanismes optimisés de planification des états des capteurs pour la maximisation de la durée de vie dans les réseaux de capteurs sans fil* » ; Université Montréal ; AOÛT 2009.

[**8**] Samira Allam, « Approche multi agents pour contrôler l'inondation dans un réseau de capteurs » , 2008/2009.

[**9**] Fatima Mourchid, *« Nouveau modèle pour le positionnement des senseurs avec contraintes de localisation »*, Département de Génie Informatique et Génie Logiciel ; Ecole Polytechnique de Montréal ; Avril 2010.

[**10**] Wassim ZNAIDI, « *Modélisation formelle de réseaux de capteurs à partir de TinyOS* »; Ecole polytechnique de Tunisie; Juin 2006.

[**11**] Sofiane MOAD, *« Optimisation de la consommation d'énergie dans les réseaux de capteurs sans fil »* ; Université : IFSIC-Rennes 1 Laboratoire de recherche : DYONISOS-IRISA Année universitaire : 2007/2008.

[**12**] Abdellah Makhoul *« Réseaux de capteurs : localisation, couverture et fusion de données »* ; de Franche-Comté (LIFC) dans le cadre de l'École Doctorale Sciences Pour l'Ingénieur et Microtechniques (SPIM) ; novembre 2008.

[**13**] Werner-ALLEN (G.), Lorincz (K.), Welsh(M.) et al., *« Deploying a wireless sensor network on an active volcano »*, IEEE Internet Computing, April 2006.

[**14**] Gérard Chalhoub ; *« Réseaux de capteurs sans fil »* ; CLERMENT Université; décembre 2009.

[**15**] LEHSAINI Mohamed , *« Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique »* ; Université A.B Tlemcen Faculté des Sciences pour l'Ingénieur && Université de Franche-Comté U.F.R Sciences et Techniques École Doctorale SPIM ; 2009.

[**16**] MAINWARING, A., CULLER, D., POLASTRE, J., SZEWCZYK, R. et ANDERSON, J. *«Wireless sensor networks for habitat monitoring»*. Proc. 1st ACM international workshop on Wireless sensor networks and applications (WSNA), 2002.

[**17**] S. Macker, J. Corsen. «*Mobile Ad hoc Networking*» «*Routing Protocol Performance Issues and Evaluation Consideration*». RFC2501; January 1999.

[**18**] N.N et al. *« AILISA plateformes d'évaluation pour des technologies de télésurveillance médicale et d'assistance en gérontologie »;* Fond. Nationale de Gérontologie et société 2005/2 - n° 113, 2005.

[**19**] T.BELUCH, *« Architecture et modélisation de réseaux de capteurs sans fils synchronisés et localisés à faible consommation »* ; CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France.

[**20**] http ://www.argo.ucsd.edu/ (accessed 17 06 2008) .

[**21**] Gérard Chalhoub *; « MaCARI : Une méthode d'accès déterministe et économe en énergie pour les réseaux de capteurs sans fil »*; Université Blaise Pascal Ecole Doctorale Sciences pour l'ingénieur de CLERMONT-FERRAND ; 2009.

[**22**] Ted Herman, Sébastien Tixeuil, *« Un algorithme TDMA réparti pour les réseaux decapteurs »*, University of Iowa, Department of Computer Science LRI - CNRS UMR 8623, INRIA Projet Grand Large, Université Paris-Sud XI.

[**23**] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. *« Energy effcient communication protocol for wireless microsensor networks»*. In Proceedings of the Bibliographie 11933rd Hawaii International Conference on System Sciences (HICSS'00), volume 2, Washington, DC, USA, 2000. IEEE Computer Society.

[**24**] Kamal BEYDOUN *; « Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs »* Université de Franche-Comte ; décembre 2009.

[**25**] SAYAD Maya *; « Energy Efficient Protocol (EEP) : un protocole de routage efficace en énergie pour réseaux de capteurs sans fil»* ; Ecole nationale Supérieure d'Informatique (ESI) Oued-Smar, Alger ; 2008/2009.

[**26**] BALDUS (H.), KLABUNDE (K.) et MUESCH (G.), *« Reliable setup of medical body sensor networks »*, In the European conference on Wireless Sensor Networks (EWSN), Germany, 2004.

[**27**] GAO (T.), GREENSPAN (D.), WELSH (M.) et al., *" Vital signs monitoring and patient tracking over a wireless network"*, 2005.

[**29**] ANASTASI, G., CONTI, M., FRANCESCO, M. D. et PASSARELLA, A. «Energy conservation in wireless sensor networks : A survey. Elsevier Ad Hoc Networks», 2009.

[**30**] K. Holger and Andreas Willig. Wiley *«Protocols and Architectures for Wireless Sensor Networks»*., 2005.

[**31**] Carla-Fabiana Chiasserini, Imrich Chlamtac, Paolo Monti, and Antonio Nucci. *«Energy efficient design of wireless ad hoc networks.»*, London, UK, 2002. Springer-Verlag.

[**32**] Adrian VAN DEN BOSSCHE*; « Proposition d'une nouvelle méthode d'accès déterministe pour un réseau personnel sans fil à fortes contraintes temporelles »* ;LATTIS Laboratoire de Recherche LATTIS EA4155 I.U.T de Blagnac - 1, Place Georges Brassens – BP 60073 31 703 Blagnac cedex; Juillet 2007.

[**33**] Di Tianet and Nicolas D. Georganas. *«A coverage preserving node scheduling scheme for large wireless sensor networks»* New York, NY, USA, 2002. ACM.

[**34**] Wei Mo, Daji Qiao, and Zhengdao Wang. *"Mostly-sleeping wireless sensor networks : Connectivity, k-coverage, and alpha-lifetime"*. IL, USA, 2005.

[**35**] Mihaela Cardei My T. Thai, Yingshu Li, and Weili Wu. *"Energy-efficient target coverage in wireless sensor networks"*, Miami, USA, March 2005.

[**36**] Manish Bhardwaj and Anantha P. Chandrakasan. *"Bounding the lifetime of sensor network via optimal role assignments"*. New York, USA, June 2002.

[**37**] Honghai Zhang and Jennifer C. Hou. *"Maximizing lifetime for wireless sensor networks"*, 2005.

[**38**] Kui Wu, Yong Gao, Fulu Li, and Yang *Xiao "Lightweight deployment-aware scheduling for wireless sensor networks"*. Mobile Networks and Applications 2005.

[**39**] Bogdan C rbunar, Ananth Grama, Jan Vitek, and Octavian C rbunar. *"Redundancy and coverage detection in sensor networks"*. ACM Transactions on Sensor Networks, 2006.

[**40**] Wei Mo, Daji Qiao, and Zhengdao Wang. *"Mostly-sleeping wireless sensor networks : Connectivity, k-coverage, and alpha-lifetime"*. IL, USA, 2005

[**41**] Kewei Sha and Weisong Shi. *"Modeling the lifetime of wireless sensor networks"*. Sensor Letters, 2005.

[**42**] Santosh Kumar, Anish Arora, and Ten .H. Lai. *"On the lifetime analysis of always-on wireless sensor network applications"*. Washington, DC, USA, November 2005.

[43] ZIXIANG, X., LIVERIS, A. et CHENG, S. *"Distributed source coding for sensor networks"*. IEEE Signal Processing Magazine. 2004.

[44] SLEPIAN, D. et WOLF, J. *"Noiseless coding correlated information sources"*. IEEE Transactions on Information Theory. 1973.

[45] WYNER, A. et ZIV, J. *"The rate-distortion function for source coding with side information at the decoder"*. IEEE Transactions on Information Theory 1976.

[46] CHOU, J., PETROVIC, D. et RAMCHANDRAN, K. *"A distributed and adaptive signal processing approach to exploiting correlation in sensor networks"*. Elsevier Ad Hoc Networks (2004).

[47] PANTAZIS, N. et VERGADOS, D. *"A survey on power control issues in wireless sensor networks"*. IEEE Communications Surveys & Tutorials, 2007.

[48] RAJENDRAN, V., OBRACZKA, K. et GARCIA-LUNA-ACEVES, J. J. *"Energy-effcient collision-free medium access control for wireless sensor networks"*. 2003.

[49] MAINWARING, A., CULLER, D., POLASTRE, J., SZEWCZYK, R. et ANDERSON, J. *"Wireless sensor networks for habitat monitoring"*.2002.

[50] Alberto Cerpa and Deborah Estrin. Ascent : *Adaptive self-configuring sensor networks topologies. IEEE Transactions on Mobile Computing*, 2004.

[51] YE, W., HEIDEMANN, J. et ESTRIN, D. *"Medium access control with coordinated adaptive sleeping for wireless sensor networks"*. IEEE/ACM Transactions on Networking 2004.

[52] CI, S., SHARIF, H. et NULI, K. *"Study of an adaptive frame size predictor to enhance energy conservation in wireless sensor networks*". IEEE Journal on Selected Areas in Communications, 2005.

[53] HEINZELMAN, W., CHANDRAKASAN, A. et BALAKRISHNAN, H. *"Energy-efficient communication protocol for wireless microsensor networks"*.2000.

[54] INTANAGONWIWAT, C., GOVINDAN, R. et ESTRIN, D. *"Directed diffusion : A scalable and robust communication paradigm for sensor networks"*. 2000.

[55] SHAH, R. et RABAEY, J. *"Energy aware routing for low energy ad hoc sensor networks"*,2002.

[56] XU, Y., HEIDEMANN, J. et ESTRIN, D. "Geography-informed energy. Conservation for ad hoc routing", 2001.

[57] STANN, F. et HEIDEMANN, J. *"RMST : reliable data transport in sensor networks"*. IEEE International Workshop on Sensor Network Protocols and Applications, 2003.

[58] AKAN, O. B. et AKYILDIZ, I. F. *"Event-to-sink reliable transport in wireless sensor networks"*. IEEE/ACM Transactions on Networking, 2005.

[59] WAN, C.-Y., CAMPBELL, A. T. et KRISHNAMURTHY, L. *"PSFQ : a reliable transport protocol for wireless sensor networks"*. 2002.

[60] A. Gallais, J. Carle, D. *« Simplot-Ryl, and I. Stojmenovic. Localized sensor area coverage with low communication overhead. In Proc. of PerCom"*, Italy, 2006.

[61] XING, G., WANG, X., ZHANG, Y., LU, C., PLESS, R. et GILL, C. *"Integrated coverage and connectivity configuration for energy conservation in sensor networks"*. ACM Transactions on Sensor Networks 2005.
[62] LU, J., WANG, J. et SUDA, T. *"Scalable coverage maintenance for dense wireless sensor networks"*. 2006.

[63] Chamam A. et Pierre, S. *"Optimal Scheduling of Sensors' States to Maximize Network Lifetime in Wireless Sensor Networks"*. IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS) 2007.

[64] YAN, T., HE, T. et STANKOVIC, J. *"Differentiated surveillance for sensor networks"*. ACM, New York, NY, USA 2003.

[**65**] LINDSEY, S. et RAGHAVENDRA, C. *"PEGASIS : Power-efficient gathering in sensor information systems"*. 2002.

[66] CHATTERJEE, M., DAS, S. et TURGUT, D. *"WCA : A weighted clustering algorithm for mobile ad hoc networks"*. Journal of Cluster Computing (Special Issue on Mobile Ad hoc Networks) 2002.

[67] YOUNIS, O. et FAHMY, S. *"HEED : a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks"*. IEEE Transactions on Mobile Computing 2004.

[68] BANDYOPADHYAY, S. et COYLE, E. *"An energy efficient hierarchical clustering algorithm for wireless sensor networks.* 2003.

[69] ZHOU, W., CHEN, H.-M. et ZHANG, X.-F. *"An energy effcient strong head clustering algorithm for wireless sensor networks.* 2007.

[70] SUN, B., GAO, S.-X., CHI, R. et HUANG, F. *"Algorithms for balancing energy consumption in wireless sensor networks"*. ACM, New York, NY 2008.

[71] LIU, J.-S. et LIN, C.-H. R. *"Energy-efficiency clustering protocol in wireless sensor networks"*. Elsevier Ad Hoc Networks 2005.

[72] KUBISCH, M., KARL, H., WOLISZ, A., ZHONG, L. et RABAEY, J. *"Distributed algorithms for transmission power control in wireless sensor networks"*. 2003.

[73] PANICHPAPIBOON, S., FERRARI, G. et TONGUZ, O. *"Optimal transmit power in wireless sensor networks"*. IEEE Transactions on Mobile Computing 2006.

[74] RAGHUNATHAN, V., SCHURGERS, C., P., S. et SRIVASTAVA, M *"Energy-aware wireless microsensor networks"*. IEEE Signal Processing Magazine. 2002.

[75] TIAN, D. et GEORGANAS, N. *"A coverage-preserving node scheduling scheme for large wireless sensor networks"*. ACM, New York, NY, USA, 2002.

[76] H.WANG, S.-F., SU, Y.-Y., LIN, Y.-Y. et DOW, C.-R. *"A cluster-based coverage-preserved node scheduling scheme in wireless sensor networks"*, 2006.

[77] IMAD MAHGOUB,Florida, MOHAMMAD ILYAS, *"Sensor Network Protocols";* Florida Atlantic University Boca Raton, Florida, U.S.A. by Taylor & Francis Group, LLC CRC Press is an imprint of Taylor & Francis Group, 2006.

[78] Wei Ye, Member, IEEE, John Heidemann, Member, IEEE, and Deborah Estrin Senior Member, IEEE Medium Access Control with Coordinated, *"Adaptive Sleeping for Wireless Sensor Networks"*.

[79] Kazem Sohraby; Daniel Minoli;Taieb Znati; *"Wireless sensor networks Topology, Protocols, and Application"*s; WILEY-INTESCIENCE. AJOHN WILEY SONS,INC.,PUBLICATION Copyright(c) by John Wiley & Sons, Inc. All rights reserved 2007.

[80] Dragos NICULESCU, *« Topics In Ad-Hoc Networks: Communication Paradigms for Sensor Networks »*, NEC Laboratories America, IEEE Communications Magazine, Mars 2005.

[81] Ou Yang and Wendi Heinzelman *"A Better Choice for Sensor Sleeping"* Dept. of ECE,University of Rochester, Rochester, NY, 14620, USA oyang@ece.rochester.edu,wheinzel@ece.rochester.edu, 2009

[82] A. Warrier, J. Min, and I. Rhee. *"ZMAC : a Hybrid MAC for Wireless Sensor Networks"*. Technical report, Department of Computer Science, North Carolina State University, April 2005.

[83] A. Manjeshwar, D.P. Agrawal. *"TEEN: a routing protocol for enhanced efficiency in wireless sensor networks"*. Proceedings 15th International Parallel and Distributed Processing Symposium. 2001.

[84] rahul C.shah and jan M.Rabaey, *"Energy Aware Routing for Low Energy Ad Hoc Sensor Networks"*, Berkeley Wireless Research Center University of California, Berkley,2002.

[**85**] Zheng, R., Kravets, R.*: "On-demand Power Management for Ad Hoc Networks"*. In: 22nd Annual Joint Conference of the IEEE Computer and Communications Societies. 2003

[**86**] M. Dalmau - IUT de Bayonne : *Réseaux de capteurs*

[**87**] Alliance ZigBee, http://www.zigbee.org

[**88**] Zoghi, M.R., Kahaei, M.H *"Sensor Selection for Target Tracking in WSN Using Modified INS Algorithm"*. In: 3rd Internatioanl Conference on Information and Communication Technologies: From Theory to Applications.

[**89**] Yang and Wendi Heinzelman, "A Better Choice for Sensor Sleeping" Dept. of ECE, University of Rochester, Rochester, NY, 14620, USA, oyang@ece.rochester.edu,wheinzel@ece.rochester.edu

[90] D. BRAGINSKY and D. ESTRIN, « *Rumor Routing Algorithm for Sensor Networks* », 1st Workshop on Sensor Networks and Apps, Atlanta, GA, 2002.

[**91]** MAHIOUT Lilia *« les Impacts de la Compression de Données sur la Consommation d'énergie dans un WSN »*, Département Informatique ; Université UMMTO 2010/2011

[**92**]Gianni A. Di Caro *« Analysis of simulation environments for mobile ad hoc networks»*. Technical Report No. IDSIA-24-03. Dalle Molle Institute for Artificial Intelligence, Galleria 2, 6928 Manno, Switzerland. 2003-

[**93**]P. Erard, P. Déguénon, *"Simulation par événements discrets"*, presses polytechnique et universitaire Romandes, 1iere édition 1996.

[**94**]Ankit Singh, FH Frankfurt am Main ;*"TinyOS Manual For Beginners in Linux (Ubuntu)"* February 10, 2011

[**95**]Rev. A , *"TinyOS Getting Started Guide"*, October ©2002-2003 Crossbow Technology, Inc. All rights reserved

[**96**]EL Mehdi Damou « *Simulation d'un réseau de capteurs avec TinyOS »;* VERIMAG Centre Equation 2 avenue de Vignate 38610 Gières – France Tuteur : Laurent MOUNIER VERIMAG CTL 2 avenue de Vignate 38610 Gières – France.

[97]-TinyOS official website: http://www.tinyos.net/

[**98**]David Gay dgay, Robert von Behren , Matt Welsh mdw , Net Eric Brewer, David Culler ,Philip Levis ; *"The nesC Language: A Holistic Approach to Networked Embedded Systems"* culler@cs.berkeley.edu EECS Department Intel Research, Berkeley University of California, Berkeley 2150 Shattuck Ave, Suite 1300 Berkeley, CA 94720 Berkeley.

Annexes

Annexe A : le système d’exploitation TinyOS

Annexe B : le langage de programmation NesC

Annexe C : l’interface graphique TinyViz

Annexe D : le simulateur PowerTOSSIM

Annexe A : Le système d'exploitation TinyOS

A.1. Présentation générale de TinyOS

TinyOS est un système d'exploitation open source conçu pour les réseaux de capteurs par l'université américaine de BERKELEY. Le caractère open source permet à ce système d'être régulièrement enrichi par une multitude d'utilisateurs. Sa conception a été entièrement réalisée en NesC, langage orienté composant syntaxiquement proche du C. Il respecte une architecture basée sur une association de composants, réduisant ainsi la taille du code nécessaire à sa mise en place. Cela s'inscrit dans le respect des contraintes de mémoires qu'observent les capteurs pourvus de ressources très limitées dues à leur miniaturisation.

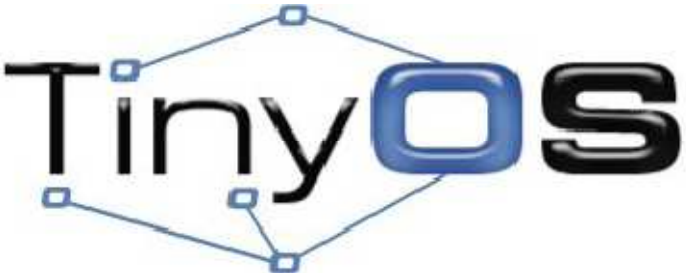


Fig. A-1 : Cigle du système d'exploitation TinyOS.

Pour autant, la bibliothèque de composants de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. Un programme s'exécutant sur TinyOS est constitué d'une sélection de composants systèmes et de composants développés spécifiquement pour l'application à laquelle il sera destiné (mesure de température, taux d'humidité…).

TinyOS s'appuie sur un fonctionnement évènementiel, c'est-à-dire qu'il ne devient actif qu'à l'apparition de certains évènements. Le reste du temps, le capteur se trouve en état de veille, vu les faibles ressources énergétiques des capteurs, garantissant ainsi une durée de vie maximale. Ce type de fonctionnement permet une meilleure adaptation à la nature aléatoire de la communication sans fil entre capteurs.

A.2. Caractéristiques de TinyOS

TinyOS a été créé pour répondre aux caractéristiques et aux nécessités des RCSF telles que :

- **Taille réduite** : TinyOS a une empreinte mémoire très faible puisqu'il ne prend que 4 Ko de mémoire libre et 300 à 400 octets dans le cadre d'une distribution minimale.

- **Applications orientées composants**: Un programme s'exécutant sur TinyOS est constitué d'une sélection de composants qui peut être utilisée telle quelle ou bien adaptée à une application précise (mesure de température, du taux d'humidité, etc.). A cette fin, TinyOS fournit une réserve de composants systèmes utilisables au besoin. Parmi les plus fréquents, on cite ceux concernant les entrée/sorties, les timers, etc.

TinyOS utilise un Langage de Description d'Architecture ou ADL6 afin de définir quels sont les composants impliqués dans la création de l'application ainsi que la manière dont ils sont reliés. Cette liaison entre composants repose sur la notion d'interface.

- **Programmation orienté évènement**: Le plus gros avantage de TinyOS est qu'il est basé su un fonctionnement événementiel, c'est à dire qu'il ne devient actif qu'à l'apparition de certains évènements. Le reste du temps, le capteur se trouve en état de veille afin de garantissent une durée de vie maximale aux faibles ressources énergétiques du capteur. Ce fonctionnement évènementiel (event-driven) s'oppose au fonctionnement dit temporel (time-driven) où les actions du système sont gérées par une horloge donnée.

- **Non 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. Autrement dit, une tâche ne peut pas interrompre une autre tâche. Ce mode de fonctionnement permet de bannir les opérations pouvant bloquer le système et donne la priorité aux interruptions matérielles (i.e. les événements peuvent interrompre les tâches). TinyOS est donc basé sur une structure à deux niveaux de planification :
  - Les évènements : ils sont utilisés pour réaliser des processus urgents et courts.
  - Les tâches : les tâches sont pensées pour réaliser une plus grande quantité de traitements et elles ne sont pas critiques dans le temps. Les tâches sont exécutées complètement, mais l'initialisation et la terminaison d'une tâche sont des fonctions séparées. Les tâches ne peuvent pas prendre de paramètre en entrée.

- **Pas de temps réel** : Lorsqu'un système est dit « temps réel » celui ci gère des niveaux de priorité dans ses tâches permettant de respecter des échéances données par son environnement. Dans le cas d'un système strict, aucune échéance ne tolère de dépassement contrairement à un système temps réel mou. TinyOS se situe au delà de ce second type car il n'est pas prévu pour avoir un fonctionnement temps réel.

A.3. Equipements supportés par TinyOS

TinyOS peut être implémenté sur un PC capteur (ATMega8, AVRMote, Mica, Rene2, MSP430, Telos). Au delà de cette liste, il est possible d'implémenter tout type de plateforme embarquée physique en redéveloppant les bibliothèques nécessaires à la prise en compte des entrées sorties nécessaires. Citant comme exemple le résultat d'une thèse mettant en œuvre TinyOS sur un dispositif Freescale MC13192-EVB (semi-conducteur utilisé pour évaluer des plateformes) sur un réseau ZigBee**.**

A.4. Allocation de la mémoire

Il est très important d'aborder la façon avec laquelle un système d'exploitation gère la mémoire, d'autant plus lorsque ce système travaille dans un environnement aussi restreint. TinyOS occupe un espace mémoire faible répartie en :

- Pile : sert de mémoire temporaire au fonctionnement du système notamment pour l'empilement et le dépilement des variables locales.
- Variables globales : réservent un espace mémoire pour le stockage de valeurs pouvant être accessible depuis des applications différentes.

- Mémoire libre : pour le reste du stockage temporaire.

TinyOS possède une mémoire fixe. En effet, il interdit les allocations dynamiques ainsi que celles se produisant à l'exécution. De plus, les pointeurs de fonctions n'existent pas. Pour cela, TinyOS s'appuie sur le graphe de composants précédemment décrit afin de déterminer la taille de chaque composant et ainsi établir statiquement leurs liaisons à la compilation. Par ailleurs, il n'existe pas de mécanisme de protection de la mémoire sous TinyOS, ce qui rend le système particulièrement vulnérable aux corruptions de la mémoire.

A.5. Allocation de ressources

Le choix d'un ordonnanceur détermine le fonctionnement global du système et le dotera de propriétés précises telles que la capacité à fonctionner en évènementiel. L'ordonnanceur TinyOS se compose de :

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

A l'appel d'une tâche, celle-ci va prendre place dans la FIFO en fonction de sa priorité (plus elle est grande, plus le placement est proche de la sortie). Dans le cas où la file d'attente est pleine, la tâche dont la priorité est la plus faible est enlevée de la file FIFO. Lorsque la file est vide, le système met en veille le dispositif jusqu'au lancement de la prochaine interruption.

A.6. Guide d'installation :

Deux principales versions de TinyOS sont disponibles : la version stable (1.1.0) et la version en développement (2.0.2) qui nécessite l'installation de l'ancienne version pour fonctionner. TinyOS peut être installé sur Windows (2000 et XP), GNU/Linux, Mac OS ou sur un capteur. Nous avons procédé à l'installation de la première version de TinyOS sur Windows Vista et ubuntu 10.10.

Annexe B : Le langage de programmation NesC

B.1. Présentation générale de NesC

NesC est une extension du langage de programmation C. Il est conçu pour incarner les concepts structurant et le modèle d'exécution de TinyOS. Les composants sont les éléments de base pour former une application NesC. Chaque composant correspond à un élément matériel (LED, timer, ADC …) et peut être réutilisé dans différentes applications.

Les composants NesC fournissent ou utilisent des interfaces bidirectionnelles qui définissent d'une manière abstraite les interactions entre deux composants. L'utilisation des mots clés **use** et **provide** au début d'un composant permet de savoir respectivement si celui-ci fait appel à une fonction de l'interface ou redéfini son code. Il est à noter que tous les composants NesC doivent posséder l'interface StdControl car celle-ci est utilisée pour initialiser, démarrer et arrêter les composants.

Les composants NesC présentent des similarités avec des objets. Les états sont encapsulés et on peut y accéder par des interfaces. En NesC, l'ensemble des composants et leurs interactions sont fixés à la compilation pour plus d'efficacité. Ce type de compilation permet d'optimiser l'application pour une exécution plus performante. En langage objet, cette phase est réalisée lors de l'exécution ce qui rend celle ci plus lente.

B.2. Implémentation d'une application NesC

Pour implémenter une application NesC, il faut avoir connaissance sur la structure et le fonctionnement des composants et des interfaces qui la constituent. Cette partie permet de bien expliquer ces notions. Il est néanmoins recommandé de faire recours aux leçons au niveau du tutorial TinyOS qui englobe tous les besoins de programmation NesC en accédant à **/opt/tinyos-1.x/doc/tutorial**

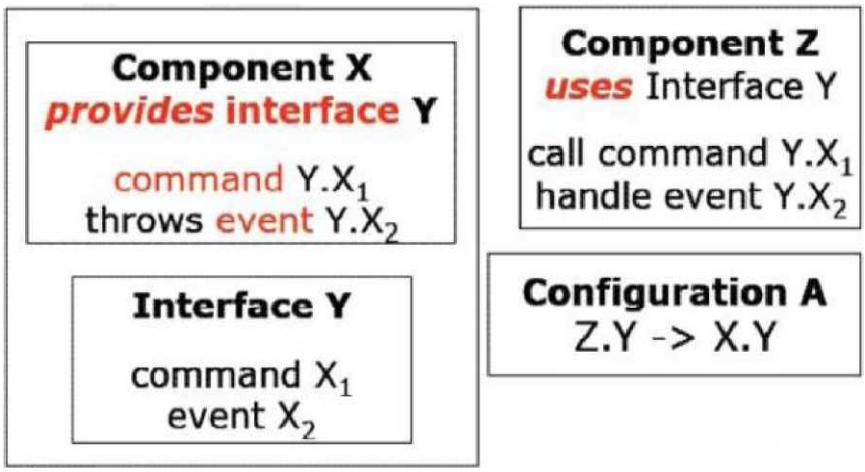


Fig. B-1 : Architecture générale d'une application NesC

B.2.1. Les interfaces

Une interface déclare deux types de fonctions: commandes et événements. Afin de distinguer ces fonctions, leurs en-têtes sont précédés des mots-clés respectifs ***event*** ou ***command***.

Les commandes font typiquement des appels du haut vers le bas (des composants applicatifs vers les composants plus proches du matériel). Tandis que les événements remontent les signaux du bas vers le haut.

Pour appeler une commande, il faut utiliser le mot-clé **call**. Par exemple:

call Send.send (1, sizeof(Msg), &msg1) ;

Pour signaler un événement, il faut utiliser le mot-clé **signal**. Par exemple:

signal Send.sendDone(&msg1, SUCCESS);

Le modèle mémoire fixé par TinyOS n'autorise pas les pointeurs de fonctions. Afin de proposer un mécanisme alternatif, NesC utilise des interfaces paramétrées. Celles-ci permettent à l'utilisateur de créer un ensemble d'interfaces identiques et d'en sélectionner une seule à appeler grâce à un identifiant. Par exemple : interface **SendMsg [uint8_t id]**

B.2.2. Les composants

Il existe deux types de composants : les configurations et les modules.

B.2.2.1. Les configurations

Elles permettent de décrire les composants composites, i.e., des composants composés d'autres composants. Elles relient les interfaces utilisées par certains composants aux interfaces offertes par d'autres composants. Une configuration est donc constituée de modules et/ou d'interfaces ainsi que de la description des liaisons entre ces composants. Il existe trois possibilités de connexion:

- End-point1 = End-point2
- End-point1 -> End-point2
- End-point1 <- End-point2 (équivalent à : endpoint2 -> endpoint1)

Les éléments connectés doivent être compatibles : Interface à interface, event à event, etc. Il faut toujours connecter un utilisateur d'une interface à un fournisseur de l'interface.

Il est à noter que la configuration Main est obligatoirement présente dans la configuration décrivant l'ensemble de l'application car son rôle est de démarrer l'exécution de l'application.

B.2.2.2. Les modules

Ce sont les éléments de base de la programmation. Ils permettent de fournir les codes des applications NesC. Pa ailleurs, il est à noter que le modèle d'exécution proposé par NesC repose sur les tâches et les gestionnaires d'interruption. Donc, les modules permettent aussi d'implémenter ces tâches.

Une tâche est un ordonnancement FIFO utilisée pour réaliser un travail qui nécessite beaucoup de calculs. Elle peut être postée par une commande ou un événement. C'est un élément de contrôle indépendant défini par une fonction retournant **void** et sans arguments :

task void NomTask() { ... }

Les tâches sont lancées en les préfixant par post:

post NomTask();

B.3. Compilation d'une application NesC

Les fichiers de NesC portent l'extension *.nc*. Par ailleurs, le compilateur de NesC est appelé ncc. Pour effectuer la compilation, les fichiers sources doivent se situer dans le même répertoire contenant aussi un makefile de la forme :

COMPONENT= nom de l'application
include ../Makerules

Ce Makefile permet de compiler le composant en spécifiant en paramètre la plateforme sur laquelle doit fonctionner l'application. Par exemple, pour un capteur de type mica2, la commande permettant de compiler l'application sera : make mica2. Le compilateur ncc offre aussi la possibilité de pouvoir compiler l'application pour l'utiliser sur un simulateur de TinyOS. Dans ce cas, la commande sera : **make pc**. Cette commande génère un exécutable **main.exe** dans l'arborescence **/repertoire_courant/build/pc**.

B.4. Exemple illustratif d'une application NesC

On va donner l'exemple universel « Bonjour » ou « Hello » pour mieux illustrer la structure d'une application NesC.

Hello.nc

```
configuration Hello {
}

implementation {
components Main, HelloM,
SingleTimer, LedsC ;
Main.StdControl --->
HelloM.StdControl;
Main.StdControl – –> SingleTimer;
HelloM.Timer --->
SingleTimer.Timer ;
HelloM.Leds ---> LedsC ;

}
```

HelloM.nc (3)

```
module HelloM {
provides {
interface StdControl;
}
uses {
interface Timer; interface
Leds;
}
}
```

HelloM.nc (2)

```
implementation {
command result_t StdControl.init(){
call Leds.init();
return SUCCESS;
} }
command result_t
StdControl.start(){
return call
Timer.start(TIMER_ONE_SHOT,
1000);
}
command result_t
StdControl.stop(){
return call Timer.stop();
```

HelloM.nc (3)

```
event event Timer.fired(){
call Leds.redOn();
call Leds.greenOn();
call Leds.yellowOn();
return SUCCESS;
}
} // implementation
```

Fig. B-2 : Exemple illustratif d'une application NesC.

Annexe C : L'interface graphique TinyViz

C.1. Présentation générale de TinyViz

TinyViz est fourni avec TinyOS. Il s'agit d'une interface graphique programmée en langage JAVA. Elle permet de représenter un RCSF émulé grâce au simulateur TOSSIM.

Pour plus d'informations sur l'utilisation de TinyViz, aller à:
http://www.tinyos.net/tinyos-1.x/doc/tutorial/lesson5.html

Une fois TinyViz est lancé, on peut visualiser une fenêtre comme celle illustrée dans la figureC.1. Dans la partie gauche de cette figure, on distingue les capteurs qui sont déplaçables dans l'espace. Quant à la partie droite, on distingue les commandes permettant d'intervenir sur la simulation:

- **On/Off**: met en marche ou éteint un capteur.
- **Delay**: permet de sélectionner la durée au bout de laquelle se déclenche le timer.
- **Play**: permet de lancer la simulation où de la mettre en pause
- **Bouton de grilles**: affiche un quadrillage sur la zone des capteurs afin de pouvoir les situer dans l'espace.
- **Clear**: efface tous les messages qui avaient été affichés lors de la simulation.
- **Stop**: arrête la simulation et ferme la fenêtre.

Pour lancer une application, il faut régler le **Delay** souhaité entre chaque application, choisir les plugins de visualisation que l'on souhaite, et, appuyer sur **Play**. La simulation démarre.

Chaque onglet contient un plugin qui permet de visualiser la simulation de façon plus ou moins détaillée. Par exemple, en activant le plugin **Debug Messages**, tous les messages de type **Debug** apparaîtront dans l'onglet correspondant. Le plugin **Radio Links** permet de visualiser graphiquement par des flèches, les échanges effectués entre les capteurs. Plus précisément, si un capteur envoie un broadcast, il sera repéré par un cercle. Par contre, s'il envoie un message direct (unicast) alors le lien de communication sera repéré par une flèche.

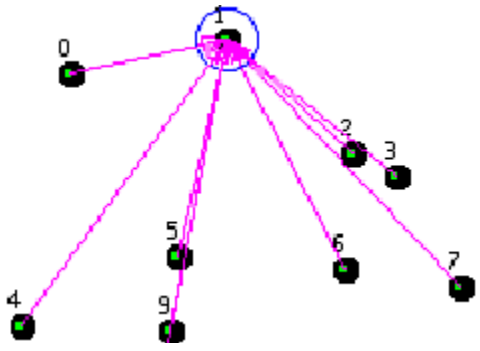


Fig. C-2 : Echange de messages entre les nœuds.

Annexe D: PowerTOSSIM

D.1. Présentation générale

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 VI-5), 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.

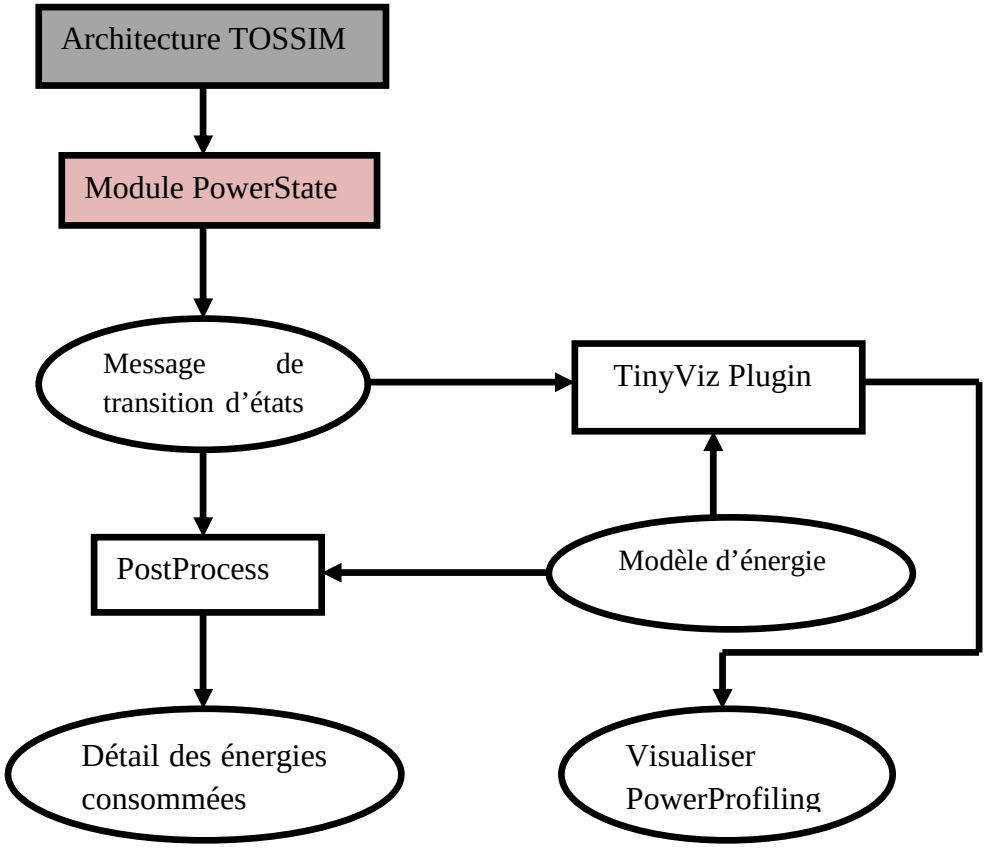


Fig. D-1: Architecture de PowerTOSSIM.

D.2. Lancer PowerTOSSIM

A- 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 d 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=60 -p 10 > NomApp.trace** (Le temps est égal à 60 secondes et le nombre de noeuds 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 paramètre *–sb* spécifie si les noeuds sont attachés à un autre nœud (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 :

Mote 0, cpu total: 719.503906
Mote 0, radio total: 1235.255862
Mote 0, adc total: 0.000000
Mote 0, leds total: 571.570576
Mote 0, sensor total: 0.000000
Mote 0, eeprom total: 0.000000
Mote 0, cpu_cycle total: 0.000000
Mote 0, Total energy: 2526.330344
.
.
Mote 9, cpu total: 635.394462
Mote 9, radio total: 1090.990102
Mote 9, adc total: 0.000000
Mote 9, leds total: 504.416514
Mote 9, sensor total: 0.000000
Mote 9, eeprom total: 0.000000
Mote 9, cpu_cycle total: 0.000000
Mote 9, Total energy: 2230.801078

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.

B- Pour récupérer l'état de l'horloge lors de la transmission et de la réception de paquets, il faut passer par ces étapes:

1- Accéder à l'application à simuler et la compiler en tapant: **make pc**
2- Taper **export DBG=clock**

Pour afficher des messages en parallèle avec l'horloge, taper **export DBG=clock, usr1**

3- Exécuter **main.exe** en tapant **: /build/pc/main.exe -t=60 -p 10 > NomApp.trace**

4- Accéder au fichier **NomApp.trace**
Il contient des lignes sous la forme suivante :

Moment d'envoi du paquet
Heure : Minute : Seconde

2: CLOCK: event handled for mote 2 at **0:0:36.47777400** (347634 ticks).
2: CLOCK: Setting clock interval to 218 @ 0:0:36.47777400
2: j'envoie le paquet de données à la destination 42 ///DBG usr1
.
.
42: j'ai reçu le paquet de données de la source 2 ///DBG usr1
.
.
.
.
.

Moment de réception du paquet. Délai de propagation du paquet=36, 60979650-36, 47777400=0, 13202250 secondes

42: CLOCK: event handled for mote 42 at **0:0:36.60979650** (902286 ticks).
42: CLOCK: Setting clock interval to 231 @ 0:0:36.60979650