République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieure et de la Recherche Scientifique

Université MOULOUD MAMMERI Tizi-Ouzou
Faculté du Génie Electrique et Informatique
Département Informatique

Mémoire de fin d'études

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

Spécialité: Systèmes Informatiques

Théme

Etude et réalisation d'un algorithme de routage plat

Cas : Sensor Protocol for Information via Negotiation (SPIN)

Réalisé par:

M^elle^: BOUHATEM Fariza

M^elle^: DJOUDER Lydia

proposé et dirigé par:

M^r^: DEMRI Mohamed

Promotion 2014/2015

Remerciements

Notre profonde gratitude et sincères remerciements vont à notre promoteur M^er DEMRI pour nous avoir confié ce travail, pour leur suivi, leur disponibilité, leurs orientations et leurs remarques pertinentes et précieuses.

Nous remercions chaleureusement les membres du jury pour l'honneur qu'ils nous font en acceptant de juger ce mémoire de fin d'études.

Enfin, nous remercions toutes les personnes ayant contribué de prés ou de loin au bon accomplissement de notre travail.

Fariza & Lydia

Dédicace

Dieu merci, Dieu merci Dieu merci… !

Je dédie ce modeste travail

- ✓ A ma famille qui m'ont soutenue tout au long de mes études pour réaliser ce mémoire et dont le rêve était toujours de nous voir réussir.
- ✓ A mes chères sœurs, ma mère.
- ✓ A mes neveux Maya, Tina, Walid, Ilyas, Malek, Hocine, Hewa.

Fariza

Dédicace

Dieu merci, Dieu merci Dieu merci… !

JE DÉDIE CE MODESTE TRAVAIL

A mes chers parents que Dieu les bénissent, qui ont su me guider par leurs précieux conseils, leurs encouragements continuels et que tout simplement m'ont appris et continues de m'apprendre que l'école de la vie n'à point de vacance et que le plus puissant de tous les leviers c'est la volonté.

A mon frère Sofiane et son épouse Aziza.

A mon frère Lyes et son épouse Fatima.

A mes freres Salah et Rafik.

A ma sœur Naima et son mari Mouhamed.

A ma sœur Nabila et son mari Arezki.

A mes nièces : Thiziri, Maylisse et Kenza

A mes neveux : Kouseila, Rayane et Yani Kyliane.

A ma binôme Wassila ainsi que toute sa famille.

A tous ceux qui m'aiment.

Lydia

La liste des acronymes

ADC:Analog **D**igital **C**onverter

ADV:ADVertisement message

API:Application **P**rogramming **I**nterface

CH:Cluster **H**ead

DD:Directed **D**iffusion

GPS : Global **P**osition **S**ystem

LEACH: Low-**E**nergy **A**daptive **C**lustering **H**ierarchy

NesC:Network **E**mbedded **S**ystem **C**

OSI: Open **S**ystems **I**nterconnect

QoS: Quali**t**y **o**f **S**ervice

RCSF : **R**éseaux de **C**apteurs **S**ans **F**ils.

REQ:join**REQ**uest message

RR:Rumour **R**outing

SPIN: Sensor **P**rotocol for **I**nformation via **N**egotiation.

SPIN-BC: Sensor **P**rotocol for **I**nformation via **N**egotiation-**B**roadcast **C**hanel

SPIN-EC: Sensor **P**rotocol for **I**nformation via **N**egotiation-**E**nergy **C**onservation

SPIN-PP: Sensor **P**rotocol for **I**nformation via **N**egotiation-**P**oint to **P**oint

SPIN-RL: Sensor **P**rotocol for **I**nformation via **N**egotiation-**R**e**L**iable version of SPIN-BC

SQL: Structured **Q**uery **L**anguage

TEEN: Threshold sensitive **E**nergy **E**fficient **N**etwork

TinyOS:**T**iny **O**pen **S**ource

TOSSIM:Tiny**OS SIM**ulator

WSN: Wireless **S**ensor **N**etwork

Table des figures

Table des figures

La liste des tableaux

Table de matiére

Table des matières

Table des matières

Chapitre II : Le protocole de routage SPIN

Chapitre III : Implémentation et Simulation

Table des matières

Introduction Générale

Introduction Générale

Les progrès réalisés ces dernières décennies dans les domaines de la microélectronique, de la micromécanique, et des technologies de communication sans fil, ont permis de produire avec un coût raisonnable des composants de quelques millimètres cubes de volume. Ces derniers, appelés micro-capteurs, intègrent : une unité de captage chargée de capter des grandeurs physiques (chaleur, humidité, vibrations) et de les transformer en grandeurs numériques, une unité de traitement informatique et de stockage de données et un module de transmission sans fil.

De ce fait, les micro-capteurs sont de véritables systèmes embarqués. Le déploiement de plusieurs d'entre eux, en vue de collecter et transmettre des données environnementales vers un ou plusieurs points de collecte, d'une autonome, forme un réseau de capteurs sans fil.

La taille réduite des micro-capteurs, la large gamme de type de capteurs disponibles, ainsi que le fonctionnement autonome des réseaux de capteurs, leur confère un brillant avenir dans plusieurs domaines d'application tels que le domaine environnemental, le domaine militaire ou le domaine médical. Dans un futur proche, ils pourront être présents dans nos vies quotidiennes tout comme les micro-ordinateurs ou les téléphones portables le sont aujourd'hui.

Cependant, le développement de tels réseaux est confronté à de nombreux problèmes liés aux caractéristiques des applications (exigences en nombre de nœuds et en densité de déploiement, tolérance aux pannes, etc.) et aux limitations des capacités physiques des micro-capteurs. La consommation d'énergie est d'une importance primordiale. Chaque micro-capteur est alimenté par une source d'énergie limitée et généralement irremplaçable.

De ce fait, sa durée de vie (ou son autonomie) et, en conséquence, celle de tout le réseau est limitée. Dès lors, une attention particulière à la consommation d'énergie est requise lors de la conception des réseaux de capteurs ainsi que des protocoles de communications qu'ils utilisent. Par exemple, les protocoles de routages doivent garantir l'acheminement des données captées vers les points de collecte, tout en réduisant la consommation d'énergie induite par les transmissions.

Parmi les propositions éminentes traitant le problème de routage dans les réseaux de capteurs, on distingue le protocole SPIN. SPIN (Sensor Protocol for Information via Negotiation) repose sur le modèle de négociation afin de propager l'information dans un réseau de capteur sans fils.

Introduction Générale

Notre travail vise à étudier le fonctionnement du protocole SPIN et le réaliser en utilisant le langage de programmation nesC. Ainsi il est organisé en trois chapitres ; Le premier est une introduction au routage dans les réseaux de capteurs sans fil. Nous y présentons les critères de performance des protocoles de routage en RCSF ainsi qu'une classification des différents protocoles de routage proposés dans la littérature. Le second est consacré à l'étude du protocole SPIN ainsi un aperçu sur LEACH. Le dernier chapitre décrit la réalisation de l'algorithme de routage plat SPIN et la discussion des résultats obtenus on utilisant les outils de simulation de TinyOS.

Chapitre I:
Le routage dans les réseaux de capteurs sans fils

Chapitre I : Le routage dans les réseaux de capteurs sans fil

I.1Introduction

Dans la vie courante, l'utilisation des capteurs sans fil est en demande croissante pour la supervision et la sécurité. Les industries proposent alors des capteurs sans fil qui peuvent renseigner l'utilisateur sur plusieurs données. [1]

L'établissement correct et efficace d'itinéraires entre une paire de nœuds afin que des messages puissent être acheminés est l'objectif principal des protocoles de routage. Ces protocoles permettent aux nœuds de communiquer pour relayer les messages par des sauts multiples et de transmettre les données vers un point de collecte.

Avant d'entrer dans l'étude des protocoles de réseau de capteurs sans fil, nous commençons par présenter un capteur sans fil, son architecture et ses applications. Ensuite nous expliquons les différentes contraintes dans un réseau de capteur et particulièrement la consommation d'énergie et les différentes sources causant cette consommation et les techniques pour minimiser cette perte.

I.2.Définition d'un capteur sans fil

Un capteur sans fil est un petit dispositif électronique capable de mesurer une valeur physique environnementale (température, lumière, pression, etc.) et de la communiquer à un centre de contrôle via une station de base. Les progrès conjoints de la microélectronique, des technologies de transmission sans fil et des applications logicielles ont permis de produire à coût raisonnable des micro-capteurs de quelques millimètres cubes de volume, susceptibles de fonctionner en réseaux [2].

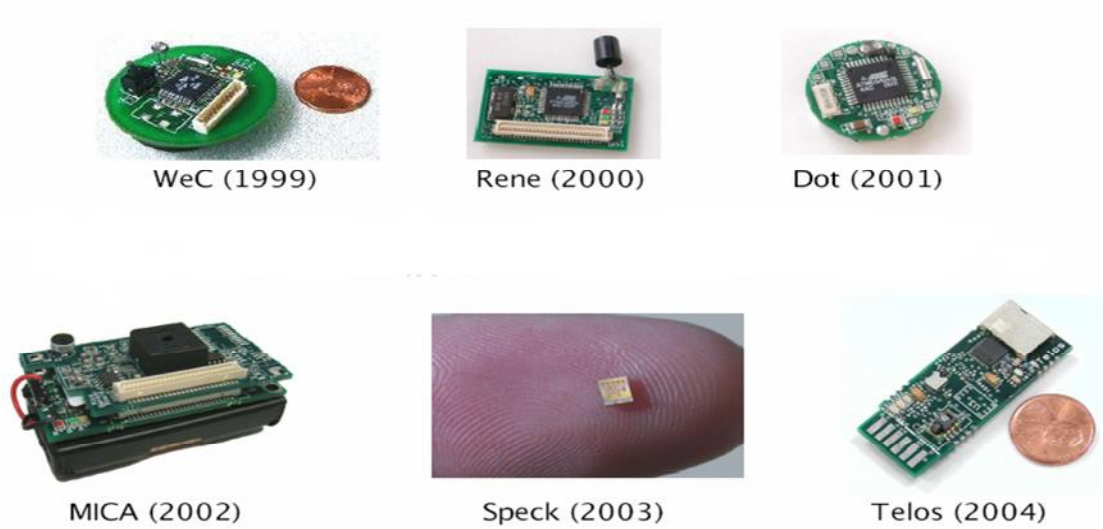


Figure I.1 Exemple de capteur. [41]

Un capteur est composé de quatre unités de base (figure I.2).

1. ***L'unité d'acquisition*** : elle est généralement composée de deux sous-unités qui sont le capteur et le convertisseur analogique-numérique ADC (Analog-Digital Converter). Les capteurs obtiennent des mesures sur les paramètres environnementaux et les transforment en signaux analogiques. Les ADCs convertissent ces signaux analogiques en signaux numériques.

2. ***L'unité de traitement*** *:* composée d'un processeur et d'une mémoire intégrant un système d'exploitation spécifique (TinyOS, par exemple). Cette unité possède deux interfaces, une interface pour l'unité d'acquisition et une interface pour l'unité de communication. Elle acquiert les informations en provenance de l'unité d'acquisition et les transmet à l'unité de communication. Cette unité est chargée aussi d'exécuter les protocoles de communications qui permettent de faire collaborer le capteur avec d'autres capteurs. Elle peut aussi analyser les données captées.

3. ***Un module de communication (Transceiver)*** *:* il est composé d'un émetteur/récepteur permettant la communication entre les différents nœuds du réseau via un support de communication radio.

4. ***Batterie (unité d'alimentation)****:* elle alimente les unités que nous avons citées et elle n'est généralement ni rechargeable ni remplaçable. La capacité d'énergie limitée au niveau des capteurs représente la contrainte principale lors de conception de protocoles pour les réseaux de capteurs. - Il existe des capteurs qui sont dotés d'autres composants additionnels tels que les systèmes de localisation GPS (Global Position System),…, etc.

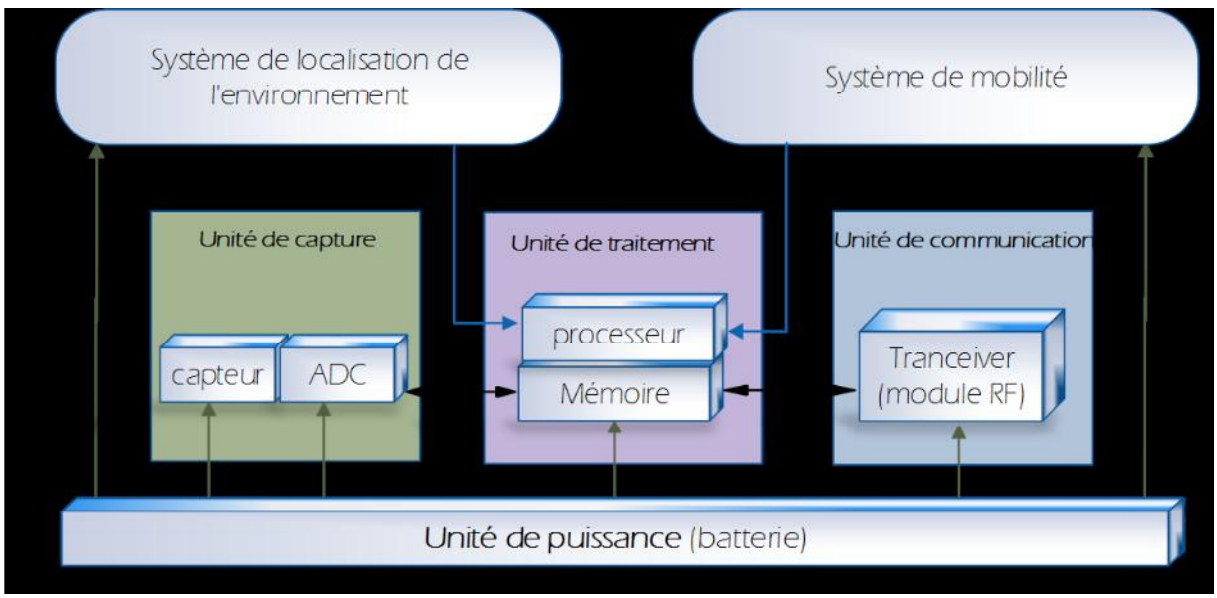


Figure I. 2 : Architecture physique d'un capteur [33]

Système de localisation de l'environnement : les tâches de détection et les techniques de routage ont besoin de connaitre souvent la localisation géographique d'un nœud. Ainsi, un nœud peut être équipé d'un système de localisation géographique. Ce système peut se composer d'un module de GPS pour un nœud de haut niveau ou bien d'un module de software qui implémente des algorithmes de localisation qui fournissent les informations sur l'emplacement du nœud par des calculs distribués.

Système de mobilité : la mobilité est parfois nécessaire pour permettre à un nœud de se déplacer pour accomplir ses tâches. Le support de mobilité exige des ressources énergétiques étendues qui devraient être fourni efficacement. Le système de mobilité peut, également, opérer dans l'interaction étroite avec l'unité de détection et le processeur pour contrôler les mouvements du nœud.

I.3. Architecture d'un RCSF

Tous les capteurs respectent globalement la même architecture basée sur un noyau central autour duquel s'articulent les différentes interfaces d'entrée-sortie, de communication et d'alimentation [10, 11]. La figure I.3 montre un exemple d'un réseau de capteurs.

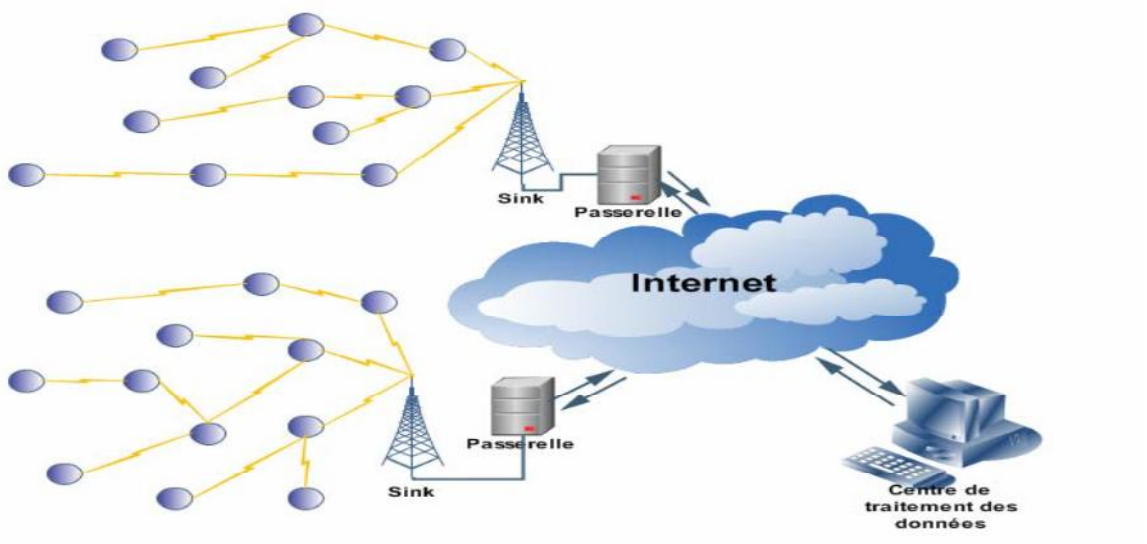


Figure I.3 : Exemple de réseaux de capteurs. [33]

Un RCSF est composé d'un ensemble de nœuds capteurs qui sont organisés en champs «Sensor Fields». Chacun de ces nœuds a la capacité de collecter des données et de les transférer au nœud passerelle par l'intermédiaire d'une architecture multi-sauts. Le nœud passerelle transmet ensuite ces données par Internet ou par satellite à l'ordinateur central «Gestionnaire de tâches» pour analyser ces données et prendre des décisions.

Comme tous les types de réseaux, les RCSF utilisent une architecture de communication en couches, ce sont les cinq premières couches du modèle OSI : la couche physique, la couche liaison de données, la couche réseau, la couche transport et la couche application. Chaque couche a son propre rôle et ses propres protocoles pour atteindre son objectif.

L'objectif d'un RCSF n'est pas seulement la communication elle-même, mais il est soumis à de fortes contraintes énergétiques, par conséquence, d'autres unités doivent être ajouté afin de gérer la consommation d'énergie, la mobilité des nœuds et l'ordonnancement des tâches.

Ces plans de gestions d'énergie, de mobilité et de tâche aident les nœuds capteurs à coordonner les tâches et minimiser la consommation d'énergie. Ils sont donc nécessaires pour que les nœuds capteurs puissent collaborer ensemble, afin d'acheminer les données dans un

réseau mobile et de partager les ressources entre eux en utilisant efficacement l'énergie disponible. Ainsi, le réseau peut prolonger sa durée de vie.

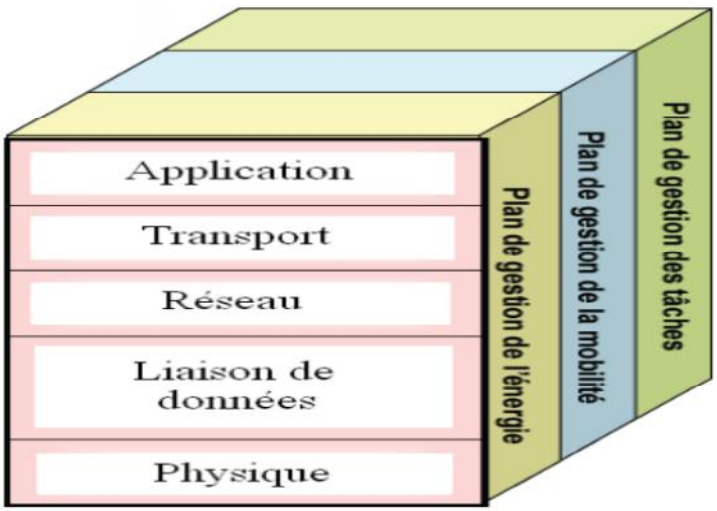


Figure I.4 : Pile protocolaire dans les RCSF.[34]

Couche application

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

Elle apporte sa contribution dans les réseaux de capteurs sans fil en satisfaisant les différents exigences du client, à travers des collaborations avec les autres couches de la pile protocolaire, dans le but de pouvoir proposer des services pour la gestion de l'énergie, la synchronisation, etc.

Couche transport

Elle est chargée du transport des données, de leur découpage en paquet, du contrôle de flux, de la conservation de l'ordre des paquets et de la gestion des éventuelles erreurs de transmission.

Couche réseaux

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

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

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

- l'absence d'adressage fixe des nœuds tout en utilisant un adressage basé-attribut.
- L'établissement des communications multi-sauts.

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

Parmi ces protocoles, nous citons : LEACH (Low-Energy Adaptive Clustering Hierarchy)

Couche liaison de données

La couche de liaison de données spécifie comment les données sont expédiées entre deux nœuds/routeurs dans une distance d'un seul saut.

Elle est responsable du multiplexage des données, du contrôle d'erreurs, de l'accès au média, etc. Ainsi elle assure la liaison point à point et multipoints dans le réseau.

La couche de liaison de données garantie une faible consommation d'énergie et minimiser les collisions entre les données diffusées par les nœuds voisins.

Couche physique

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

Elle est responsable de la sélection de fréquence, la génération de la fréquence porteuse, la détection du signal, la modulation/démodulation et le cryptage/décryptage des informations.

La consommation d'énergie au niveau de la couche physique peut être affectée par l'environnement de l'application, le choix du type de la modulation ou la bande de fréquence utilisée.

Il est avantageux en matière d'économie d'énergie que le concepteur de la couche physique choisisse une transmission à multi-sauts plutôt qu'une transmission directe qui nécessite une puissance de transmission très élevée.

Plan de gestion d'énergie

La couche de gestion d'énergie contrôle la manière d'utiliser l'énergie par le nœud capteur, et gère la consommation d'énergie selon le mode de fonctionnement employé (capture, calcul, et communication par radio). Par exemple pour éviter de recevoir des messages redondants, le nœud capteur change son mode en « Off » après une réception d'un message d'un de ses voisins. En outre un nœud capteur annonce à ses voisins qu'il a atteint un bas niveau d'énergie, par conséquence il ne va pas participer au routage des données, et l'énergie restante va être utilisée pour capter et détecter des tâches.

Plan de gestion de la mobilité

La couche de gestion de mobilité détecte et enregistre le mouvement/mobilité des nœuds capteurs. En utilisant ces positions, les nœuds capteurs peuvent connaitre qui sont leurs voisins. Parfois une auto-organisation des nœuds est nécessaire à cause de la destruction de

quelques nœuds. Dans ce cas, la couche de gestion de mobilité doit être capable de faire changer la position des nœuds.

- **Plan de gestion des taches**

La couche de gestion des tâches assure la coopération des efforts des nœuds capteurs, elle ordonnance les événements captés, et les tâches détectées dans une zone de capture spécifique. Par conséquent, les nœuds capteurs qui appartiennent à la même zone de capture ne sont pas obligés d'effectuer les tâches de capture en même temps. Selon leur niveau d'énergie, quelques nœuds capteurs peuvent accomplir des tâches de capture mieux que d'autres.

I.4.Quelques applications des réseaux de capteurs

La diminution de taille et de coût des micro-capteurs, l'élargissement de la gamme des types de capteurs disponibles (thermique, optique, vibrations, ...) et l'évolution des supports de communication sans fil ont élargi le champ d'application des réseaux de capteurs. Les RCSF peuvent être utilisés dans plusieurs applications [3, 4, 5, 6]. Parmi elles, nous citons :

·Découverte de catastrophes naturelles *:* on peut créer un réseau autonome en dispersant les nœuds dans la nature. Des capteurs peuvent ainsi signaler des événements tels que les feux de forêts, les tempêtes, les volcans ou les inondations. Ceci permet une intervention beaucoup plus rapide et efficace des secours [7].

· Détection d'intrusions *:* en plaçant à différents points stratégiques des capteurs, on peut ainsi prévenir des cambriolages ou des passages de gibier sur une voie de chemin de fer (par exemple) sans avoir à recourir à de coûteux dispositifs de surveillance vidéo.

· Gestion de stock *:* on pourrait imaginer devoir stocker des denrées nécessitant un certain taux d'humidité et une certaine température. Dans ces applications, le réseau doit pouvoir collecter ces différentes informations et alerter en temps réel si les seuils critiques sont dépassés.

· Contrôle de la pollution *:* des capteurs au-dessus d'un emplacement industriel offrent la possibilité de détecter et de contrôler des fuites de gaz ou de produits chimiques. Ces applications permettent de donner l'alerte en un temps record et de pouvoir suivre l'évolution de la catastrophe [8].

·Agriculture *:* des nœuds peuvent être incorporés dans la terre et on peut interroger le réseau sur l'état du champ et déterminer par exemple les secteurs les plus secs afin de les arroser en priorité. On peut aussi imaginer équiper des troupeaux de bétail de capteurs pour

connaître en tout temps, leur position ce qui éviterait aux éleveurs d'avoir recours à des chiens berger.

· ***Surveillance médicale*** *:* en implantant sous la peau de mini capteurs vidéo, on peut recevoir des images d'une partie du corps en temps réel sans aucune chirurgie. On peut ainsi surveiller la progression d'une maladie ou la reconstruction d'un muscle [9].

· ***Surveillance de barrages*** *:* on peut inclure sur les parois des barrages des capteurs qui permettent de calculer en temps réel la pression exercée. Il est donc possible de réguler le niveau d'eau si les limites sont atteintes. On peut aussi imaginer inclure des capteurs entre les sacs de sables formant une digue de fortune. La détection rapide d'infiltration d'eau peut servir à renforcer le barrage en conséquence. Cette technique peut aussi être utilisée pour d'autres constructions tels que ponts, voies de chemins de fer, routes de montagnes, bâtiments et autres ouvrages d'art.

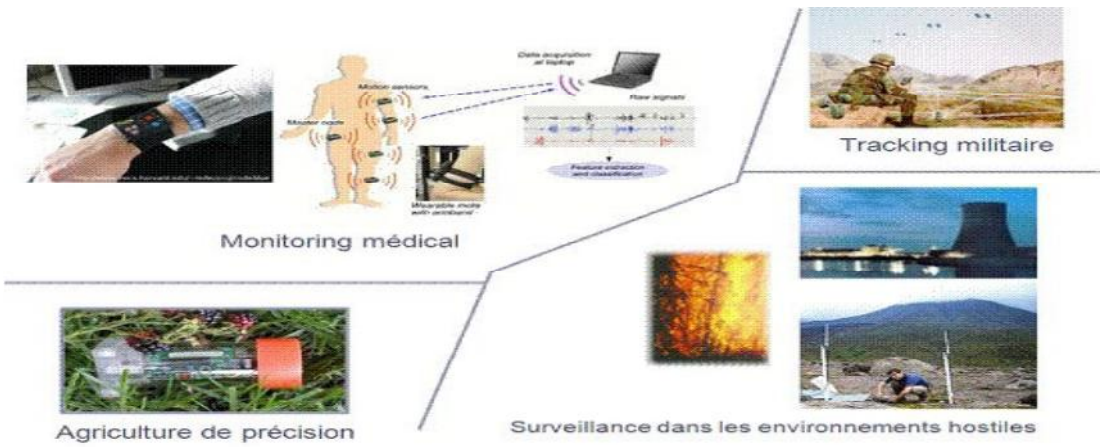


Figure I.5 : Application des RCSF.[43]

I.5. Contraintes de conception des RCSF

Les principaux facteurs et contraintes influençant l'architecture des réseaux de capteurs peuvent être résumés comme suit :

· **La tolérance aux fautes** [12, 13] : la tolérance aux fautes est la capacité de maintenir les fonctionnalités du réseau en présence de fautes. La fiabilité des réseaux de capteurs sans fil est affectée par des défauts qui se produisent à cause de diverses raisons telles que le mauvais fonctionnement du matériel ou à cause d'un manque d'énergie. Ces problèmes n'affectent pas le reste du réseau.

· **Le facteur d'échelle** (Scalability) [14]: le nombre de nœuds de capteurs augmente sur un réseau sans fil et ce nombre peut atteindre le million. Un nombre aussi important de nœuds

engendre beaucoup de transmissions entre les nœuds et peut imposer des difficultés pour le transfert de données.

· Les coûts de production [11]: souvent les réseaux de capteurs sont composés d'un très grand nombre de nœuds. Le prix d'un nœud est critique afin de pouvoir concurrencer un réseau de surveillance traditionnel.

· L'environnement : les capteurs sont souvent déployés en masse dans des endroits tels que des champs de bataille, à l'intérieur de grandes machines, au fond d'un océan, dans des champs biologiquement ou chimiquement souillés [15],... Par conséquent, ils doivent pouvoir fonctionner sans surveillance dans des régions géographiques éloignées.

· La topologie de réseau [16]: le déploiement d'un grand nombre de nœuds nécessite une maintenance de la topologie. Cette maintenance consiste en trois phases : déploiement, post-déploiement (les capteurs peuvent bouger, ne plus fonctionner,...) et redéploiement de nœuds additionnels.

· Les contraintes matérielles [17]: la principale contrainte matérielle est la taille du capteur. Les autres contraintes sont la consommation d'énergie qui doit être moindre pour que le réseau survive le plus longtemps possible, qu'il s'adapte aux différents environnements (fortes chaleurs, eau,..), qu'il soit autonome et très résistant vu qu'il est souvent déployé dans des environnements hostiles.

· Les médias de transmission : dans un réseau de capteurs, les nœuds sont reliés par une architecture sans fil. Pour permettre des opérations sur ces réseaux dans le monde entier, le média de transmission doit être standardisé. On utilise le plus souvent l'infrarouge, le Bluetooth [18] et les communications radio [19].

· La consommation d'énergie [20]: un capteur, de par sa taille, est limité en énergie (<1.2V). Dans la plupart des cas le remplacement de la batterie est impossible. Ce qui veut dire que la durée de vie d'un capteur dépend grandement de la durée de vie de la batterie. Dans un réseau de capteurs (multi-sauts) chaque nœuds collecte des données et envoie/transmet des valeurs.

Le dysfonctionnement de quelques nœuds nécessite un changement de la topologie du réseau et un ré-routage des paquets. Toutes ces opérations sont gourmandes en énergie, c'est pour cette raison que les recherches actuelles se concentrent principalement sur les moyens de réduire cette consommation [21].

I.6. Consommation d'énergie dans les RCSF

La première étape dans la conception de système énergétique de capteurs consiste à analyser les caractéristiques de consommation d'énergie d'un nœud de capteur sans fil. Cette analyse systématique de l'énergie d'un nœud capteur est extrêmement importante pour identifier les problèmes dans le système énergétique pour permettre une optimisation efficace. L'énergie consommée par un capteur est principalement dû aux opérations suivantes : la détection, le traitement et la communication [22].

I.6.1. Energie de capture

Les sources de consommation d'énergie des nœuds pour les opérations de détection ou de capture sont : l'échantillonnage, la conversion analogique numérique, le traitement de signal et l'activation de la sonde de capture [23].

I.6.2. Energie de traitement

L'énergie de traitement est composée de deux sortes d'énergie: l'énergie de commutation et l'énergie de fuite. L'énergie de commutation est déterminée par la tension d'alimentation et la capacité totale commutée au niveau logiciel (en exécutant un logiciel). Par contre, l'énergie de fuite correspond à l'énergie consommée lorsque l'unité de calcul n'effectue aucun traitement. En général, l'énergie de traitement est faible par rapport à celle nécessaire pour la communication.

I.6.3. Energie de communication

L'énergie de communication se décline en trois parties : l'énergie de réception,
l'énergie de l'émission et l'énergie en état de veille. Cette énergie est déterminée par la quantité des données à communiquer et la distance de transmission, ainsi que par les propriétés physiques du module radio. 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. Notons que l'énergie de communication représente la portion la plus grande de l'énergie consommée par un nœud capteur.

I.7. Techniques de minimisation de la consommation d'énergie

Dans les réseaux ad hoc, la consommation de l'énergie a été considérée comme un facteur déterminant mais pas primordial car les ressources d'énergie peuvent être remplacées par l'utilisateur. Ces réseaux se focalisent plus sur la QoS (Quality of Service) que sur la consommation de l'énergie. Par contre, dans les réseaux de capteurs, la consommation d'énergie est très importante puisque généralement les capteurs sont déployés dans des zones inaccessibles. Ainsi, il est difficile voire impossible de remplacer les batteries après leur

épuisement. De ce fait, la consommation d'énergie au niveau des capteurs a une grande influence sur la durée de vie du réseau. Après la description des principales causes de consommation d'énergie dans les RCSF, nous présentons dans ce qui suit les différentes techniques utilisées pour minimiser cette consommation.

L'énergie du capteur peut être économisée soit au niveau de la capture, au niveau de traitement ou au niveau de la communication.

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

B. L'énergie de calcul peut être optimisée en utilisant :

● 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 [25].

I.8. Les critères de performance des protocoles de routage en RCSF

La performance des réseaux de capteurs sans fil est fondée sur les facteurs suivants :

● Evolutivité : l'évolutivité est un facteur important dans les réseaux de capteurs sans fil. Une zone de réseau n'est pas toujours statique, elle change selon les besoins des utilisateurs. Tous les nœuds dans le domaine du réseau doivent être évolutifs ou être en mesure de s'adapter aux changements dans la structure du réseau en fonction de l'utilisateur.

● L'énergie : chaque nœud utilise peu d'énergie pour des activités telles que la détection, le traitement, le stockage et la transmission. Un nœud dans le réseau doit savoir combien d'énergie sera utilisée pour effectuer une nouvelle tâche à laquelle il est soumis. L'énergie consommée peut varier selon le type de fonctionnalité ou l'activité qu'il a à accomplir.

● Le temps de traitement: il se réfère au temps pris par le nœud dans le réseau pour assurer l'ensemble de l'opération commençant par la détection, le traitement des données ou le stockage de données, la transmission ou la réception sur le réseau.

● Le schéma de transmission: la transmission de données par les nœuds de capteurs vers la destination ou la station de base se fait par un schéma de routage à un seul saut ou à multi saut.

● La capacité du réseau : tous les nœuds du réseau de capteurs utilisent certaines ressources du réseau qui les aident à accomplir certaines activités comme la détection ou la transformation.

● **Synchronisation** : dans les communications radio entre les nœuds de capteurs d'un WSN, les capteurs écoutent en permanence les transmissions et consomment de l'énergie s'ils ne sont pas synchronisés les uns les autres. Pour cela, un nœud doit avoir la même notion de temps pour se mettre en veille et se réveiller que ses voisins.

● **Contrôle de paquets**: un paquet envoyé avant la transmission entre deux nœuds est appelé le paquet de contrôle. Le paquet de contrôle contient le nombre de bits de données envoyés, l'adresse du nœud de destination et certaines informations qui contribuent à éviter les collisions pendant la transmission.

I.9.Classification des protocoles de routage dans les RCSF

Les protocoles de routage dans les réseaux peuvent être classés selon deux concepts :

- la structure du réseau.
- le type de protocole.

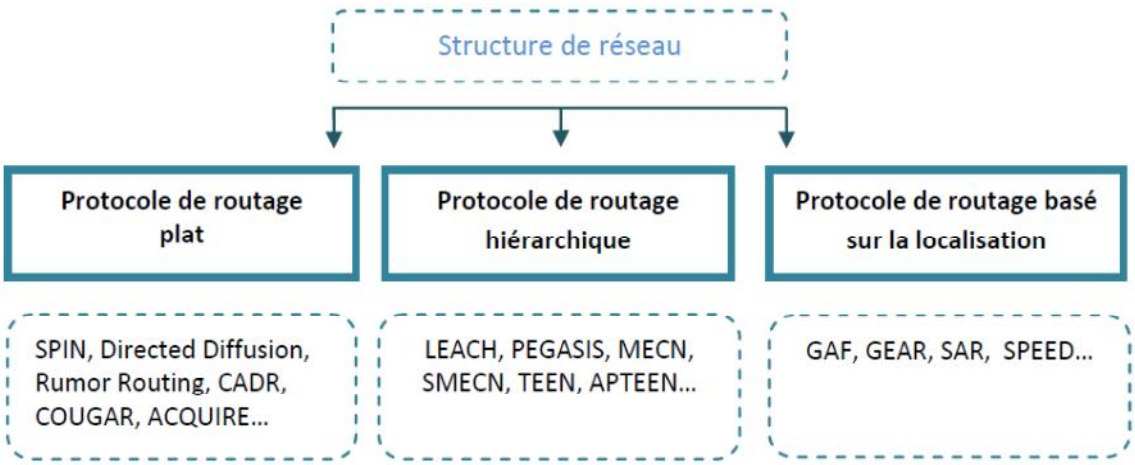


Figure I.6:Protocoles de routage pour les RCSF selon la structure du réseau. [35]

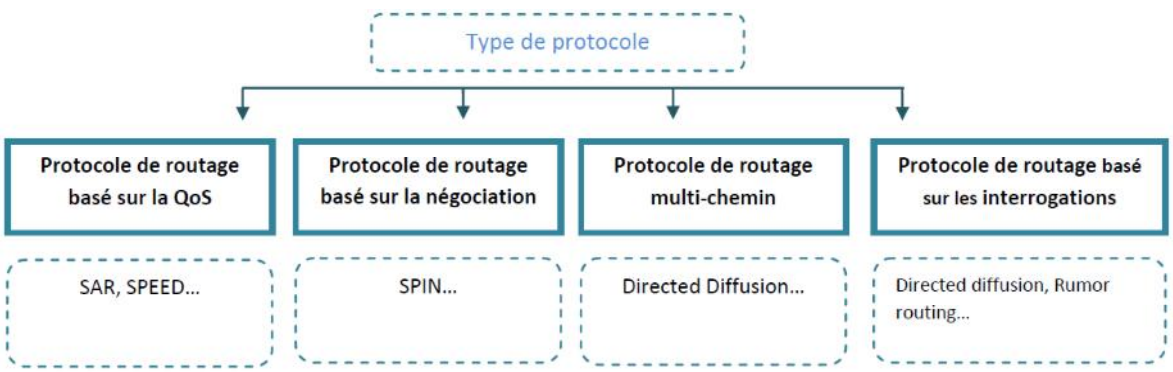


Figure I.7 : Protocoles de routage pour les RCSF selon le type de protocole. [35]

I.9.1.Selon le type de protocole

- **Protocole de routage multi-chemin**

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

DD (Directed Diffusion) est un des protocoles de cette classe, qui utilise un schéma de nommage sous forme de paire (Attribut-valeur) pour les requêtes et les données. Chaque nœud qui recense un évènement crée et diffuse un gradient au voisinage direct. Chaque nœud puits diffuse l'intérêt (qui est une demande ou une interrogation) vers tous les voisins, par la suite le gradient spécifie la direction dans laquelle les données répondant à l'intérêt seront envoyées. De cette manière, plusieurs routes reliant la station de base à la source de données peuvent être trouvées, puis la meilleure route sera renforcée pour éviter la redondance.

Son fonctionnement se résume en trois phases :

a). La propagation des intérêts.

b). Etablissement des gradients.

c). La livraison des données et renforcement des chemins.

Le DD définit des règles de renforcement positif et négatif pour la sélection des chemins.

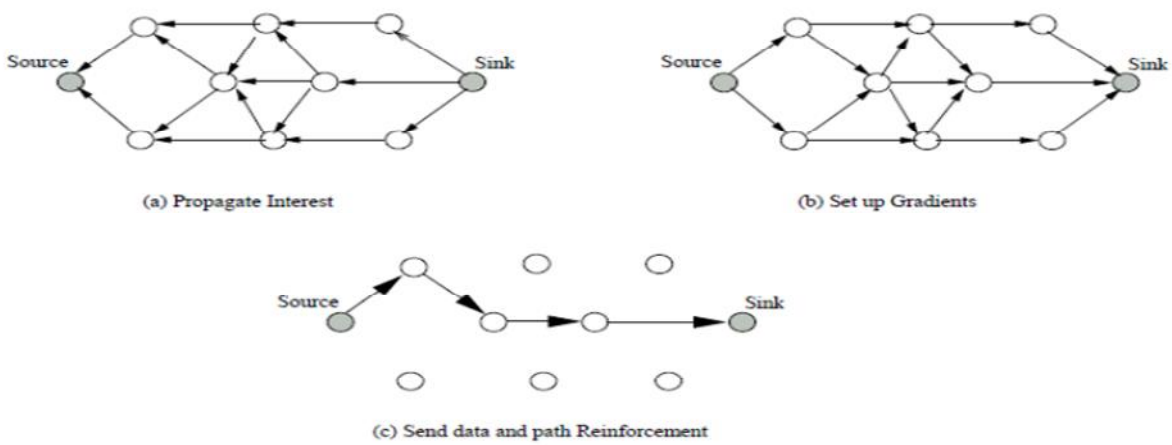


Figure I.8: Fonctionnement de Directed Diffusion [36]

- **Protocole de routage basé sur la négociation des données**

En détectant le même phénomène, les nœuds capteurs inondent le réseau par les mêmes paquets de données. Ce problème de redondance peut être résolu en employant des protocoles de routage basés sur la négociation. En effet, avant de transmettre, les nœuds capteurs négocient entre eux leurs données en échangeant des paquets de signalisation spéciales, appelés *META-DATA.* Ces paquets permettent de vérifier si les nœuds voisins disposent des mêmes données à transmettre [26]. Cette procédure garantit que seules les informations utiles seront transmises et élimine la redondance des données.

SPIN (*Sensor Protocol for information via Negociation)* est un exemple de protocole de cette classe qu'on va étudier dans le chapitre suivant.

- **Protocole de routage basé sur les interrogations**

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

RR (RumourRouting) [27] est un des protocoles de cette classe, l'idée est de transmettre les requêtes aux nœuds qui ont observé un événement particulier. Quand un nœud détecte un événement, il l'ajoute à sa table locale et génère un agent. L'agent parcoure le réseau afin de propager des informations sur les événements locaux aux nœuds distants. Quand un nœud génère une requête pour un événement, les nœuds qui connaissent l'itinéraire peuvent répondre en référant la table d'événements.

- **Protocole de routage basé sur la QoS**

Ce type de routage est utilisé dans les applications qui ont des exigences temps réel. Par exemple, le système de sécurité ou la détection d'intrusion et les applications de surveillance (centrales nucléaires, applications militaires). Dans ces protocoles, le réseau doit équilibrer entre la consommation d'énergie et la qualité des données. En particulier, le réseau doit satisfaire certaines métriques de QoS comme : le retard, l'énergie et la largeur de bande passante.

SPEED [27] est un des protocoles de routage de cette classe qui est basé sur la QoS.Sa caractéristique principale est la garantie d'un délai de bout-en-bout raisonnable qui répond au

besoin de l'application. Le prochain saut est choisi parmi les voisins qui sont plus proches de la destination tout en assurant une vitesse de livraison des paquets constante. Grace à cette spécification, il est le plus approprié pour les applications avec des exigences temps réel.

I.9.2.Selon la structure de réseau

- **Les protocoles de routage plat (flat based-routing)**

Ces protocoles considèrent que tous les nœuds sont identiques, c'est à dire ont les mêmes fonctions à exécuter sauf le nœud de contrôle (*sink*) qui est chargé de collecter toutes les informations issues des différents nœuds capteurs pour les transmettre vers l'utilisateur final. La décision d'un nœud de router des paquets vers un autre dépendra de sa position et pourra être remise en cause au cours du temps.

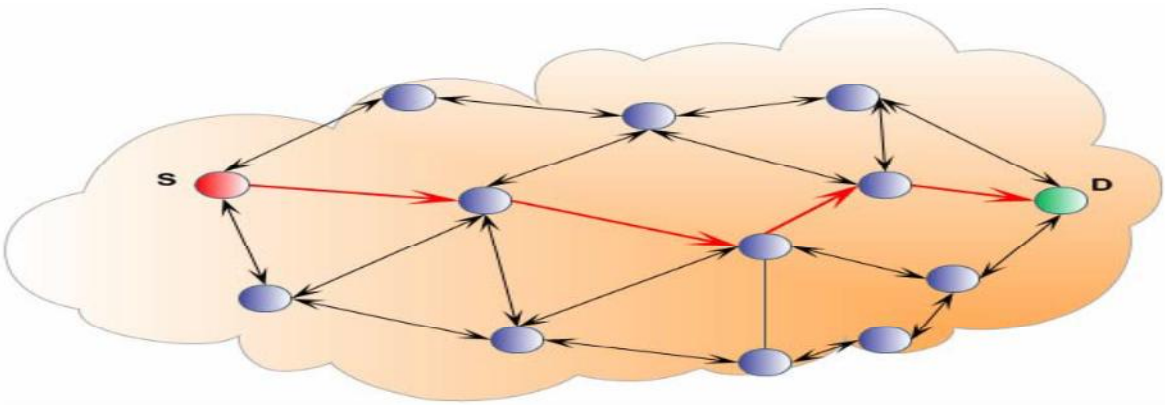


Figure I.9 :Topologie plat. [37]

COUGAR [27] est un des protocoles de cette classe, où le réseau est vu comme un énorme système de base de données répartie où quelques nœuds contenant les renseignements sont temporairement inaccessibles. Les données produites par le réseau de capteurs sont modélisées comme une table relationnelle. Dans cette table, chacun des attributs représente soit des informations sur le nœud capteur ou bien des données produites par ce nœud. Pour pouvoir interroger les nœuds du réseau, COUGAR fournit une interface semblable à SQL (*Structured Query Language*) étendue, au niveau du puits. Cette interface permet au puits de générer des requêtes pour interroger un nœud spécial du réseau, appelé leader. L'approche COUGAR effectue une agrégation partielle au niveau des nœuds. Chaque nœud maintient une liste d'attente contenant les nœuds fils qui doivent envoyer les paquets. Le nœud n'émet le paquet agrégé au prochain saut que s'il a reçu les paquets de tous les nœuds de la liste d'attente. Cependant, un nœud peut devenir inaccessible à cause du mouvement ou d'un problème de batterie. Pour cela, COUGAR utilise un timer afin d'éviter une attente indéfinie.

- **Les protocoles de routage hiérarchique**

Ces protocoles fonctionnent en confiant des rôles différents aux nœuds du réseau. Certains nœuds sont sélectionnés pour exécuter des fonctions particulières. Un nœud peut être, par exemple, une passerelle pour un ensemble de nœuds. Dans ce cas, le routage devient plus simple, puisqu'il s'agit de passer par les passerelles pour atteindre le nœud destination qui lui est directement attaché.

Le principe des protocoles de routage hiérarchique est basé essentiellement sur les nœuds passerelles. En fait, les nœuds ordinaires savent que si le destinataire n'est pas dans leur voisinage direct, il suffit d'envoyer la requête à la passerelle qui la prendra en charge. À son tour, elle transmettra cette requête vers le nœud ciblé. Ce type de routage présent de nombreux avantages pour les réseaux dont leurs nœuds sont sédentaires et disposent de suffisamment d'énergie [26].

TEEN (*Threshold sensitive Energy Efficient Network*) est un des protocoles de cette classe, Manjeshwar et Agrawal [26] ont proposé une technique de clustering appelée TEEN qui est conçu pour être sensible aux changements soudains des attributs captés tels que la température. L'architecture du réseau est basée sur un groupement hiérarchique où des nœuds les plus proches forment des clusters. Après la construction des clusters, le *cluster-Head* diffuse deux seuils aux nœuds, qui sont la valeur minimale d'un attribut pour pouvoir être transmit et le degré minimale du changement de cet attribut. Ce protocole consomme moins d'énergie que l'APTEEN. Le TEEN adaptatif (APTEEN) est une extension du TEEN basée sur la capture périodique des données et la réaction aux événements temps-réel. Quand la station de base forme les clusters, les cluster-Head diffusent les attributs, les seuils et le plan de transmission à tous les nœuds et effectuent également l'agrégation des données afin d'économiser l'énergie [26].

- **Les protocoles de routage avec localisation géographique**

Un routage est dit géographique lorsque les décisions de routage sont basées sur la position des nœuds. Les prés-requis pour effectuer un routage géographique dans un réseau ad hoc sont :

- Tous les nœuds possèdent un moyen de localisation, soit un système natif comme le GPS (*Global Position System*), soit un système logiciel comme un protocole de localisation.
- Un nœud source connaît toujours la position du nœud destinataire. Pour ce faire, soit tous les nœuds connaissent les positions initiales de tous les nœuds, soit un service de localisation doit être utilisé.

I .10 Conclusion

La technologie des réseaux de capteurs reste très prometteuse, et leur défis majeur est de trouver des protocoles de routage qui permettent, à la fois, de :

- consommer le moins d'énergie possible,
- assurer la connectivité du réseau et la couverture du champ surveillé,
- assurer une livraison fiable et rapide,
- Tolérance aux pannes,
- s'adapter aux changements de topologie …

Dans ce chapitre, nous avons classifié quelques protocoles de routage pour les RCSFs. De façon, générale, ces protocoles sont classifiés selon la structure du réseau, et le type de protocole.

Dans le chapitre suivant nous allons étudier protocole de routage plat SPIN, basés sur la négociation.

Chapitre II:
Le protocole de routage SPIN

II.1Introduction

Après avoir présenté un bref aperçu sur les différentes catégories de protocoles de routage pour les réseaux de capteurs, nous nous intéresserons dans ce chapitre au protocole Sensor Protocol for Information via Negotiation(SPIN) qui est le premier protocole centré-données dans la littérature, classé dans des types d'application *event-driven*. Il a été proposé par Heinzelman et al, [27] et est devenu l'un des protocoles les plus répandus dans les réseaux de capteurs. Sa création représente une importante avancée dans le domaine du routage. Il repose sur un modèle de négociation afin de propager l'information dans un réseau de capteur.

Pour cela, nous consacrons ce chapitre à une étude détaillée du fonctionnement de SPIN mais avant, on va donner une brève explication du paradigme de communication centré-données et les applications *event-driven*. Enfin on termine par un aperçu sur le protocole hiérarchique LEACH.

II.2 C'est quoi le paradigme de communication Centrés-Données

A la différence des réseaux ad hoc classiques, les nœuds d'un réseau de capteurs sont dépourvus d'une identification globale (tel que l'adressage IP). Cette absence d'identification explicite est due à certaines caractéristiques des réseaux de capteurs. Tout d'abord, l'obtention et la gestion d'adresses nécessitent une étape de configuration qui peut être complexe, à cause du nombre très élevé de capteurs qui peuvent être déployés aléatoirement. De plus, étant donné la nature des applications des réseaux de capteurs, l'utilisateur ne s'intéresse pas à communiquer avec un nœud particulier du réseau mais envoie des requêtes, à tous les nœuds du réseau ou à une partie de celui-ci, afin d'identifier les nœuds concernés. Ces caractéristiques ont mené à la conception d'un nouveau type de protocoles appelé *centrés-données*. Dans ce dernier, le routage ne se fait pas en fonction d'une adresse de destination (comme c'est le cas pour les réseaux ad hoc), mais suivant les données disponibles au niveau des capteurs ; donc les communicants sont identifiés par leurs données, et tout le système (routage, interrogation,…) doit être régit par cette propriété. Ainsi, le système peut être vu comme une base de données distribuée, ou les nœuds forment des tables virtuelles, alimentées par les données captées.

Un exemple d'une approche data-centric est présenté dans la figure II.1, dans lesquelles les données provenant des deux sources sont agrégées au nœud B. Ensuite, la donnée combinée (1+2) est envoyée de B vers la destination.

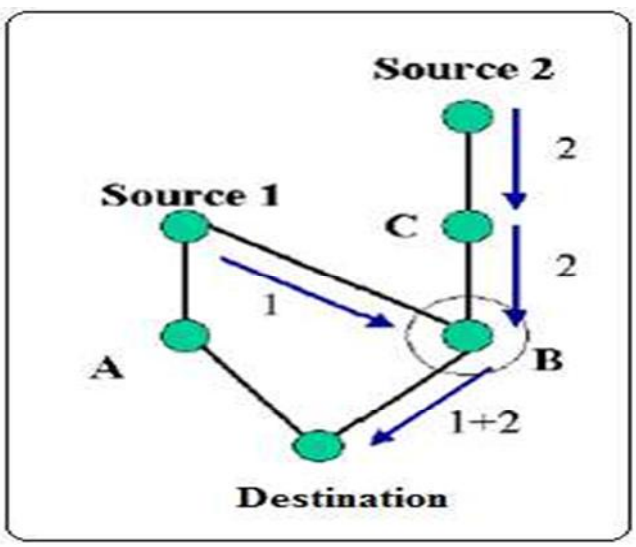


Figure II.1:Le routage data-centric. [38]

II.3 C'est quoi le type d'application *event-driven*

Plusieurs protocoles de routages pour les RCSF ont été largement étudiés et différentes études ont été publiées pour classifier les protocoles. Le type d'application est l'un des critères proposé pour la classification des protocoles.

La méthode de captage des données dans un RCSF dépend de l'application et de l'importance de la donnée. De ce fait, les RCSF peuvent être catégorisés comme *time-driven* ou *event-driven.*

- *Event-driven :* dans des applications temps réel, les capteurs doivent réagir immédiatement à des changements soudains des valeurs captées.
- *Time-driven :* un réseau *time-driven* est approprié pour des applications qui nécessitent un prélèvement périodique des données.

II.4 Protocole SPIN

II.4.1Definition

Le protocole SPIN est conçu pour améliorer les techniques classiques de routage comme l'inondation. En effet, il permet à un nœud de non pas diffuser les données à tous ses voisins mais seulement aux voisins intéressés par ces données. Cela surmonte le problème d'implosion dû aux redondances que génère la technique d'inondation. De plus, le protocole SPIN permet aux nœuds de vérifier continuellement leur niveau d'énergie. Un nœud peut donc changer son mode de fonctionnement s'il vérifie bien que son énergie a considérablement diminuée.

Le protocole SPIN nomme les données par des descripteurs appelés métadonnées. Un nœud voulant transmettre des données commence tout d'abord par diffuser ces métadonnées à tous ses voisins. Ceux-ci répondent par une requête s'ils sont intéressés et récupèrent ainsi ces données.

II.4.2 But de protocole SPIN

. Le but de SPIN est de pallier aux problèmes de l'inondation, qui sont :

- L'implosion due à la duplication inutile des réceptions d'un même message.
- Le chevauchement lié au déploiement dense des capteurs. En utilisant l'inondation, les capteurs d'une zone émettrons tous la même donnée (ou presque).

- L'ignorance des ressources, car l'inondation ne prend pas en considération les ressources des nœuds.

Ces trois problèmes affectent grandement la durée de vie et les performances du réseau. Pour les résoudre, SPIN adopte deux principes :

- La négociation : pour éviter le problème d'implosion, SPIN précède l'émission d'une donnée par sa description, en utilisant la notion de métadonnées. Le récepteur aura le choix par la suite d'accepter la donnée ou non. Ce mécanisme permet aussi de régler le problème de chevauchement.
- L'adaptation aux ressources : d'une manière continue, les nœuds contrôlent leur niveau d'énergie. Le protocole SPIN accommode son exécution suivant l'énergie restante du capteur, et modifie en conséquence le comportement du nœud.

II.4.3 Type de messages

- Message ADV (ADVertising) : un message ADV contenant une description de la donnée en question.
- Message REQ (REQuête) : un message REQ a généralement la même structure qu'un message ADV.
- Message DATA (DATA) : un message DATA contient la donnée et sa description.

II.4.4 Fonctionnement

Les communications dans SPIN se font en trois étapes :

- Lorsqu'un nœud veut émettre une donnée, il émet d'abord un message ADV Un nœud recevant un message ADV, consulte sa base d'intérêt. S'il est intéressé par cette information, il émet un message REQ vers son voisin.
- En recevant un message REQ, l'émetteur transmet à l'intéressé la donnée sous forme d'un message DATA.

La figure suivante illustre ces trois étapes :

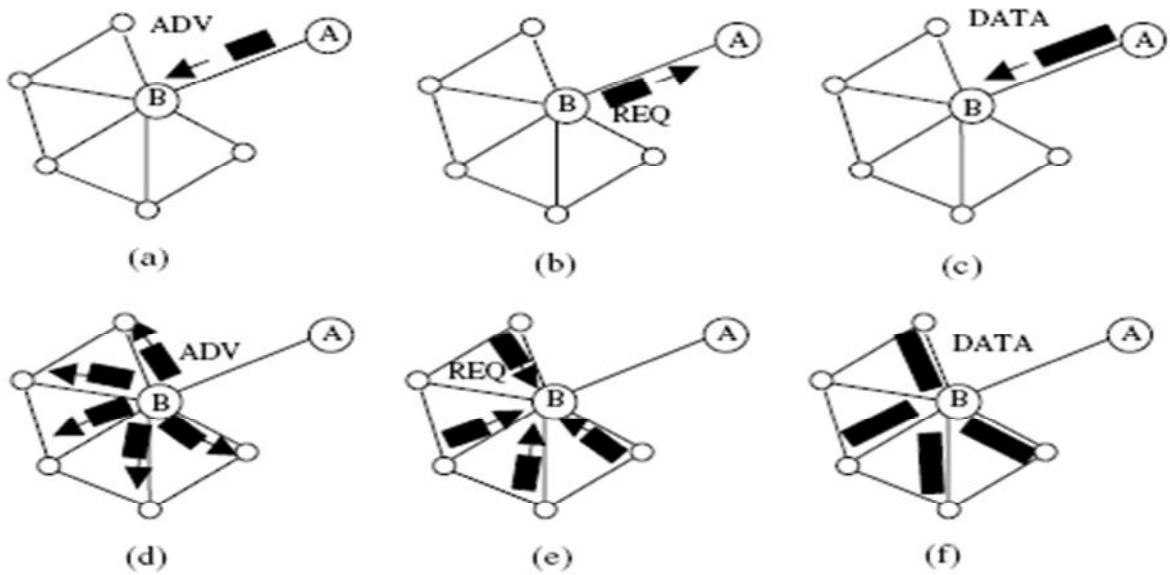


Figure II.2 : Fonctionnement de SPIN. [39]

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

Remarque : Lorsque le nœud s'aperçoit que son énergie est en dessous d'un certain seuil, il change son mode de fonctionnement et ne répond à aucun message ADV.

II.5 La famille du protocole SPIN

La famille des protocoles SPIN comprend :

1. SPIN-PP (Sensor **P**rotocol for **I**nformation via **N**egotiation-**P**oint to **P**oint) : ce protocole est conçu pour la communication point-a-point, en supposant que deux nœuds peuvent communiquer entre eux sans interférence avec d'autres nœuds. Ce protocole suppose également que la consommation d'énergie ne constitue pas une contrainte, et les paquets ne sont jamais perdus.

2. SPIN-EC (Sensor **P**rotocol for **I**nformation via **N**egotiation-**E**nergy **C**onservation) : ce protocole ajoute au précédent une heuristique de consommation d'énergie, un nœud participe au processus de routage si seulement il peut accomplir toutes les étapes du protocole sans atteindre un niveau faible d'énergie.

3. SPIN-BC (Sensor **P**rotocol for **I**nformation via **N**egotiation-**B**roadcast **C**hanel : ce protocole a été défini pour les canaux de diffusion (*Broadcast Channels*), et possède

l'avantage que tout nœud se trouvant dans la portée de l'émetteur peut recevoir la diffusion, mais l'inconvénient se présente quand un nœud doit annuler sa transmission si le canal de diffusion est utilisée. Une autre différence par rapport au protocole précédent se montre dans le fait qu'un nœud n'envoie pas un message REQ immédiatement après la réception du message ADV, mais il initialise une horloge aléatoire, et envoi le message REQ après son expiration. Les autres nœuds ignorent donc tout message REQ reçu avant l'expiration de l'horloge initialisée, ceci permettra d'éviter la réception redondante des messages de requête.

4. SPIN-RL(Sensor **P**rotocol for **I**nformation via **N**egotiation-**R**eliable version of SPIN-BC) : ce protocole a introduit deux modifications principales : premièrement, chaque nœud garde une trace des messages ADV reçus, il ré-envoi sa requête de donnée si le nœud sollicité ne répond pas dans un délai déterminé. Deuxièmement, les nœuds ont une limite respectée pour la fréquence de transmission de données, chaque nœud attend, pour une période de temps prédéterminée, pour servir une requête réclamant la même pièce de donnée.

II.6 Avantages et inconvénients de SPIN

➢ Avantage

Un des avantages de SPIN est que les changements n'ont qu'une influence locale puisque chaque nœud doit uniquement connaître ses voisins à distance d'un simple saut. SPIN diminue la consommation d'énergie par rapport à l'inondation classique et la négociation par métadonnées divise presque par deux les données superflues.

➢ Inconvénient

Le mécanisme d'annonce des données ne peut pas garantir la délivrance de ces données. Ce sera le cas si les nœuds qui sont intéressés par les données sont lointains du nœud de source et que certains nœuds entre la source et la destination ne sont pas intéressés par ces données.

II.7 Un aperçu sur LEACH

LEACH (*Low Energie Adaptive Clustering Hierarchy*) est un des protocoles hiérarchique, et c'est l'algorithme de routage le plus populaire pour les RCSFs. L'idée est de former des clusters de nœuds de capteurs basés sur les zones où il y a un fort signal reçus, puis utiliser des *Clusters-Head* locaux comme passerelles pour atteindre la destination. Cela permet d'économiser de l'énergie car les transmissions ne sont effectuées que par les *Clusters-Head* plutôt que par tous les nœuds capteurs. LEACH suppose que chaque nœud du réseau peut communiquer directement avec le puits ; alors que, les nœuds non- *Clusters-Head* ne peuvent

communiquer qu'avec leurs *Clusters-Head* choisi, en utilisant la technique TDMA (Time Division Multiple Access).Cette technique permet de minimiser les collisions en allouant à chaque nœud un intervalle de temps (*Slot*) pour transmettre ses données vers son CH. [27]

LEACH préconise, également, une agrégation de données au niveau des CHs pour plus de conservation d'énergie. Cependant, plusieurs critiques sont apportées au protocole LEACH relativement à ses hypothèses contraignantes de départ, à savoir :

- La possibilité de communiquer avec le puits à travers n'importe quel nœud du réseau exige une consommation d'énergie importante des nœuds lointains. Ce qui rend le protocole moins apte au passage à l'échelle.
- L'agrégation des données est centrée au niveau des CHs, ce qui les rend les maillons faibles du réseau.
- La rotation du rôle du CH sur l'ensemble des nœuds du cluster, permet d'une part d'équilibrer la consommation de l'énergie du cluster. Mais, génère une consommation d'énergie, car chaque rotation de CH nécessite une phase de diffusion pour faire connaitre le nouveau CH.
- LEACH ne garantit pas une distribution homogène des CHs sur le réseau, car le seul critère d'élection de CH est une probabilité aléatoire. Cela n'empêche pas une concentration des CHs dans une région limitée au détriment de l'ensemble du réseau. Plusieurs variantes de LEACH ont été proposées pour palier aux problèmes de la version originale, à savoir ; LEACH-C qui est une version centralisée, où l'algorithme s'exécute au niveau du puits pour permettre une meilleure distribution des CHs sur le réseau.

La figure II.3 montre le routage hiérarchique du protocole LEACH.

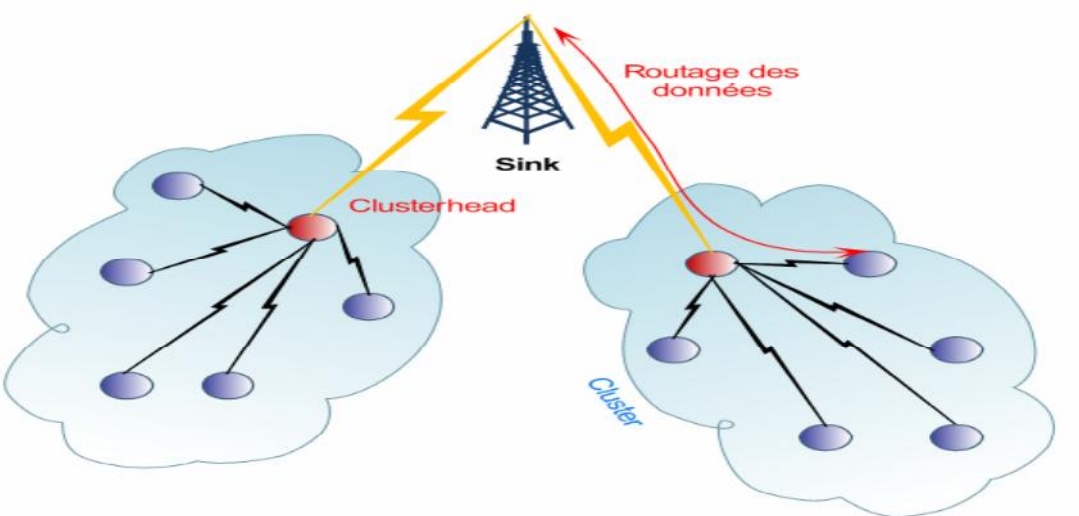


Figure II.3 : Routage hiérarchique de LEACH [40]

II.8 Comparaison de SPIN et LEACH

	Protocole SPIN	**Protocole LEACH**
Structure du réseau	plat	hiérarchique
basée sur la Négociation	Oui	Non
Agrégation de données	Oui	Oui
Localisation ***Initiateur de communication***	Non source	Oui source
Etablissement de la route	réactif	proactif
Multi chemin	oui	non

Tableau II.1 : Comparaison de SPIN et LEACH.

II.9 Conclusion

Dans ce chapitre nous avons présenté (les différent types de messages ainsi le fonctionnement et la famille) du protocole SPIN et un aperçu sur LEACH. Dans le chapitre suivant, nous allons présenter la simulation de notre application qui a pour but la réalisation d'un algorithme de routage plat SPIN et de comparer la consommation d'énergie avec le protocole LEACH.

Chapitre III: Implémentation et Simulation

III.1 Introduction

Le déploiement d'un réseau de capteurs nécessite une phase de simulation, afin d'assurer le bon fonctionnement de tous les dispositifs. En effet, pour les grands réseaux, le nombre de capteurs peut atteindre plusieurs milliers et donc un coût financier relativement important. Il faut donc réduire au maximum les erreurs de conception possibles en procédant à une phase de validation.

Dans le cadre du projet, nous avons mis en place une plateforme d'expérimentation qui a pour but de tester, de valider et de simuler le fonctionnement d'un réseau de capteurs. Sa principale fonction est de vérifier le comportement des capteurs développés avant même de les avoir déployé en situation réelle.

III.2 Description de l'application :

• Les Structures de données :

La structure des messages « ADV », « REQ» et « DATA» définit dans le fichier: « header.h» sont montrés ci-dessous :

ADV

```
typedef struct{
   uint16_t idd;//identificateur(numero) du noeud
   uint16_t idd_origine;//identificateur du noeud origine
   int8_t type_capt;//le type de captage

}ADV;
```

REQ

```
typedef struct{
   uint16_t idd;//identificateur(numero) du noeud
   uint16_t idd_origine;//identificateur du noeud origine
   int8_t type_capt;//le type de captage

}REQ;
```

DATA

```
typedef struct{
      uint16_t idd;//identificateur(numero) du noeud
      uint16_t idd_origine;//identificateur du noeud origine
      int8_t type_capt;//le type de captage
      uint16_t donnee;//represente la temperature captée
}DATA;
```

- ***Aspect algorithmique du programme :***

1. Le Composant configuration: mettre en relation les composants via leurs interfaces.

```
#include"header.h"
configuration Maco{
}
implementation{
//composants utilisés
components Main,GenericComm,DemoSensorC,MoM,TimerC,LedsC,RandomLFSR;

//initialisation
Main.StdControl->MoM;
Main.StdControl->GenericComm.Control;
Main.StdControl->TimerC;

//communication
MoM.SendMsgADV->GenericComm.SendMsg[AM_MSG];
MoM.ReceiveMsgADV->GenericComm.ReceiveMsg[AM_MSG];
MoM.ReceiveMsgREQ->GenericComm.ReceiveMsg[AM_MSG];
MoM.SendMsgREQ->GenericComm.SendMsg[AM_MSG];
MoM.SendMsgSink->GenericComm.SendMsg[AM_MSG];
MoM.ReceiveSink->GenericComm.ReceiveMsg[AM_MSG];
MoM.ReceiveMsgDATA->GenericComm.ReceiveMsg[AM_MSG];
MoM.SendMsgDATA->GenericComm.SendMsg[AM_MSG];
MoM.ReceiveReveil->GenericComm.ReceiveMsg[AM_MSG];
MoM.SendMsgReveil->GenericComm.SendMsg[AM_MSG];
```

```
//Timer
MoM.TimerCapt->TimerC.Timer[(uint8_t)unique("Timer")];
MoM.Timer->TimerC.Timer[(uint8_t)unique("Timer")];
MoM.TimerRELAY->TimerC.Timer[(uint8_t)unique("Timer")];
MoM.TimerREQ->TimerC.Timer[(uint8_t)unique("Timer")];
MoM.TimerInit->TimerC.Timer[(uint8_t)unique("Timer")];

MoM.ADC->DemoSensorC;
MoM.Random->RandomLFSR.Random;
MoM.Leds->LedsC;

}
```

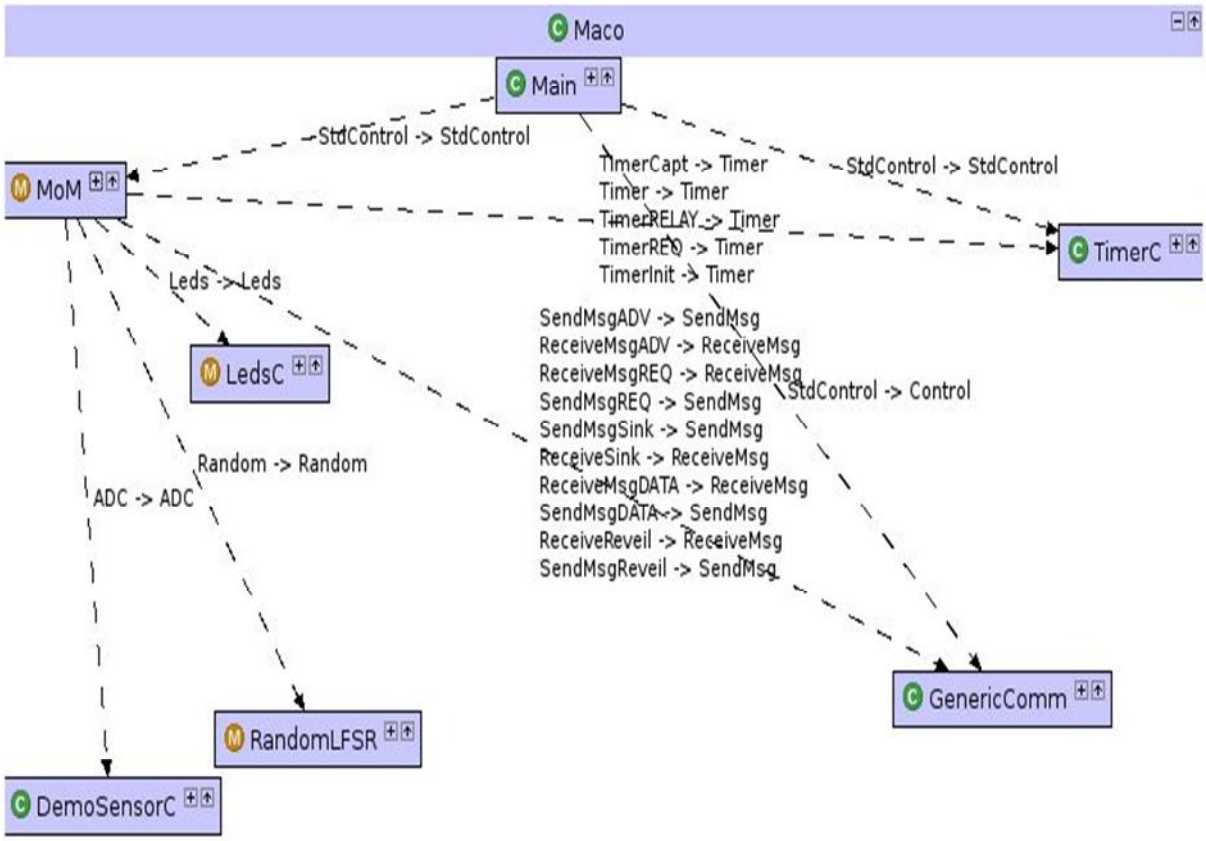


Figure III.1 : Représentation graphique du composant configuration

Les composants utilisés sont :

- **Main :** est le premier composant exécuté dans une application TinyOS.
- **MoM :** nom du module.
- **LedsC :** ce composant implémente une interface Led qui Permet l'utiliation des leds
- **RandomLFSR :** ce composant implémente une interface Random permet de générer des nombres aléatoires.
- **TimerC :** ce composant implémente une interface Timer qui permet d'attendre et de lancer un événement après le délai d'attente.
- **GenericComm :** implémente les interfaces **SendMsg** et **ReceiveMsg** permettant d'envoyer et recevoir des messages respectivement.
- **DemoSensorC :** ce composant est un Capteur de démonstration auquel on associe le nom Sensor implemente une interface ADC et StdControl.

2. ***Le Composant module :*** Implémente les commandes et les événements

```
#include"header.h"
module MoM{
    //interface fournie
    provides interface StdControl;
    //interfaces utilisées
    uses {interface ReceiveMsg as ReceiveMsgADV  ;
          interface SendMsg as SendMsgADV;
          interface ReceiveMsg as ReceiveMsgREQ  ;
          interface SendMsg as SendMsgREQ;
          interface ReceiveMsg as ReceiveMsgDATA  ;
          interface SendMsg as SendMsgDATA;
          interface ReceiveMsg as ReceiveSink  ;
          interface SendMsg as SendMsgSink;
          interface ReceiveMsg as ReceiveReveil  ;
          interface SendMsg as SendMsgReveil;

          interface Timer;
          interface Timer as TimerRELAY;
          interface Timer as TimerCapt;
          interface Timer as TimerREQ;
          interface Timer as TimerInit;

          interface Random;
          interface ADC;
          interface Leds;


       }}
```

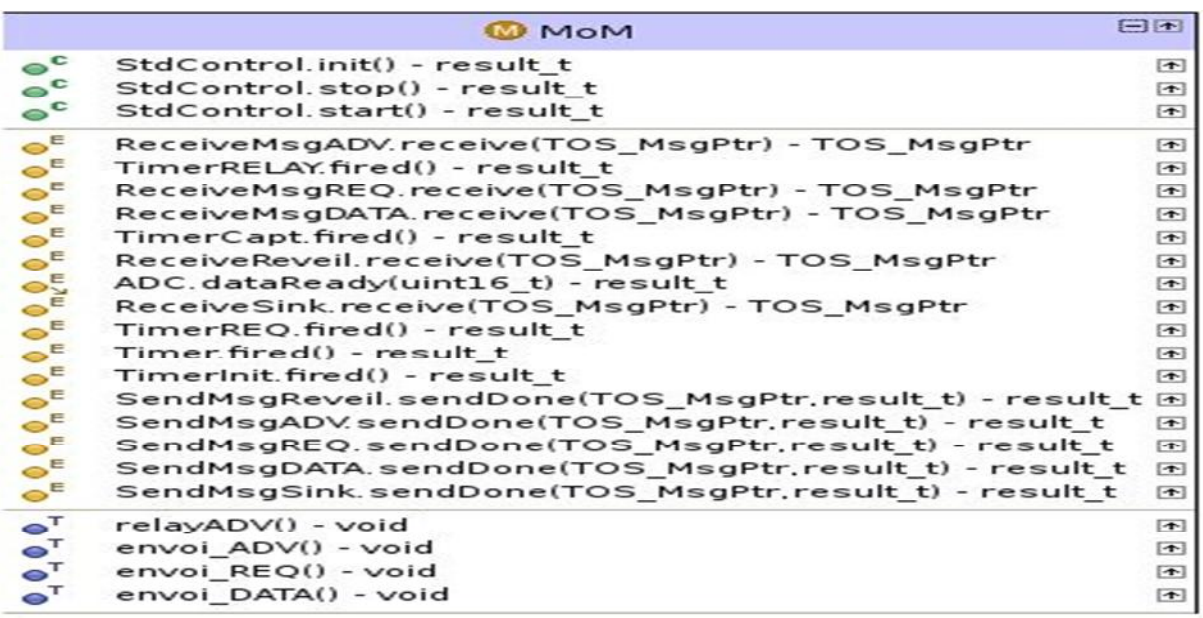


Figure III.2 : Représentation graphique du composant module

III.3 Environnement de simulation

Dans cette section, nous présentons les outils utilisés pour la simulation en commençant tout d'abord par présenté le système d'exploitation utilisé qui est TinyOS, conçu pour les systèmes embarqués en particulier les RCSF. Nous décrivons ensuite le langage de programmation adapté nesC avec lequel nous avons implémenté le code l'application. Et on fini cette section par la présentation des outils de simulation des RCSF.

III.3.1 Le système d'exploitation TinyOS

TinyOS est un système d'exploitation open-source conçu pour des réseaux de capteurs sans-fil. Il respecte une architecture basée sur une association de composants, réduisant la taille du code nécessaire à sa mise en place afin de respecter les contraintes de mémoires qu'observent les réseaux de capteurs [28]. 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. 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.

Le fonctionnement d'un système basé sur TinyOS s'appuie sur la gestion des évènements se produisant. Ainsi, l'activation de taches, 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 s'oppose au fonctionnement dit temporel où les actions du système sont gérées par une horloge donnée. [29]

- **Aperçus générale de TinyOS**

- Système d'exploitation pour réseaux de capteurs sans fil.
- Ensemble de composants logiciels qui peuvent être reliés ensemble en un seul exécutable sur une mote(*nœud capteur).*

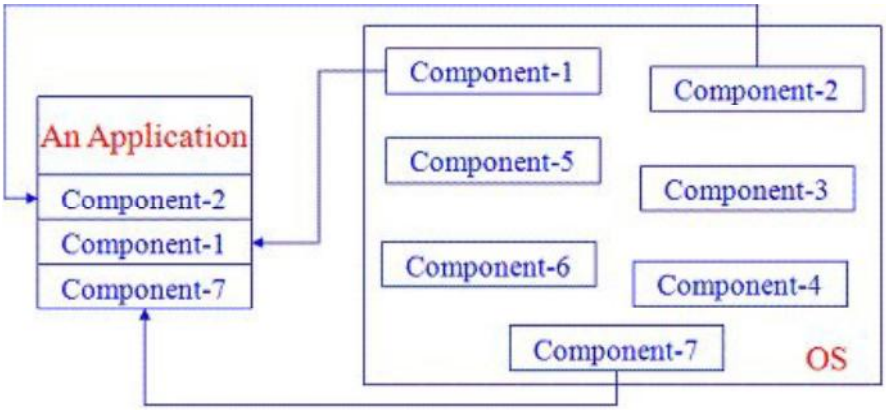


Figure III.3 : TinyOS : un ensemble de composants logiciels. [42]

♣ Modèle mémoire de TinyOS

❖ Allocation statique de la mémoire

- Pas de heap (malloc)
- Pas de pointeur sur fonction
- Pas d'allocation dynamique

❖ Variables globales

- Disponibles per-frame
- Conservation de la mémoire
- Utilisation de pointeurs

❖ Variables locales

- Sauvegardées sur la pile (stack)
- Déclarées dans une méthode

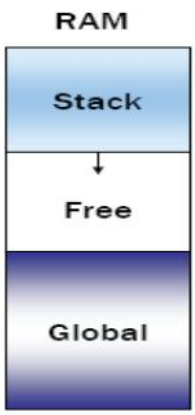


Figure III.4 : Organisation de la mémoire dans TinyOS. [31]

III.3.2 Le langage de programmation de TinyOS (nesC)

NesC(Network Embedded System C) est un langage de programmation orienté composants syntaxiquement proche du langage C. Il est conçu pour la réalisation des systèmes embarqués distribués, en particulier, les RCSF.

Grâce à NesC il est possible de créer une application à un assemblage de composant nécessaire à l'application. Chaque composant correspond à un élément matériel (LEDs, Timer, ADC ...) et peut être réutilisé dans différentes applications.

Un composant est constitué alors de trois parties essentielles : interfaces, modules et configurations [31].

1) Les interfaces permettent de spécifier des fonctions : des commandes ou des événements. Ces fonctions sont alors implémentées par le fournisseur ou l'utilisateur de l'interface, afin de distinguer les fonctions concernant les commandes de ceux concernant les événements, les fonctions sont précèdes de commande ou événement.

Exemple d'une interface

```
interface SendMsg
{
  command result_t send(uint16_t address, uint8_t length, TOS_MsgPtr msg);
  event result_t sendDone(TOS_MsgPtr msg, result_t success);
}
```

2) Les modules sont eux les éléments de base de la programmation. Ils permettent d'implémenter les composants et sont stockés dans des fichiers (.nc).

Exemple d'un module

```
module NomModuleM {
  provides {
    //liste des interfaces fournies,ex:
    interface NomInterfaceFourniel;
  }
  uses {
    //liste des interfaces requises,ex:
   interface NomInterfaceRequisel;
  }
}
implementation {
//declarations des variables,ex:
int stat;
//implementation des fonctions decrites par les interfaces fournies;
}
```

3) Les configurations permettent de d´écrire l'architecture. Une configuration est donc constituée de modules et/ou d'interfaces ainsi que de la description des liaisons entre ces composants.

Exemple d'une configuration

```
configuration Nomconfiguration {
}
implementation {
    //liste des modules et configurations utilises,ex:
  components Main, Module1,.....,ModuleN;
  //descriptifs des liaisons
  //interface fournie<-interface requise,ou
  //interface requise->interface fournies,ex
  Main.StdControl -> Module1.StdControl;

}
```

III.3.3Les outils de simulation

TinyOs offre différents outils de Simulation tel que : TOSSIM, PowerTOSSIM et TinyViz

TOSSIM

TOSSIM permet de simuler le comportement d'un capteur au sein d'un réseau de capteurs, il est un simulateur discret basé sur la programmation par événement, de même qu'il est conçu pour simuler les réseaux de capteurs qui utilisent la plateforme TinyOS. Le principal but de TOSSIM est de créer une simulation très proche de ce qui ce passe dans ces réseaux dans le monde réel.

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. TOSSIM fourni deux modèles de radios pour la communication : Le modèle par défaut « simple » où les paquets sont transmis dans le réseau sans aucune erreur et ils sont reçus par chaque nœud. Avec ce modèle il est ainsi possible que deux nœuds différents peuvent envoyer un paquet en même temps avec la conséquence que ces deux paquets seront alors détruit à cause du chevauchement des signaux. Le deuxième modèle est le modèle « lossy », dans ce modèle les nœuds sont placés dans un graphe direct formé d'un couple (a, b) ce qui signifie qu'un paquet envoyé par le nœud a peut être reçu par le nœud b.

TOSSIM est équipé aussi d'un simulateur graphique TinyViz. Cette application est équipée de plusieurs API plugins qui permet d'ajouter plusieurs fonctions au simulateur comme par

exemple contrôler les entrées radio ou bien suivre la dépense d'énergie en utilisant un autre simulateur qui s'appelle PowerTOSSIM **.**

PowerTOSSIM

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

TinyViz

TinyViz est une application graphique qui donne un aperçu du réseau de capteurs à tout instant, ainsi que les divers messages qu'ils émettent. Il permet de déterminer un délai entre chaque itération afin de permettre une analyse pas à pas du bon déroulement des actions, il possède aussi des options afin de pouvoir simuler la consommation d'énergie [32].

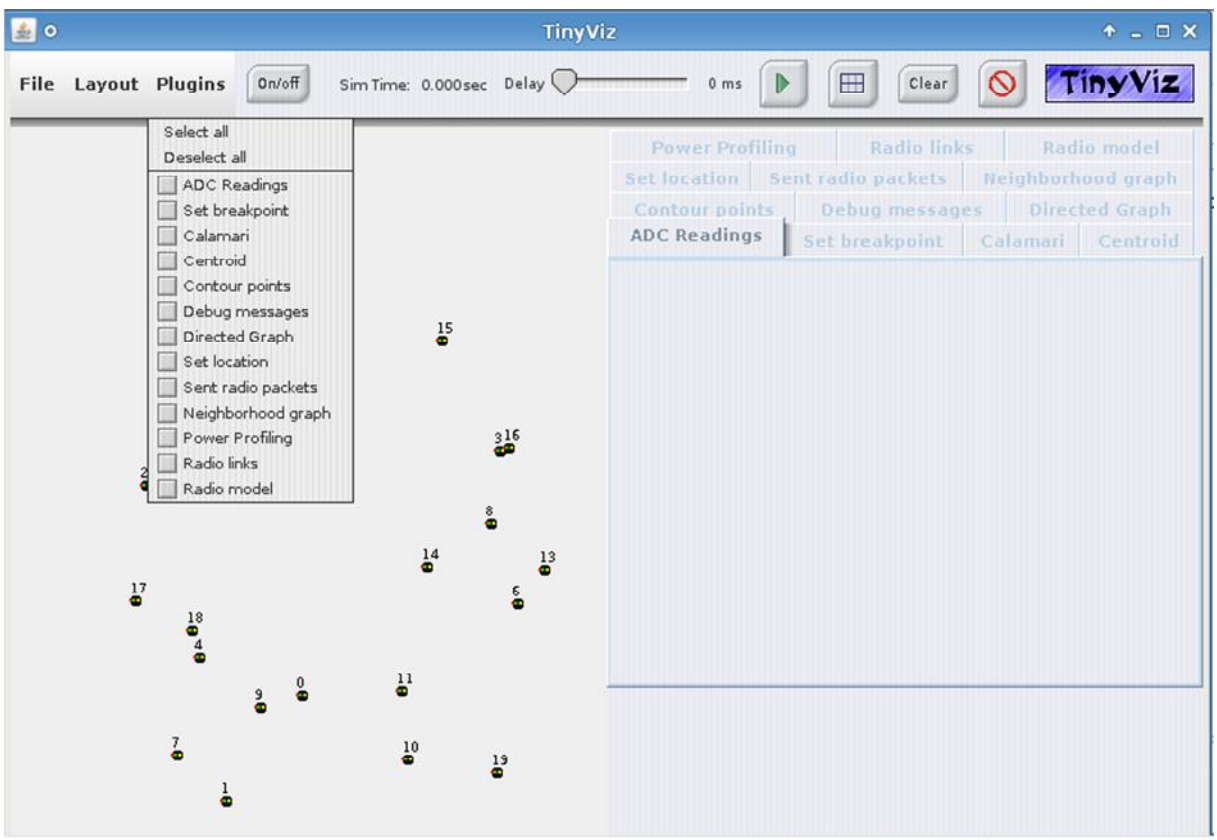


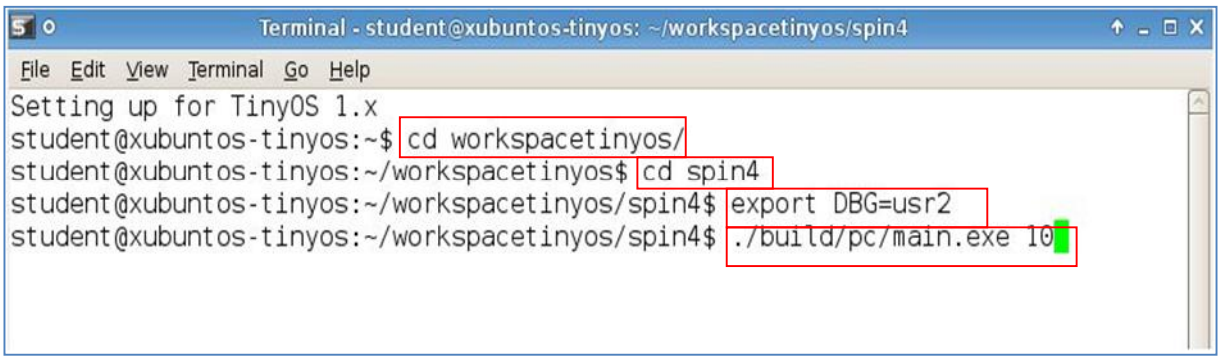
Figure III.5 : Environnement de simulation TinyViz.

- **On/Off** : met en marche ou éteint un capteur.
- **Le délai du timer :** permet de sélectionner la durée au bout de laquelle se déclenche le timer.
- **Le bouton « Play »** : permet de lancer la simulation ou de la mettre en pause.
- **Le bouton de fermeture** : arrête la simulation et ferme la fenêtre.

- **Le bouton « clear »** : il efface tous les messages qui avaient été affichés lors de la simulation.

III.4 Simulation

Apres compilation du projet, on ouvre le terminal pour l'exécuter. Avant d'exécuter le projet, il faut taper les commandes suivantes :



Apres avoir tapé les commandes précédentes on aura cette simulation :

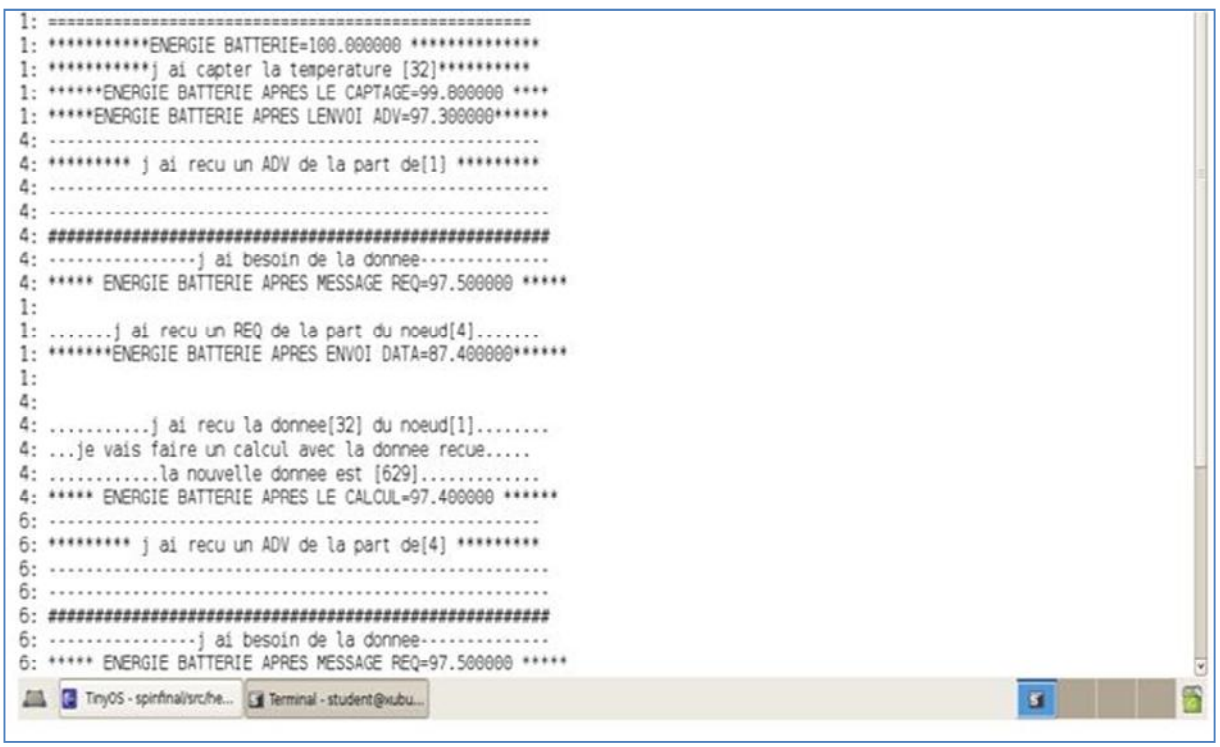


Figure III.6 : Simulation de SPIN

Avant de passer à la simulation graphique il faut d'abord générer une topologie avec les commandes suivantes (Figure III.7) :

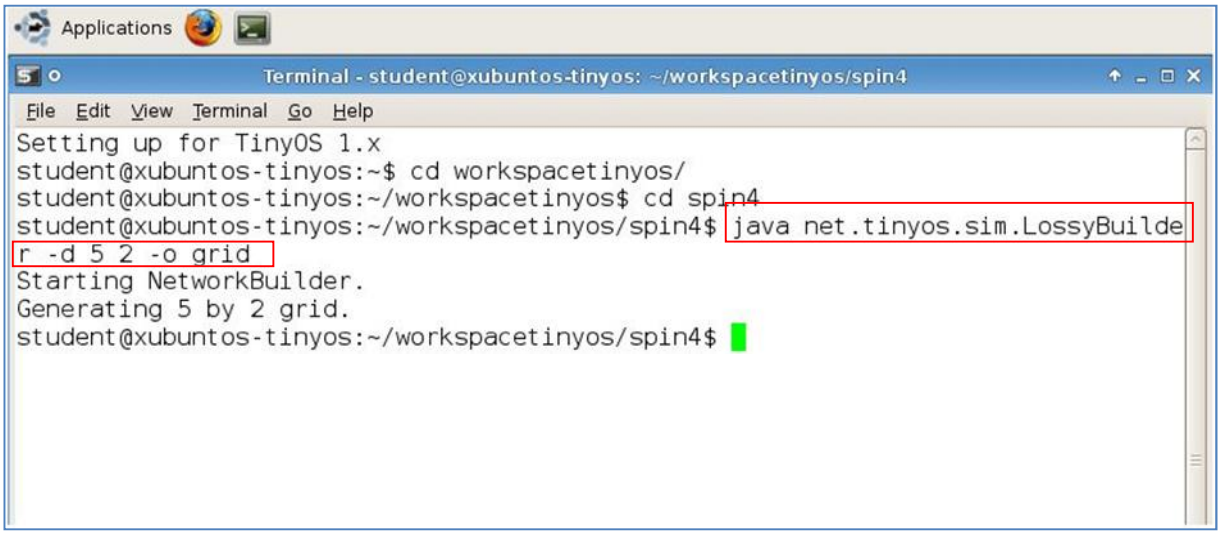


Figure III.7 : Générer la topologie du réseau.

III.4.1 Simulation graphique

- Le lancement de simulateur graphique TinyViz

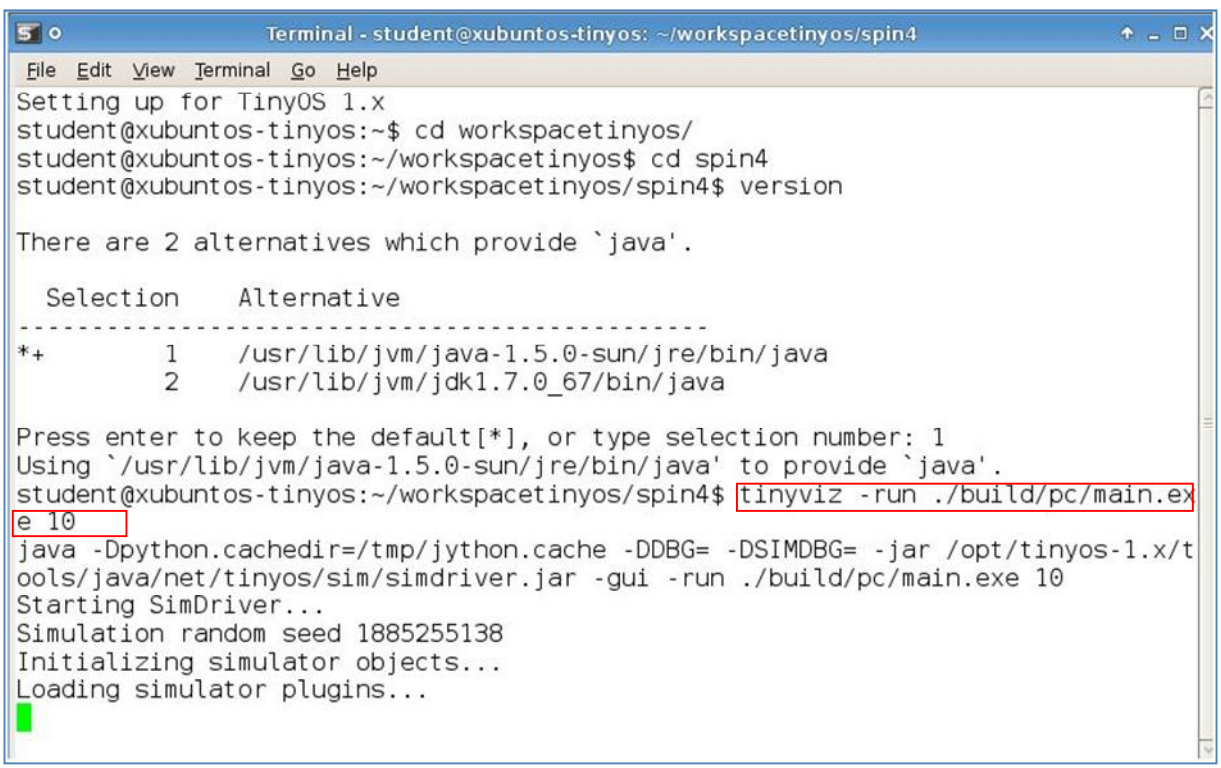


Figure III.8 : Lancement de l'interface graphique TinyViz.

- Téléchargement de la topologie pour les nœuds capteurs

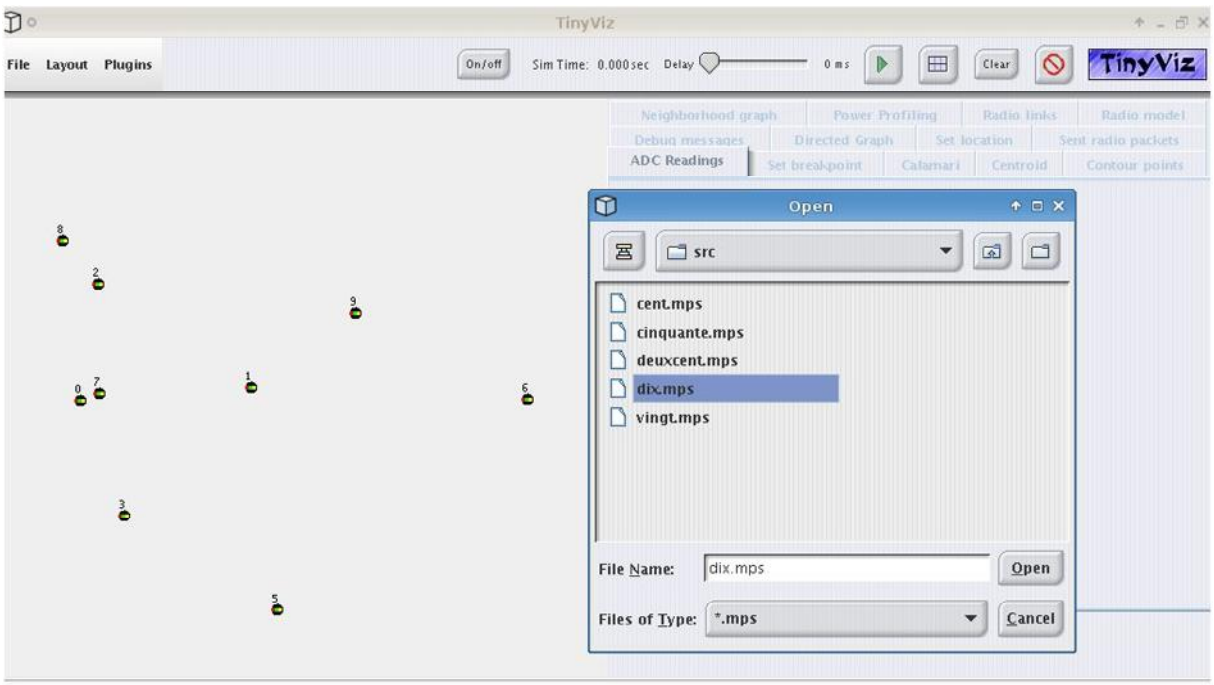


Figure III.9 : télécharger une topologie pour les nœuds capteurs.

- L'activation des plugins nécessaires

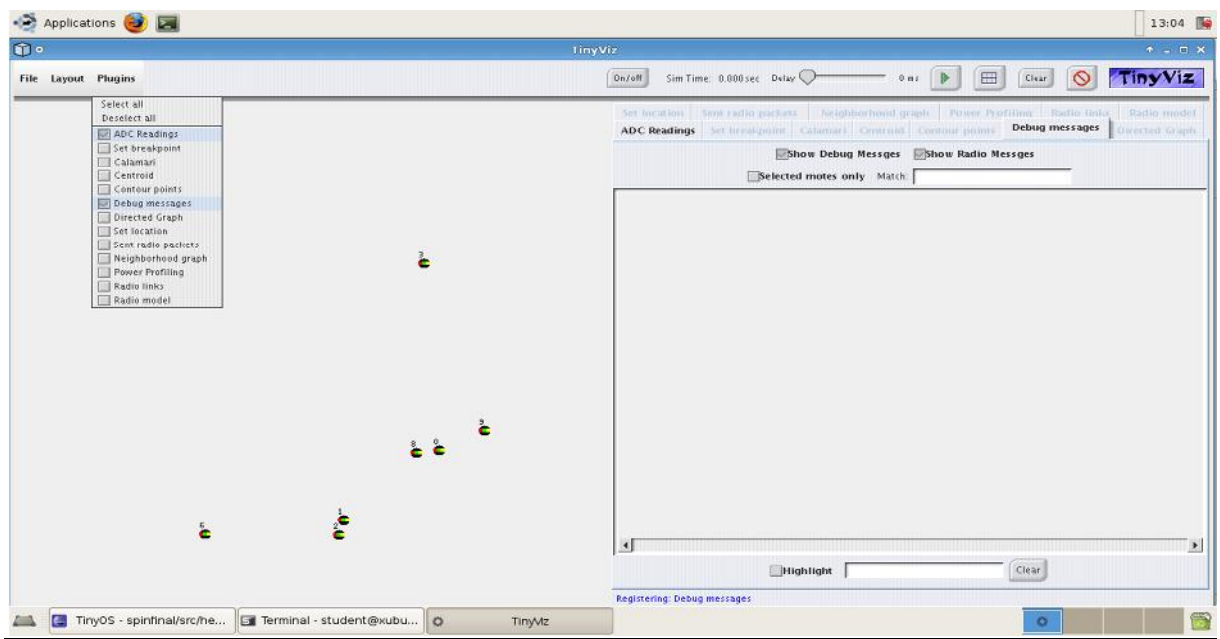


Figure III.10 : Activation des plugins

- Le lancement de la simulation par le bouton « Play »

1) Le captage et la diffusion du message ADV par le nœud origine

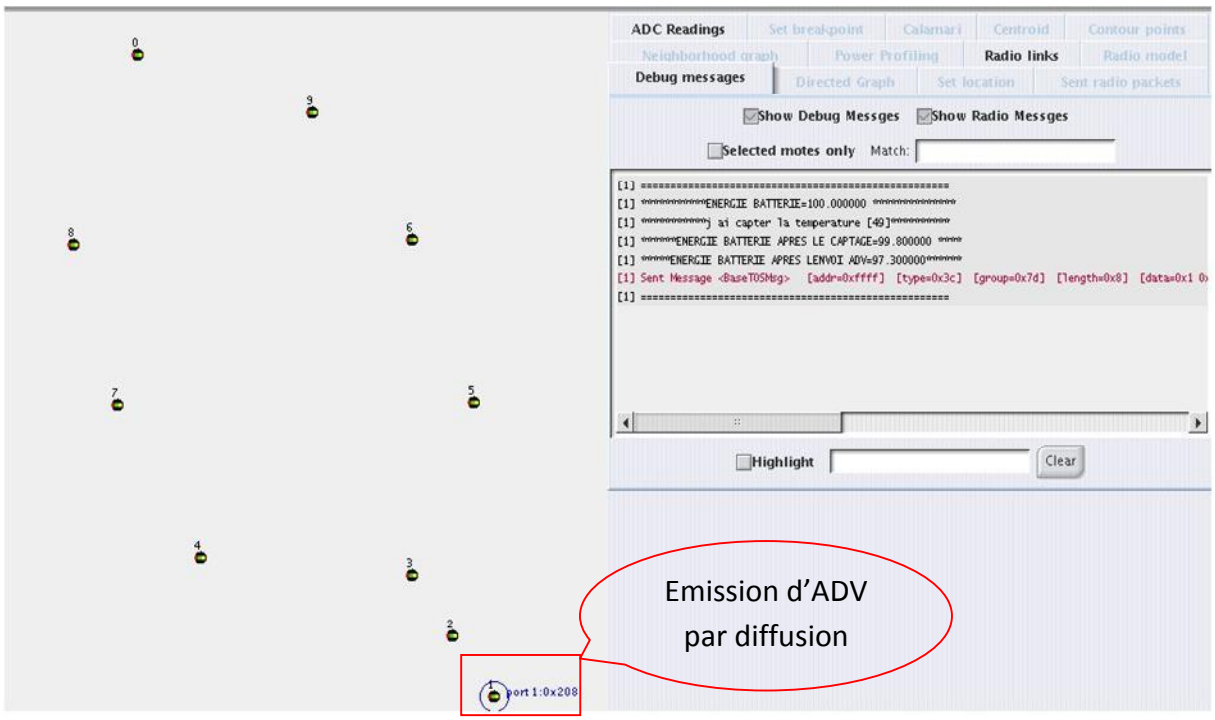


Figure III.11 : Le captage et la diffusion d'ADV par le nœud origine.

Le nœud origine (égale à 1) capte une température qui représente la donnée puis, il émet un message ADV (métadonnée) par diffusion (broadcast).

2) L'émission du message REQ par le voisin au nœud origine

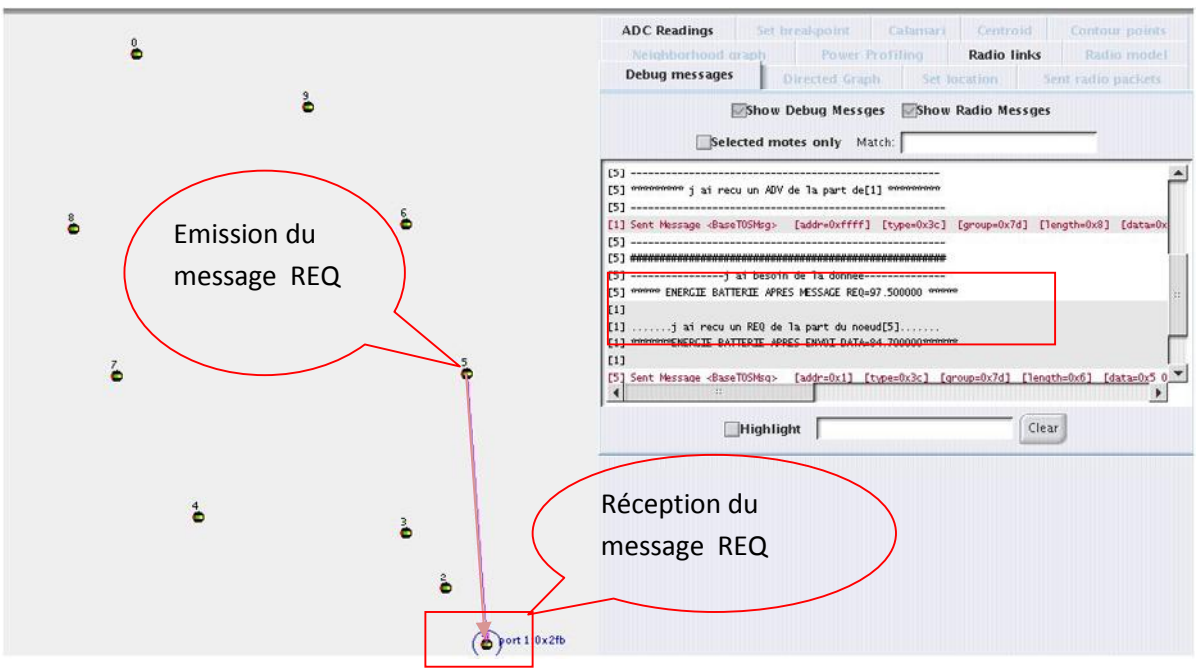


Figure III.12: Emission du message REQ par le voisin d'origine.

Après réception du nœud voisin (égale à 4) du message ADV, il consulte sa base d'intérêt. S'il trouve qu'il est intéressé par cette information donc il émet un message REQ.

3) L'émission du message DATA au voisin après réception de message REQ

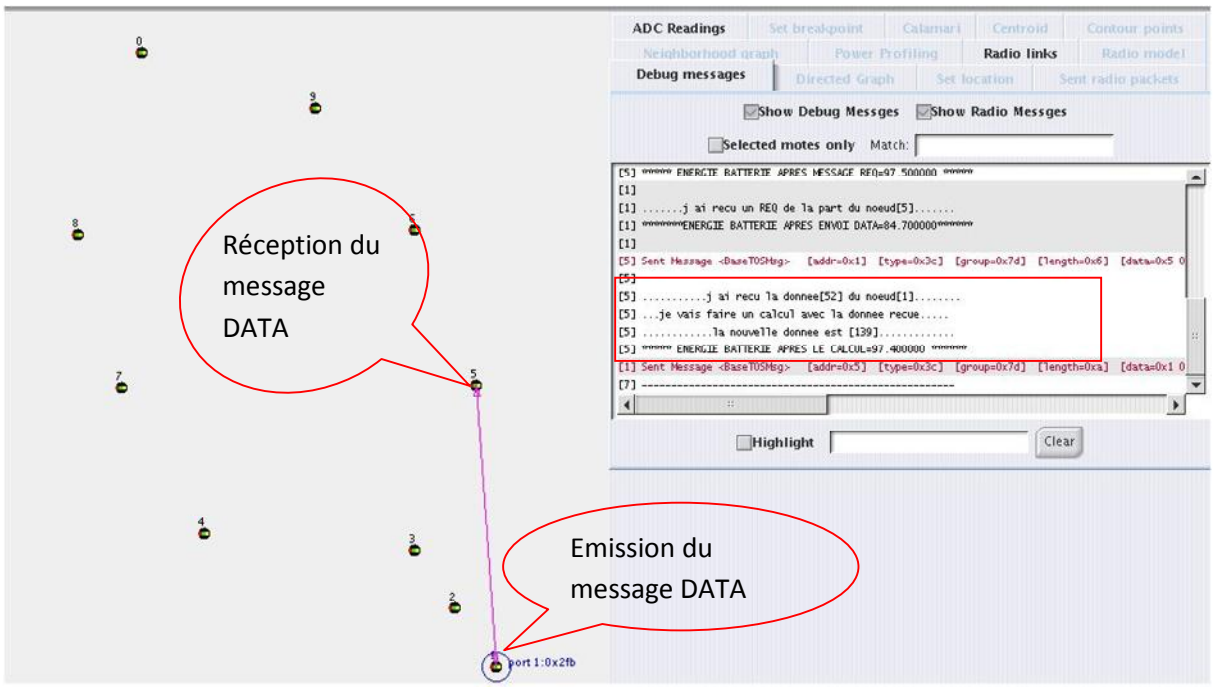


Figure III.13 : Emission du message DATA par origine au nœud voisin après réception du message REQ.

Le nœud origine(1) envoi un message DATA après réception du message REQ du nœud voisin(4).

4) Le routage de la donnée du nœud origine vers la station de base.

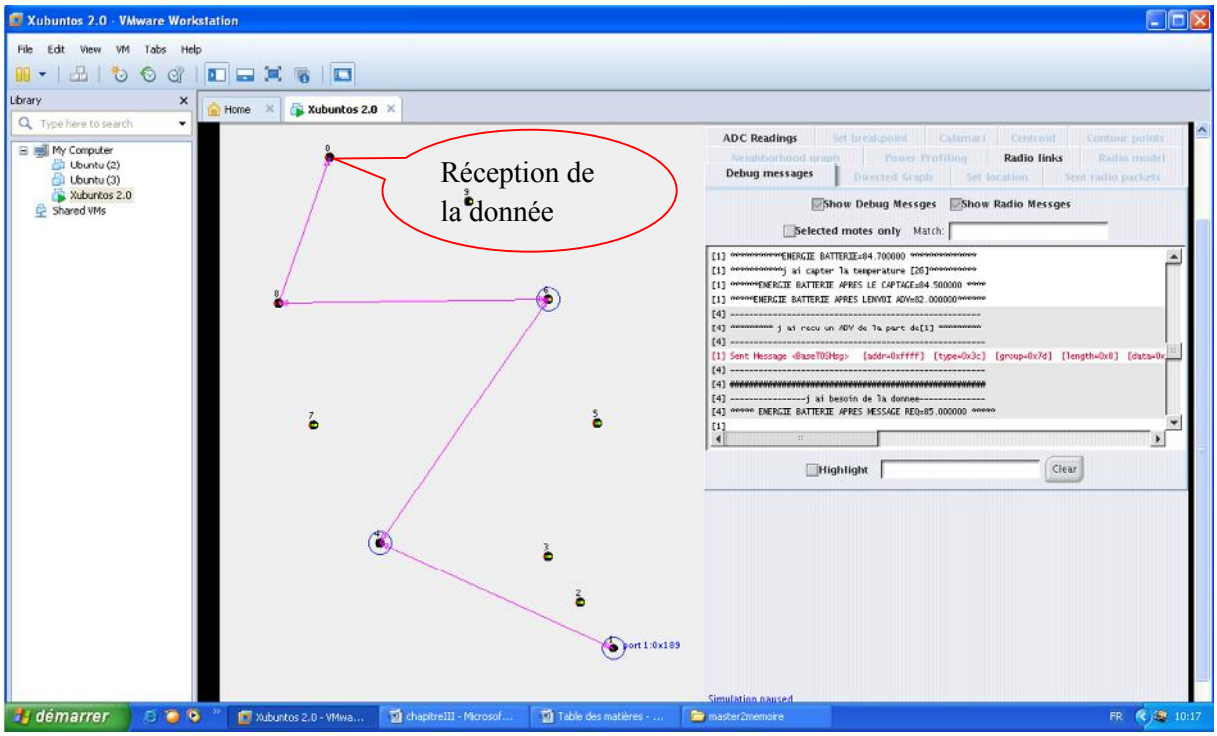


Figure III.14 : Le routage de la donnée du nœud origine vers la station de base.

Le routage de la donnée du nœud origine vers la station de base ce fait en passant par les figures (figure III.11, figure III.12, figure III.13) respectivement ainsi le nœud recevant la donnée va faire le même processus jusqu'à la réception par la station de base de la donnée.

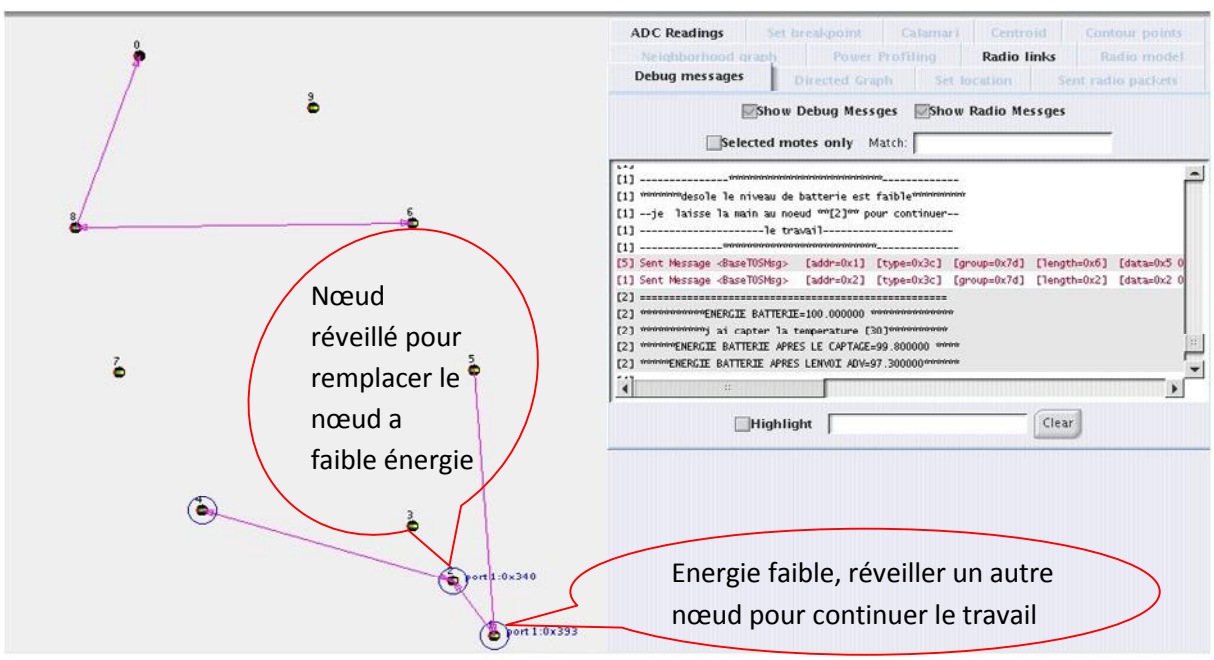


Figure III.15 : Remplacement d'origine par un autre nœud .

Lorsque le nœud origine(1) s'aperçoit que son énergie est descendue sous un certain seuil, il émet un message réveil à un autre nœud pour prendre son rôle.

III.4.1.1 Simulation graphique de la consommation d'énergie

Le plugin **Power Profiling** de TinyViz vous permet de voir l'état de la consommation de l'énergie de l'application en cours d'exécution. Tapez dans un terminal, les commandes ci-dessous :

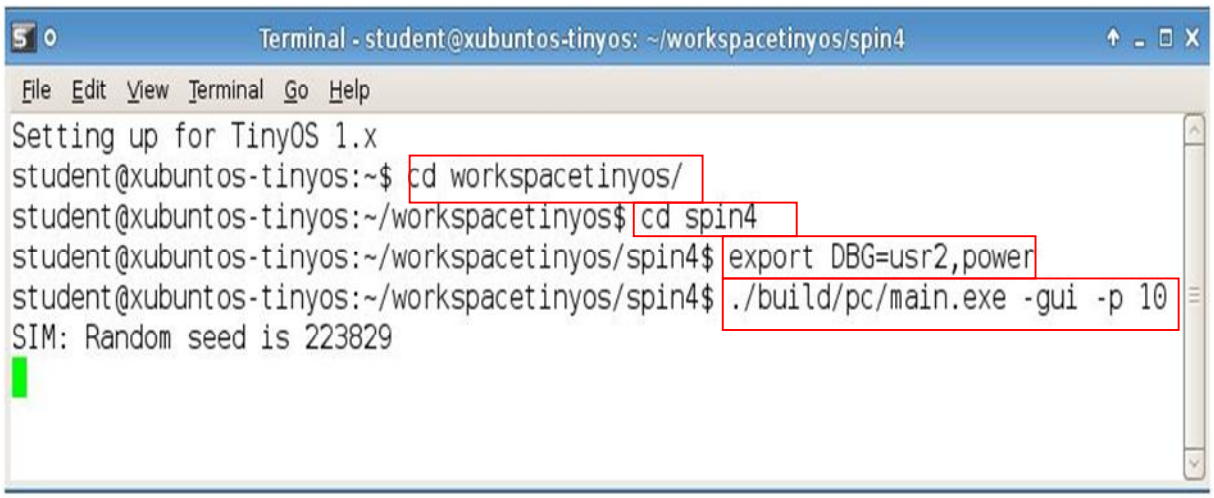


Figure III.16: Les commandes utilisées pour la simulation de la consommation d'énergie.

Puis ouvrir un autre terminal et tapez :tinyviz

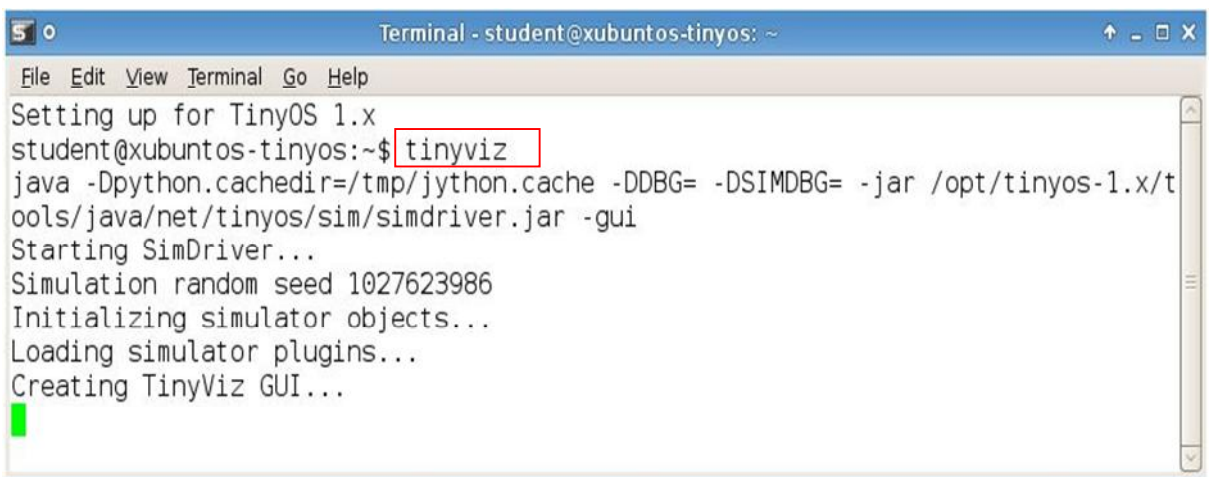


Figure III.17 : Ouverture d'un autre terminal pour lancer TinyViz.

L'interface graphique TinyViz s'ouvre et on coche le plugin **Power Profiling**, et les autres plugins nécessaires (ADCreading,Debugmessage,Radiolink)

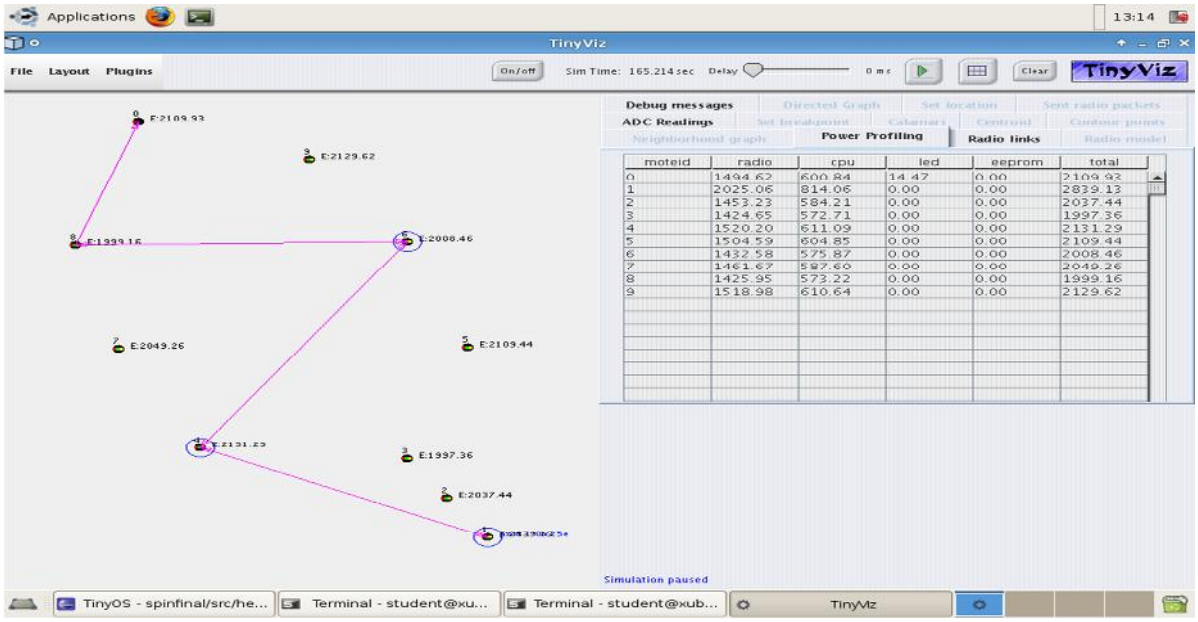


Figure III.18 : Simulation de la consommation d'énergie SPIN.

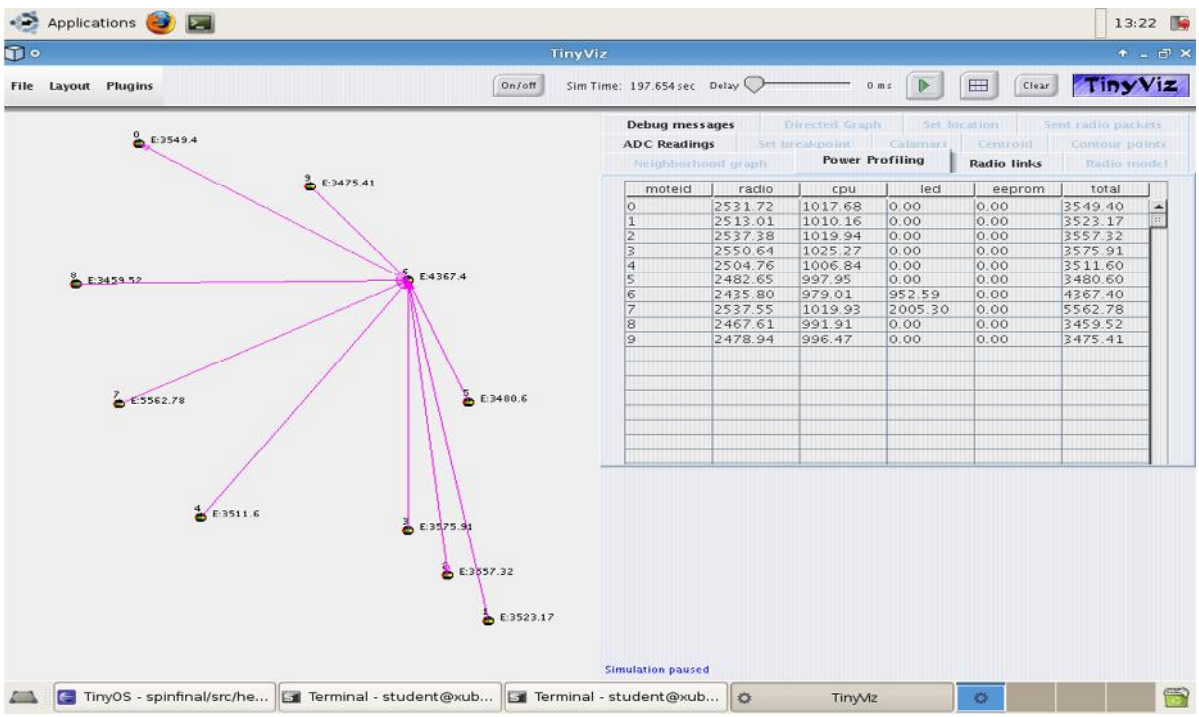


Figure III.19 : Simulation de la consommation d'énergie LEACH.

III.4.1.2 Comparaison de consommation d'énergie de LEACH et SPIN

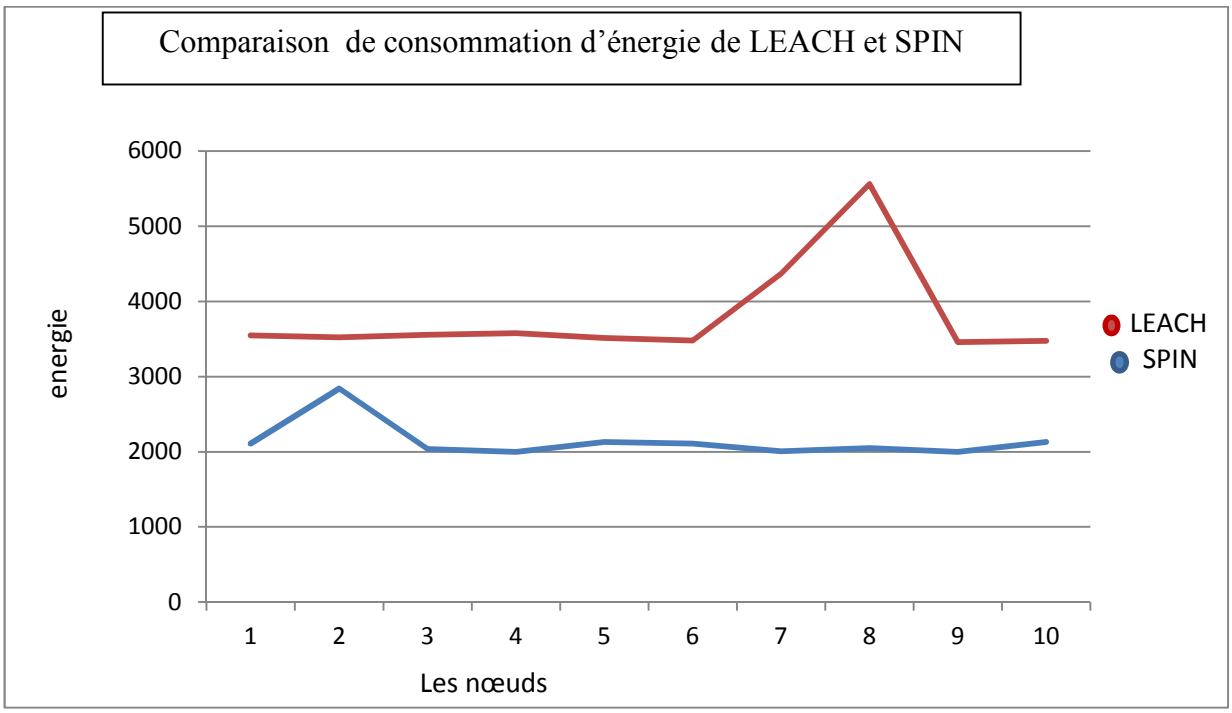


Figure III.20 : Courbe d'évaluation de protocole SPIN et LEACH

Les figures **(**figure III.20, figure III.21 montre que le protocole SPIN consomme moins d'énergie par rapport a LEACH à cause de la négociation qui évite la retransmission d'un même message.

III.5 Conclusion

La simulation occupe une place très importante comme outil d'aide à la conception, l'analyse, la formation, la planification, l'évaluation des performances ou de la sûreté de fonctionnement de tout système. Dans le domaine des réseaux de communication, la simulation est l'outil privilégié pour réaliser des études d'évaluation des performances. Ceci est dû à la complexité toujours croissante de ce type de système ainsi qu'à la disponibilité –aussi croissante– d'outils de simulation de bonne qualité orientés réseaux.

Dans ce chapitre nous avons implémentés le protocole SPIN et nous avons illustré les différents résultats obtenus. Nous avons essayé d'implémenter ce protocole avec moins d'énergie consommée afin d'augmenter la durée de vie du réseau.

Conclusion Générale

Conclusion Générale

L'étude réalisée durant ce projet nous a permis de découvrir le monde des réseaux de capteurs sans fil qui sont composés d'un très grand nombre de dispositifs de communication ultra petits, autonomes avec des ressources de calcul et d'énergie limitées. Ils sont actuellement considérés comme l'une des technologies qui bouleverse notre façon de vivre, grâce à leur utilisation dans différents domaines d'application.

Cependant, les réseaux de capteurs sans fil rencontrent plusieurs problèmes qui affectent leur bon fonctionnement dû à leurs caractéristiques ; tels que le type de communication, les environnements hostiles où sont déployés les capteurs…etc.

Une des principales problématiques lors du développement de réseaux de capteurs sans fil est le routage ; L'absence d'infrastructure et les faibles capacités de calculs et de batteries rendent la fonction de routage plus critique.

L'objectif de notre travail était d'étudier et de réaliser un algorithme de routage plat pour les RCSF cas : « Sensor Protocol for Information via Negotiation »SPIN, ensuite cet algorithme sera comparé avec un autre algorithme de routage hiérarchique qui est LEACH en termes d'énergie.

Ce travail peut ce résumer en trois parties :

- Dans la première partie, nous avons fait une étude théorique sur le routage dans les réseaux de capteurs sans fil. Nous avons mis l'accent sur les différents domaines d'application et les différents protocoles de routage.
- Dans la deuxième partie, nous avons étudiés le protocole de routage SPIN
- Dans la troisième partie, nous avons simulés le protocole SPIN et LEACH par rapport à leurs consommations énergétiques.

Les bénéfices de notre travail :

Le travail que nous avons accompli nous a permis :

- De nous initié aux différentes étapes à suivre pour l'implémentation de « SPIN »
- D'acquérir de nouvelles connaissances sur le langage nesC et le système d'exploitation TinyOS.

Perceptives

- Afin d'être adaptés à un environnement réel, nos algorithmes peuvent être toujours améliorées.
- Réalisation dans des conditions réelles de manière à comparer les performances réelles par rapport à celles effectuées par des simulations.

Références

[1]K.Beydoun. « Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs »Thèse de doctorat, Spécialité : Informatique, l'u.f.r des sciences et techniques de l'université de Franche-Comté, 2009.

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci , "A Survey on Sensor Networks". IEEE Communications Magazine, August 2002.

[3] B. Jung and G. Sukhatme, "Multi-target tracking using a mobile sensor network". In Proceedings of the IEEE International Conference on Robotics and Automation, May 2002.

[4] C. Chong and S. Kumar, "Sensor networks: Evolution, opportunities, and challenges" Proceedings of IEEE, August 2003.

[5] E. M. Petriu, N. D. Georganas, D. C. Petriu, D. Makrakis, and V. Z. Groza, "Sensor-based information appliances". IEEE Information and Measurement Magazine, December 2000.

[6] B. Sibbald, "Use computerized systems to cut adverse drug events". Canadian Medical Association Journal, June 2001.

[7] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks". MobiSys 2004.

[8] ALERT Systems. http://www.alertsystems.org/.

[9] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro, and T. Porcheron, "Monitoring behavior in home using a smart fall sensor". IEEE-EMBS Special Topic Conference on Microtechnologies in Medicine and Biology, October 2000.

[10] J. Lester Hill, "System Architecture for Wireless Sensor Networks". University of California, Berkeley, 2003.

[11] V. Handziski, A. Kopke, H. Karl, and A. Wolisz, "A common wireless sensor network architecture". Technische Universität Berlin, July 2003, pp.10 17.

[12] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, "Fault Tolerance in Wireless Ad hoc Sensor Networks". Proceedings of IEEE Sensors 2002, June 2002.

[13] L. Paradis and Q. Han, "A Survey of Fault Management in Wireless Sensor Networks". Plenum Press New York, NY, USA, 2007.

[14] F. Nekoogar, F. Dowla, and A. Spiridon, "Self organization of wireless sensor networks using ultra-wideband radios". Atlanta, GA, United States, September 2004.

[15] C.Y. Chong and S.P. Kumar, "Sensor networks: Evolution, opportunities, and challenges". Proceedings of the IEEE, vol. 91, n.8, 2003, pp. 1247-1256.

Références

[16] T. Zhao, W. D. Cai, and Y. J. Li, “A Sensor Network Topology Inference Algorithm,omputational Intelligence and Security”. 2007 International Conference on. January 2008.

[17] V. Handziski, J. Polastrey, J. H. Hauer, C. Sharpy, A. Wolisz, and D. Cullery, “Flexible Hardware Abstraction for Wireless Sensor Networks”. In Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN '05),February 2005.

[18] Jaap C. Haartsen, “The Bluetooth radio system”. IEEE Personal Communications Magazine, February 2000, pp. 28-36.

[19] Practel, Inc. ZigBee, “Technology for Wireless Sensor Networks”. April 2006.

[20] I. Teixeira, J. F. de Rezende, A. de Castro, and A. C. P. Pedroza, “Wireless Sensor Network: Improving the Network Energy Consumption”. in XXI Symposium Brazilian Telecommunications, SBT’04, Belem, Brazil, September 2004.

[21] E. Souto, R. Gomes, D. Sadok and J. Kelner, “Sampling Energy Consumption in Wireless Sensor Networks”. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing -Vol 1 (SUTC'06), June 2006.

[22] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, “Energy-aware wireless microsensor networks”. IEEE Signal Processing Magazine, Vol. 19, No. 2, March 2002, pp.40-50.

[23] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol for wireless sensor networks ”. In the Proceeding of the Hawaii International Conference System Sciences, Hawaii, January 2000.

[24] S. Ziane and A. Mellouk, “A swarm intelligent scheme for routing in mobile ad networks”. Systems Communications, IEEE, Aug 2005.

[25] Paolo Santi, “Topology Control in Wireless Ad Hoc and Sensor Networks”. Hardcover, july 2005.

[26] E.DHIB « Routage avec QoS temps réel dans les réseaux de Capteurs »Ingénieur en Télécommunications option : Ingénierie des réseaux, école supérieure de communication de Tunis, 2006/2007.

[27] C.Intanagonwiwat, R.Govindan, and D.Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", Proceeding of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Boston, 2000.

[28] système d’exploitation des capteurs TinyOS

Source : www.labstic.univ-tlemcen.dz PDF.

Références

[29]Maarouf Samia et Ouadah Souhila « implémentation et évaluation des schémas de routage sur une plateforme réelles de réseaux de capteurs sans fils Promotion 2013/2014. Source :www.dspace.univ-tlemcen.dz

[31] *TinyOS/nesC Overview*. In a Smart Dust Training Seminar CD, SanJose, February 9-10, 2005.

[32]FaresA ddefatah « Développement d'une bibliothèque de capteur »Rapport 2008 Source :www.lirmm.fr/.

[33] CHALLAL, Y. Réseaux de capteurs sans fils.2008, 103 p, support-SIT60.PDF.

[34] Akyildiz I. F. Su W., Sankarasubramaniam Y. et Cayirci E. Wireless sensor networks: a survey [Revue] // ScienceDirect. - Atlanta, USA : Elsevier Science, 15 Mars 2002. - 4 : Vol. 38. - pp. 393-422.

[35] M. Ali and S. K. Ravula, "Real-time support and energy efficiency in wireless sensor networks". Technical report, IDE0805, January 2008.

[36] .C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", in the *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00),* Boston, MA, August 2000.

[37] Jamal N. Al-Karaki Ahmed E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey", Dept. of Electrical and Computer Engineering Iowa State University, Ames, Iowa.

[38] K. Akkaya, and M. Younis, "A Survey on Routing Protocols for Wireless Sensor//data centric Networks". Journal of Ad Hoc Networks, Vol. 3, No. 3, May 2005, pp. 325-349.

[39]W. Heizelman, J. Kulik, and H. Balakrishnan. "Adaptive protocols for information dissemination in wireless sensor networks". *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 174-185*, 1999

[40] http://en.wikipedia.org/wiki/Network_topology.

[41] Marcos Augusto Menezes Vieira "BEAN: Uma Plataforma Computacional para Rede de Sensores Sem Fio" Belo Horizonte Avril 2004.

[42] Challal Y. TinyOS: Système d'exploitation pour réseaux de capteurs sans fils novembre 2008. support-SIT60.PDF.

Références

[43] Cours1 Yacine CHALLAL,Hatem BETTAHAR ,Abdelmadjid BOUABDALLAH HeuDiaSyc UMR CNRS 6599 Université de Technologie de Compiègne, France. Les Réseaux de capteurs (WSN: Wireless Sensor Networks). Support-TI60.PDF