L’agrégation de données dans un réseau de capteurs sans fil

LAHLOUH SIDIA

OMOURI SARAH

2010/2011

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE
LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU
FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT D'INFORMATIQUE

Projet Fin d'Etude

Mémoire présenté pour l'obtention du diplôme d'
MASTER 2 EN INFORMATIQUE
Option : Conduite de projet informatique(CPI)

Thème :

L'agrégation de données dans un réseau de capteurs sans fil

Réalisé par :
Mlle. LAHLOUH Sidia
Mlle. OMOURI Sarah

Encadré par :
M. DEMRI

Promotion : 2010/2011

Remerciements

Grâce à Dieu vers lequel vont toutes les louanges, ce travail s’est accompli.
Grâce à Dieu, nous avons l’honneur d’inscrire ici un immense remerciement à nos parents pour leur contribution, leur soutien et leur patience.

En préambule à ce mémoire, nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu’à la réussite de cette formidable année universitaire.

Nous tenons à remercier sincèrement Monsieur DEMRI, qui, en tant que Directeur de mémoire, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours soutenue et encouragée au cours de la réalisation de ce mémoire.
Merci à tous et à toutes.

Dédicaces

Je dédie ce modeste travail :

A ma très chère mère et mon adorable grande mère
vous êtes l'exemple de dévouement qui n'a pas cessé de m'encourager et de prier pour moi. Puisse Dieu, le tout puissant, vous préserve et vous accorde santé, longue vie et bonheur.

A mon très cher père (il n'est pas facile de faire de son enfant un enfant heureux)
Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail est le fruit de tous les sacrifices que tu as consentis pour mon éducation et ma formation.

A mon cher frère Amine, et ma chère sœur Massissilia
Vous vous êtes dépensés pour moi sans conter. En reconnaissances de tous les sacrifices consentis par tous et chacun pour me permettre d'atteindre cette étape de ma vie.

A mes oncles, tantes, cousin et cousines affectueuses reconnaissances.

A ma chère binôme et amie Sidia
Je vous remercie de votre patience vous m'avez toujours aidée à avancer vous êtes une grande amie si gentille, merci d'être toujours près de moi, amie avec laquelle je souris. Je vous souhaite beaucoup de réussite et tout le bonheur du monde.

A mes meilleurs amis Brahim, Jiji
Je vous remercie d'avoir été la pour moi.

À toutes les personnes qui connaissent « ***Sarah*** » de prés ou de loin, Seulement pour leur existence.

Sarah

Dédicaces

Je dédie ce modeste travail :

A ma très chère et regrettée grande mère

A *la mémoire de la grande dame qui a tant sacrifié pour nous . J'espère que, du monde qui est sien maintenant, elle apprécie cet humble geste comme preuve de reconnaissance .Que ton âme repose en paix*

A mon très cher père

A*ucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous. Rien au monde ne vaut les sacrifices fournis pour mon éducation et ta présence en toute circonstance m'a maintes fois rappelé le sens de la responsabilité, que dieu te protège mon cher père.*

A ma très chère mère

A*ffable, honorable, aimable : Tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi. Je te dédie ce travail en témoignage de mon profond amour. Puisse Dieu, le tout puissant, te préserver et t'accorder santé, longue vie et bonheur.*

A mes chers frères , et mes chères sœurs

L*es mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je vous porte. Mes anges gardiens et mes fidèles compagnons dans les moments les plus délicats de cette vie mystérieuse. Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.*

A mon cher oncle Achour et sa famille

V*ous avez toujours été présents pour les bons conseils. Votre affection et votre soutien m'ont été d'un grand secours au long de ma vie. Veuillez trouver dans ce modeste travail ma reconnaissance pour tous vos efforts.*

A mes chers amis et spécialement toi ma chère Sarah

E*n témoignage de l'amitié qui nous uni et des souvenirs de tous les moments que nous avons passé ensemble, je vous dédie ce travail et je vous souhaite une vie pleine de santé et de Bonheur.*

Sidia

Résumé

Les progrès récents dans les communications sans fil et le domaine de l'électronique ont permis le développement des micro-capteurs, moins coûteux et multifonctionnels. Ces caractéristiques ont permis de se projeter dans la naissance des réseaux de capteurs sans fil (RCSF), et de favoriser leur utilisation dans une multitude d'applications. Celles-ci nécessitent souvent un déploiement dans des environnements hostiles, où les nœuds sont exposé a une contrainte d'énergie.

Les applications de réseau de capteur sans fil n'exigent pas d'habitude la connaissance de chaque mesure individuelle d'un capteur dans les puits; plutôt qu'un ensemble complet des données contrôlées par des nœuds capteur différents pourrait être suffisant. Le traitement avec des données agrégées est potentiellement fortement commode dans la perspective de réseau parce qu'il mène à une réduction de la consommation d'énergie complète, même si le traitement nécessaire implique la consommation de ressource.

La raison est que par l'agrégation, la surcharge des paquets inutiles sont évités, la perte d'énergie ainsi que la largeur de la bande passante sont réduites. De plus, durant l'agrégation de données dans un réseau quelques pré-élaboration des données peut être faite, plusieurs applications de réseau de capteur sans fil peuvent bénéficier d'avoir ces données pré élaborées et peut être facilité la manipulation et le traitement des mesures de capteur.

Ainsi, l'agrégation de données est une approche prometteuse pour améliorer la performance de réseaux de capteur et a récemment attiré l'intérêt des chercheurs. Dans cette optique, nous fournissons une vue détaillé des approches principales de l'agrégation de données, nous nous sommes intéressés à l'implémentation de l'un des protocoles qui intègre cette technique intituler LEACH, pour atteindre cet objectif nous avons étudié son fonctionnement, nous avons décrit la plateforme TinyOS et ses différents avantages pour ces réseaux ainsi nous avons implémenté se protocole sur TinyOS.

Mots clés: Réseaux de capteurs sans fil, agrégation de données dans un réseau, LEACH, TinyOS

Abstract

Recent advances in wireless communications and electronics have enabled the development of tiny, low-cost, and multifunctional sensor nodes. These characteristics have contributed to the design of Wireless Sensor Networks (WSNs) and promoted their use in a multitude of applications. These applications often require deployment in hostile environments, where nodes are exposed to constraint of energy.

Wireless sensor network applications usually do not require knowledge of each individual sensor measurement at the sinks; rather, an overall aggregate of the data monitored by different sensor nodes could be sufficient. Dealing with aggregated data is potentially highly convenient in the network perspective because it leads to a reduction in the overall energy consumption, even if the needed processing involves resource consumption.

The reason is that through aggregation the unnecessary packets overhead is avoided and the waste of energy and bandwidth resources are reduced. Moreover, since during in-network data aggregation some pre-elaboration of the data can be made, several wireless sensor network applications may benefit from having this pre-elaborated data and may be facilitated in handling and processing the sensor measurements.

Thus, data aggregation is a promising approach for improving the performance of sensor networks and has recently attracted the interest of researchers. In this work, we provide a detailed view of the main approaches at data aggregation, We were interested in the implementation of one of the protocols which integrates this technique to entitle LEACH, to reach this objective we studied its functioning, we have described the TinyOS platform and its various advantages for these networks and we implemented this protocol on TinyOS.

Key words: Wireless Sensor Networks, in-network data aggregation, LEACH, TinyOS.

Table des matières

LISTE DES FIGURES XI

INTRODUCTION GENERALE.....1

I. CHAPITRE I.....3

Les réseaux de capteurs 3

I.1 Introduction 3

I.2 Le capteur et le capteur intelligent 4

I.2.1. Architecture d'un capteur 4

I.2.2. Caractéristiques principales d'un capteur 7

I.3 Les réseaux de capteurs 7

I.4 Les réseaux de capteurs sans fils.....8

I.5 Architecture d'un réseau de capteurs sans fil 9

I.6 Caractéristiques des réseaux de capteurs.....10

I.7 Topologies des réseaux de capteurs sans fils.....11

I.7.1 Topologie Hiérarchique.....11

I.7.2 Topologie plate (Flat) 12

I.7.3 Topologie basée Localisation 13

I.8 Pile protocolaire 14

I.9 Les différents facteurs de conception.....16

I.10 Contraintes de conception des RCSF 17

I.11 Applications des RCSF 18

I.11.1 Applications orientées temps 19

I.11.2 Applications orientées événements 19

I.11.3 Applications orientées requêtes 19

I.11.4 Applications hybrides 20

I.12 Conclusion 20

II. CHAPITRE II.....21

La consommation d'énergie dans les RCSF 21

II.1 Introduction 21

II.2 Consommation d'énergie dans les RCSF 21

II.2.1 Energie de capture 21

II.2.2 Energie de traitement 22

II.2.3 Energie de communication 22

II.3 Notion de durée de vie d'un réseau 23
II.4 Facteurs intervenants dans la consommation d'énergie 24
II.4.1 Etat du module radio 24
II.4.2 Accès au medium de transmission 25
II.5 Modèle de propagation radio 26
II.6 Routage des données 26
II.7 Techniques de minimisation de la consommation d'énergie 27
II.8 Conclusion 30
III. CHAPITRE III 31
L'agrégation de données dans les RCSF 31
III.1 Introduction 31
III.2 Agrégation de données 32
III.3 Avantages et inconvénients de l'agrégation de données 34
III.4 Terminologie 35
III.5 Les couches utilisant cette technique 36
III.6 Fonctionnement des agrégateurs 37
III.7 Fonctions d'agrégat 38
III.8 Méthodes d'agrégation de données 40
III.9 Modèle d'agrégation 41
III.9.1 Modèle à un agrégateur 41
III.9.2 Modèle à multiple agrégateur 41
III.10 Types d'agrégation de données dans les RCSFs 42
III.10.1 Agrégation basée sur l'architecture du réseau 42
A)- Réseaux à plat 42
1- Mode Push 42
2- Mode Pull 43
B) - Réseaux hiérarchisés 44
1- Agrégation centralisé 44
2- Agrégation par chaine 45
3- Agrégation distribué 45
III.10.2 Agrégation basée sur les flux dans le réseau 46
III.10.3 Agrégation basée sur la qualité de service 46

III.11 Protocoles d'agrégation 47
III.11.1 Réseaux à plat 47
1- SPIN 47
III.11.2 Agrégation centralisé 48
1- LEACH 48
2- TEEN 49
3- APTEEN 50
III.11.3 Agrégation par chaine 51
1- PEGASIS 51
III.11.3 Agrégation distribué 53
1- COUGAR 53
2- TAG 54
3- TINA 56
4- DQEB 56
III.12 Sécurité de l'agrégation de données 57
III.12.1 Problématique de la sécurité dans l'agrégation de données 57
III.12.2 Les principales formes d'attaques 57
III.13 Conclusion 59
IV. CHAPITRE IV 60
Fonctionnement de LEACH 60
IV.1 Introduction 60
IV.2 Protocoles MAC utilisés par LEACH 61
IV.2.1 Accès aléatoire 61
IV.2.2 Allocation fixe 61
IV.2.2.1 TDMA 62
IV.2.2.2 CDMA 62
IV.3 Architecture de communication de LEACH 63
IV.4 Algorithme détaillé de LEACH 64
IV.4.1 Phase d'initialisation 64
IV.4.1.1 Phase d'annonce 65
IV.4.1.2 Phase d'organisation de groupes 67
IV.4.1.3 Phase d'ordonnancement 67
IV.4.2 Phase de transmission 68
IV.5 Avantages et inconvénients de LEACH 68

VI.5.1 Avantages 69
VI.5.2 Inconvénients 70
IV.6 Conclusion 71
V. CHAPITRE V 72
Implémentation et évaluation de LEACH 72
V.1 Introduction 72
V.2 Environnement de simulation 73
V.2.1 TinyOS 73
V.2.1.1 Pourquoi TinyOS ? 73
V.2.2.2 Notions principales 74
V.2.2 NesC 74
V.2.3 TOSSIM 75
V.2.1.1 TinyViz 76
V.2.2.2 PowerTOSSIM 76
V.3 Description de l'exemple à étudier 76
V.4 Implémentation du protocole LEACH 77
V.4.1 Structures de données 77
V.4.2 Evénements et commandes 78
V.4.3 Déroulement 78
V.5 Evaluation et simulation de performances 93
V.5.1 Métriques à évaluer 93
V.5.1.1 Consommation énergétique 93
V.5.1.2 Perte de paquets 93
V.5.1.3 Délai de bout-en-bout 94
V.5.2 Résultats et interprétations 94
V.5.2.1 Consommation énergétique 95
V.5.2.2 Perte de paquets 96
V.5.2.3 Délai de bout-en-bout 96
V.6 Conclusion 97
CONCLUSION GENERALE 98
LISTE DES REFERENCES 100
ANNEXE 105

Liste des figures

Chapitre 1 :

Figure 1.1: Architecture d'un capteur……..5

Figure 1.2: Anatomie d'un capteur sans fil……..6

Figure 1.3 : Rayons de communication et de sensation d'un capteur…….7

Figure 1.4 : Berkeley Mote……..8

Figure 1.5 : Exemple de réseau de capteurs…….9

Figure 1.6 : Architecture d'un Réseau de Capteur Sans Fil……...10

Figure 1.7 : Topologie Hiérarchique (LEACH)……..…….12

Figure 1.8 : Topologie plate (Flat)……..…….13

Figure 1.9 : Topologie Basée Localisation.....…...14

Figure 1.10 : pile protocolaire……15

Figure 1.11: Agrégation de données……..……..18

Chapitre2 :

Figure 2.1 : Modèle de consommation d'énergie……23

Figure 2.2: La sur écoute dans une transmission……...……25

Figure 2.3 : Les techniques de conservation d'énergie……...…27

Chapitre3 :

Figure 3.1 : La relation entre les terminologies de l'agrégation données……...36

Figure 3.2 : Modèle d'agrégation……..…….42

Figure 3.3 : Configuration de clusters…….……45

Figure 3.4 : Fonctionnement du protocole SPIN…….48

Figure 3.5 : le protocole TEEN……50

Figure 3.6 : fonctionnement de COUGAR……..53

Figure 3.7 : Le protocole TAG……54

<u>Chapitre4</u> :

Figure 4.1 : Diagrammes représentant le protocole MAC TDMA…….62

Figure 4.2 : Diagrammes représentant le protocole MAC CDMA……63

Figure 4.3 : Architecture de communication du protocole LEACH…….63

Figure 4.4 : Opérations de l'étape d'initialisation de LEACH……..64

Figure 4.5 : Interférence lors d'une communication dans LEACH.....67

Figure 4.6 : Répartition du temps et différentes phases pour chaque round…….68

<u>Chapitre5</u> :

Figure 5.1 : Le positionnement des 10 nœuds sur la zone d'étude (exemple)…….79

Figure 5.2 : Déclenchement round, et annonce du CH 7……..81

Figure 5.3 : Déclenchement du round, réception des nœuds de l'annonce de puits…….82

Figure 5.4 : L'élection du nœud 7 comme CH…….83

Figure 5.5 : Les transmissions unicast du CH 7……...84

Figure 5.6 : L'annonce aux nœuds du statu de nœud 7 (CH)……..85

Figure 5.7 : Formation de groupes……..87

Figure 5.8 : Phase d'ordonnancement……88

Figure 5.9 : L'envoi des températures au CH 7…….89

Figure 5.10 : La collecte et l'agrégation des données par le CH 7……90

Figure 5.11 : Envoi du résultat d'agrégation des températures au nœud puits……..91

Figure 5.12 : Réception de la température finale et déclenchement du nouveau round…….92

Figure 5.13 : Energie consommée par un nœud……95

Introduction Générale

Les capteurs sont des composants très petits. Ils sont utilisés essentiellement pour surveiller un environnement. Plusieurs technologies ont contribué à produire des composants de type capteur plus petits et à un prix faible. En plus, récemment ces composants sont munis d'un système de communication leur permettant de communiquer avec d'autres capteurs, et ainsi les réseaux de capteurs sans fil sont nés.

Ces réseaux sont différents des autres réseaux sans fil car ils ont en général les spécificités suivantes : une grande densité, faible débit, faible capacité d'énergie et un environnement inaccessible. Ces deux dernières spécificités ont fait de l'énergie une contrainte très importante puisque les batteries des capteurs ne sont pas généralement rechargeables.

Pour prolonger la durée de vie d'un réseau de capteur sans fil tout en assurant les trois tâches principales d'un nœud capteur : capture, traitement et l'envoi des données, il faut bien conserver l'énergie des nœuds capteurs.

Parmi ces trois tâches, l'envoi des données ou la communication est la tâche qui consomme la plus grande partie de l'énergie. On estime que la transmission des données d'un capteur représente environ 70% de sa consommation d'énergie.

De plus, les réseaux de capteurs étant assez denses en général, cela signifie que des nœuds assez proches en terme de distance (voisins) peuvent capter les mêmes données (température, pression, humidité équivalentes par exemple) et donc il apparaît nécessaire d'introduire le mécanisme d'agrégation de données afin d'éviter la duplication d'information au sein du réseau de capteurs et donc de préserver leur énergie et donc d'augmenter la durée de vie du réseau. Ceci a motivé des travaux de recherche à se focaliser sur cette technique, dans notre travail, nous nous intéressons à cette dernière.

L'objectif principal de notre travail est de s'initier au domaine des réseaux de capteurs sans fil : étudier la technique d'agrégation de données, parcourir les protocoles de routage

s'appuyant sur cette technique et enfin de faire une étude détaillé de l'un de ces protocoles, de l'implémenter et représenter les résultats obtenu. Le présent rapport est scindé en 5 chapitres.

Pour mieux cerner les enjeux de notre étude, dans le premier chapitre, nous présenterons les réseaux de capteurs sans fil : leurs architectures de communication et leurs applications, nous discuterons également les principaux facteurs et contraintes qui influencent la conception des réseaux de capteurs sans fil.

Pour bien situer notre thème, dans le deuxième chapitre nous décrirons la problématique de la consommation d'énergie dans les réseaux de capteurs sans fil, nous présenterons aussi les principales solutions proposées dans la littérature pour la gestion de la consommation de l'énergie et parmi ses solutions nous trouvant l'agrégation de données dans un réseau qui fait l'objet de notre étude.

Dans le troisième chapitre, nous établissons une étude détaillée de l'agrégation de données dans un réseau, nous expliquerons ses avantages et ses inconvénients, nous décrirons les technique d'agrégation de données, nous donnerons les classifications des protocoles de routage utilisant cette technique, en citant quelques exemples de ces protocoles, et en dernier nous expliquerons la problématique de sécurité dans cette technique.

L'un de ces exemples est le protocole LEACH que nous expliquerons en détail dans le quatrième chapitre: son architecture de communication, son algorithme, ses caractéristiques ainsi que les attaques pouvant perturber son fonctionnement.

Le cinquième chapitre, quant à lui, nous permettra d'exposer les résultats d'implémentation et de tests de simulation de se protocole, précédé par une présentation des outils nécessaires pour notre réalisation à savoir le système d'exploitation TinyOS, le langage de programmation NesC et le simulateur TOSSIM.

Enfin, nous clôturons par une conclusion et des perspectives.

CHAPITRE I :
LES RESEAUX DE CAPTEURS

I.1. Introduction :

Au cours des dernières décennies, nous avons assisté à une miniaturisation du matériel informatique. Cette tendance à la miniaturisation a apporté une nouvelle génération de réseaux informatiques et télécoms présentant des défis importants. Les réseaux de capteurs sans fil sont l'une des technologies visant à résoudre les problèmes de cette nouvelle ère de l'informatique embarquée et omniprésente.

La mise en œuvre de simples possibilités de traitement, de stockage, de détection et de communication dans des dispositifs à petite échelle, à faible coût et leur intégration dans ce qu'on appelle des réseaux de capteurs sans fil ouvrent la porte à une multitude de nouvelles applications. Les réseaux de capteurs constituent une catégorie de réseaux sans fil comportant d'un très grand nombre de nœuds. Ils sont également caractérisés entre autre par un déploiement très dense et à grande échelle dans des environnements souvent limités en terme de ressources. Ces nœuds déployés autour ou dans une zone à observer sont utilisés pour l'acquisition de données et leur transmission à une station de traitement appelée communément « Station de Base ». Les spécificités les plus frappantes de ces nœuds sont leurs capacités d'auto-organisation, de coopération, leur rapidité de déploiement, leur tolérance aux erreurs et leur faible coût.

En terme de domaines d'applications, les réseaux de capteurs ont connu un très grand succès, car ils détiennent un potentiel qui révolutionne de nombreux secteurs de notre économie et notre vie quotidienne, de la surveillance et la préservation de l'environnement, à la fabrication industrielle, en passant par l'automatisation dans les secteurs de transport et de la santé, la modernisation de la médecine, de l'agriculture, de la télématique et de la logistique ... etc.

I.2. Le capteur et le capteur intelligent:

- Un capteur est un dispositif équipé de fonctionnalités de sensation avancées. Il mesure ou détecte un événement réel, comme le mouvement, la chaleur ou la lumière et convertit la valeur mesurée dans une représentation analogique ou numérique. Il prélève des informations et élabore à partir d'une grandeur physique (information d'entrée), une autre grandeur physique de nature électrique.

- Les capteurs intelligents (Smart Sensors) sont des dispositifs matériels dans lesquels coexistent le(s) capteur(s) et les circuits de traitement et de communication. Leurs relations avec des couches de traitement supérieures vont bien au-delà d'une simple « transduction de signal ». Les capteurs intelligents sont des « capteurs d'informations » et non pas simplement des capteurs et des circuits de traitement du signal juxtaposés. De plus, les « Smart Sensors » ne sont pas des dispositifs banalisés car chacun de leurs constituants a été conçu dans l'objectif d'une application bien spécifique. [1]

I.2.1. Architecture d'un capteur :

Un capteur est composé de quatre composants de base [2] comme représentée dans la figure: une unité d'acquisition, une unité de traitement, une unité de communication et une source d'énergie. Ils peuvent également avoir d'autres composants dépendant de l'application tels qu'un système de localisation, un générateur d'énergie et un mobilisateur.

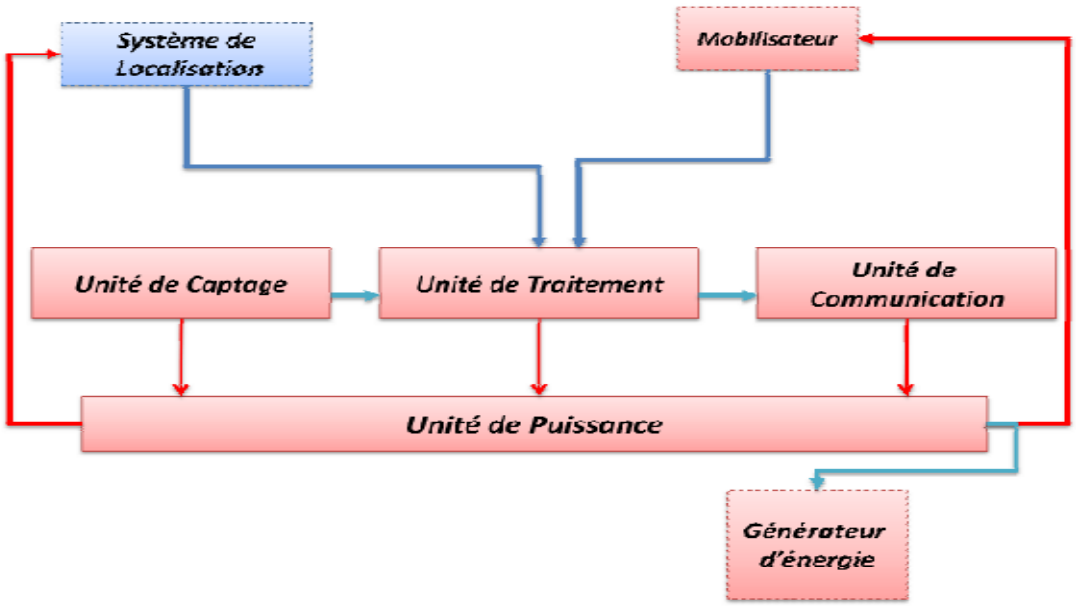


Figure 1.1: Architecture d'un capteur. [2]

- **L'unité d'acquisition des données:**

Elle se compose de deux sous unités, unité de captage qui détecte le phénomène physique (humidité, pression, accélération, sons, image, vidéo etc.) et un convertisseur analogique numérique (CAN) qui transforme le signal détecté en en signal numérique.

- **L'unité de traitement des données :**

L'unité de traitement comprend un processeur avec une petite unité de stockage RAM pour les données et cette unité fonctionne à l'aide d'un système d'exploitation spécialement conçu pour les micro capteurs (TinyOS par exemple) . Elle est chargée de gérer des procédures qui permettent à un nœud capteur de collaborer avec les autres nœuds du réseau. Elle peut aussi analyser les données captées pour alléger la tâche du nœud puits.

- **L'unité de transmission de données :**

Cette unité est responsable d'effectuer toutes les émissions et réceptions des données sur un medium sans fil. Les composants utilisés pour réaliser la transmission sont des composants classiques, les unités de transmission de type radio-fréquence (RF) sont préférables pour les RCSF parce que les paquets transportés sont de petites tailles avec un bas débit.

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 alors que pour les réseaux de capteurs, le système de transmission possède une portée de quelques dizaines de mètres. Pour augmenter ces distances tout en préservant l'énergie, le réseau utilise un routage multi sauts.

- **L'unité d'énergie :**

Les capteurs sont de petits composants alimentés avec une batterie ou avec des piles. Pour qu'un réseau de capteurs reste autonome pendant une durée de quelques mois à quelques années sans intervention humaine, la consommation d'énergie devient le problème fondamental. Celle-ci n'est pas un grand problème pour les réseaux sans fil traditionnel, car on peut toujours recharger les batteries des dispositifs sans fil comme les téléphones portables ou les ordinateurs portables.

Mais, dans un RCSF, il est difficile (parfois impossible dans certaine applications) de changer la batterie. Cette unité peut aussi contenir des systèmes de rechargement d'énergie à partir de l'environnement observé telles que les cellules solaires, afin d'étendre la durée de vie totale du réseau.

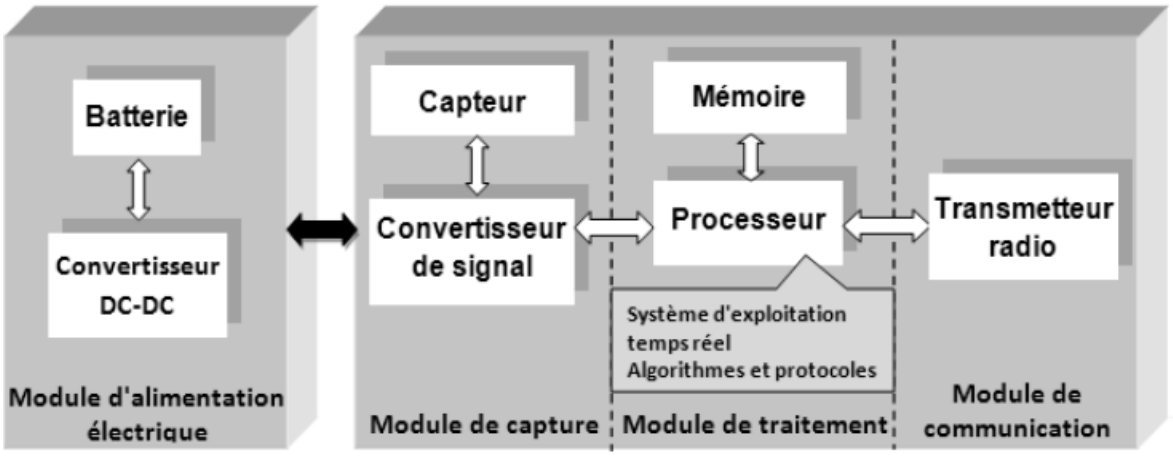


Figure 1.2:Anatomie d'un capteur sans fil. [2]

I.2.2. Caractéristiques principales d'un capteur :

Deux entités sont fondamentales dans le fonctionnement d'un capteur : l'unité d'acquisition qui est le cœur physique permettant la prise de mesure et l'unité de transmission qui réalise la transmission de celle-ci vers d'autres dispositifs électroniques. Ainsi, fonctionnellement chaque capteur possède un rayon de communication (Rc) et un rayon de sensation (Rs). La Figure 1.3 montre les zones définies par ces deux rayons pour le capteur A. La zone de communication est la zone où le capteur A peut communiquer avec les autres capteurs (le capteur B dans la Figure1.3). D'autre part, la zone de sensation est la zone où le capteur A peut capter l'événement. [3]

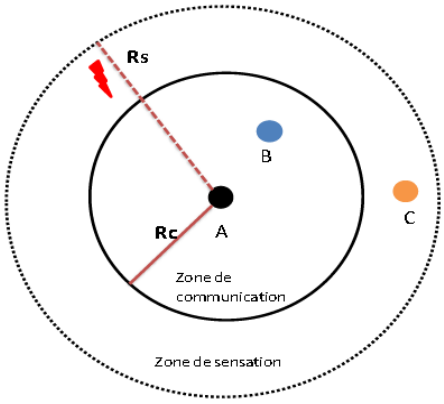


Figure 1.3 : Rayons de communication et de sensation d'un capteur [3]

En effet, pour qu'un capteur ait une portée de communication suffisamment grande, il est nécessaire d'utiliser un signal assez puissant. Cependant, l'énergie consommée serait importante.

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. La Figure 1.4 montre un capteur intelligent MICA2 fabriqué par Crossbow [4].

I.3. Les réseaux de capteurs :

Un réseau de capteur est composé de centaines ou de milliers de mini-ordinateurs. Ces appareils, appelés en anglais « **motes** », sont alimentés par des piles et sont typiquement déployés de façon aléatoire dans des environnements souvent ouverts. Généralement, ces capteurs font des mesures périodiques et envoient les données collectées à un dispositif plus

puissant, le puits (sink) ou la station de base, qui les traite en calculant par exemple leur maximum, moyenne ou médiane.

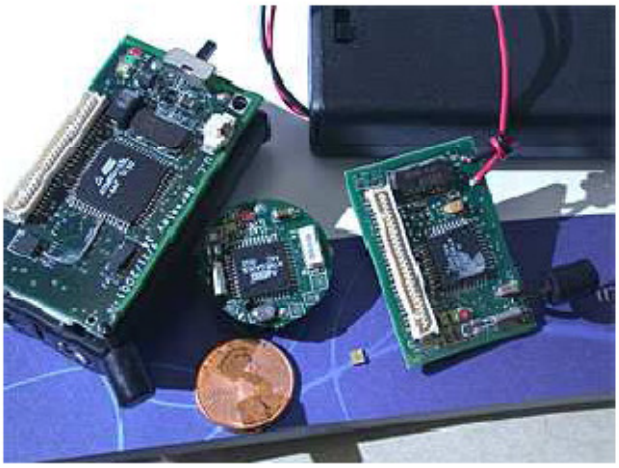


Figure 1.4 : Berkeley Mote. [5]

I.4. Les réseaux de capteurs sans fils :

Un réseau de capteurs sans fil (RCSF ou WSN : Wireless Sensor Network) est composé d'un grand nombre de nœuds qui sont des capteurs intelligents distribués sur une zone donnée afin de mesurer une grandeur physique ou surveiller un évènement et de réagir en cas de besoin en envoyant l'information collectée à un ou plusieurs points de collecte, à l'aide d'une connexion sans fil. Dans un tel réseau, chaque nœud est un dispositif électronique qui possède une capacité de calcul, de stockage, de communication et d'énergie. [5]

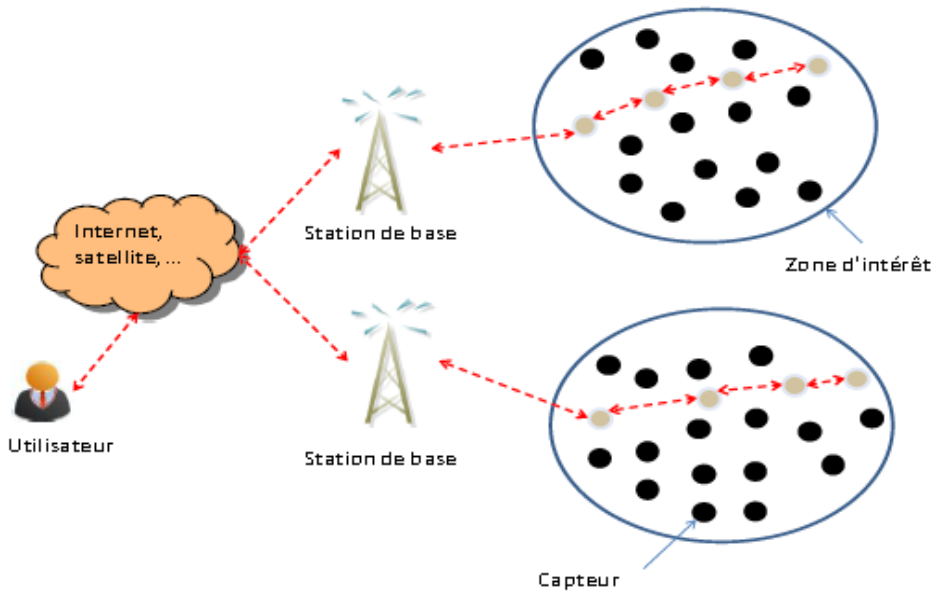


Figure 1.5 : Exemple de réseau de capteurs. [5]

Un exemple de réseaux de capteurs est représenté dans la Figure 1.5 : les capteurs sont déployés d'une manière aléatoire dans une zone d'intérêt, et une station de base, située à l'extrémité de cette zone, est chargée de récupérer les données collectées par les capteurs. Lorsqu'un capteur détecte un événement pertinent, un message d'alerte est envoyé à la station de base par le biais d'une communication entre les capteurs. Les données collectées sont traitées et analysées par des machines puissantes.

I.5. Architecture d'un réseau de capteurs sans fil :

Un réseau de capteurs sans fil est composé d'un grand nombre de nœuds. Chaque capteur est doté d'un module d'acquisition qui lui permet de mesurer des informations environnementales : température, humidité, pression, accélération, sons, image, vidéo etc.

Les données collectées par ces nœuds capteurs sont routées vers une ou plusieurs stations de base ou nœud puits ou nœud passerelle (**sink** en anglais). Ce dernier est un point de collecte de données capturées. Il peut communiquer les données collectées à l'utilisateur final à travers un réseau de communication, éventuellement l'Internet ou un satellite. L'utilisateur peut à son tour utiliser la station de base comme passerelle, afin de transmettre ses requêtes au

réseau. En général, un RCSF est composé de quatre éléments montrés par la figure : les capteurs, une station de base (puits), phénomène à mesurer et l'utilisateur. [6]

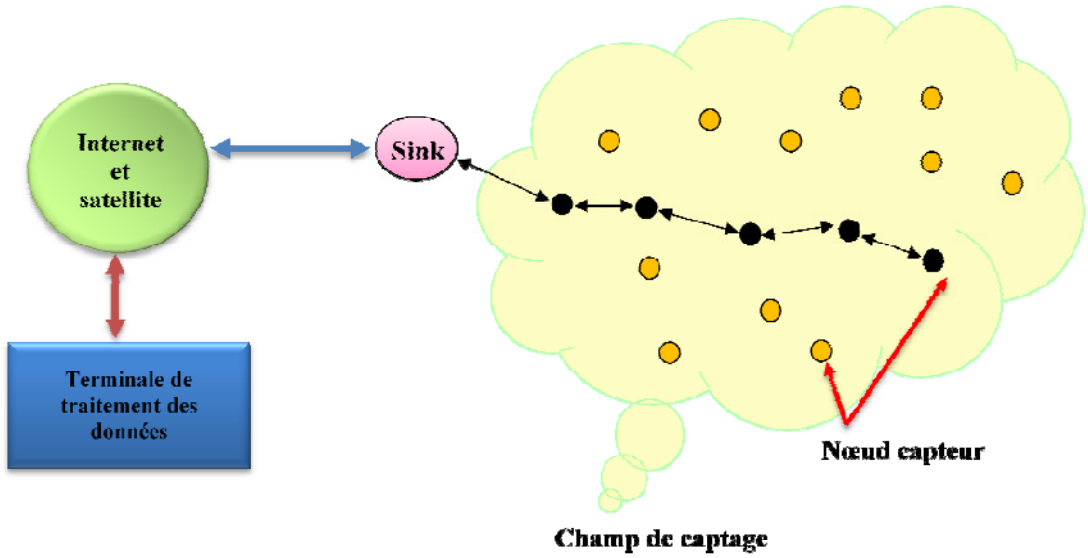


Figure 1.6 : Architecture d'un Réseau de Capteur Sans Fil. [6]

I.6. Caractéristiques des réseaux de capteurs :

Un réseau de capteurs présente les caractéristiques suivantes [5] :

- **absence d'infrastructure** : les réseaux Ad-hoc en général, et les réseaux de capteurs en particulier se distinguent des autres réseaux par la propriété d'absence d'infrastructure préexistante et de tout genre d'administration centralisée.
- **taille importante** : un réseau de capteurs peut contenir des milliers de nœuds.
- **interférences** : les liens radio ne sont pas isolés, deux transmissions simultanées sur une même fréquence, ou utilisant des fréquences proches, peuvent interférer.
- **topologie dynamique** : les capteurs peuvent être attachés à des objets mobiles qui se déplacent d'une façon libre et arbitraire rendant ainsi la topologie du réseau fréquemment changeante.
- **sécurité physique limitée** : les réseaux de capteurs sans fil sont plus touchés par le paramètre de sécurité que les réseaux filaires classiques. Cela se justifie par les contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé.

- **bande passante limitée** : une des caractéristiques primordiales des réseaux basés sur la communication sans fil est l'utilisation d'un médium de communication partagé. Ce partage fait que la bande passante réservée à un nœud est limitée.
- **contrainte d'énergie** : de stockage et de calcul - la caractéristique la plus critique dans les réseaux de capteurs est la modestie de ses ressources énergétiques car chaque capteur du réseau possède de faibles ressources en termes d'énergie (batterie). Afin de prolonger la durée de vie du réseau, une minimisation des dépenses énergétiques est exigée chez chaque nœud. Ainsi, la capacité de stockage et la puissance de calcul sont limitées dans un capteur.

I.7. Topologies des réseaux de capteurs sans fils :

Les topologies des réseaux de capteurs sont déterminées à partir des protocoles de routages utilisés pour l'acheminement des données entre les nœuds et le puits. Ces protocoles peuvent être hiérarchiques, plats (Flat) ou basés localisation.

I.7.1. Topologie Hiérarchique :

Les protocoles à topologie hiérarchique forment des réseaux dans lesquels un nœud central *Sink* (le niveau supérieur de la hiérarchie) est relié à un ou plusieurs autres nœuds qui appartiennent à un niveau plus bas dans la hiérarchie (deuxième niveau) avec une liaison point à point. Aussi, chacun des nœuds du deuxième niveau aura également un ou plusieurs autres nœuds de niveau plus bas dans la hiérarchie (troisième niveau) reliées à lui avec une liaison point à point. Chaque ensemble de nœuds forme une sorte de motif (Cluster). Le nœud central n'a aucun autre nœud au-dessus de lui dans la hiérarchie sauf le centre de traitement des données ou la passerelle si elle existe. Les nœuds du deuxième niveau jouent le rôle des passerelles entre ceux du troisième niveau et le Sink. Dans ce cas, le routage devient plus simple, puisqu'il s'agit de passer par les passerelles pour atteindre le nœud destination [7].

Dans certains types de protocoles (tel que LEACH dans les WSN), un algorithme d'élection est exécuté dans chaque cluster, les nœuds élisent un d'eux pour être Clusterhead. L'élection est basée sur des critères tels que l'énergie disponible, la qualité de communication,

et ainsi de suite, ou la combinaison de plusieurs d'entre elles. Le rôle du Clusterhead est la collecte des informations issues des nœuds et les renvoyer vers le puits.
Un réseau basé sur une topologie hiérarchique doit avoir au moins trois niveaux dans sa hiérarchie, puisqu'un réseau avec un nœud central *Sink* et seulement un niveau hiérarchique au-dessous, forme une topologie en étoile.

Si les nœuds dans un réseau basé sur la topologie hiérarchique doivent effectuer un tel traitement sur les données transmises entre les nœuds dans le réseau, alors les nœuds qui sont à des niveaux plus élevés dans la hiérarchie doivent effectuer plus de traitement que les nœuds de niveau inférieur.
Dans le cas de LEACH, les informations sont transmises d'un nœud capteur vers le nœud puits en passant par le Clusterhead déjà élu comme c'est illustré dans la figure 1.7.

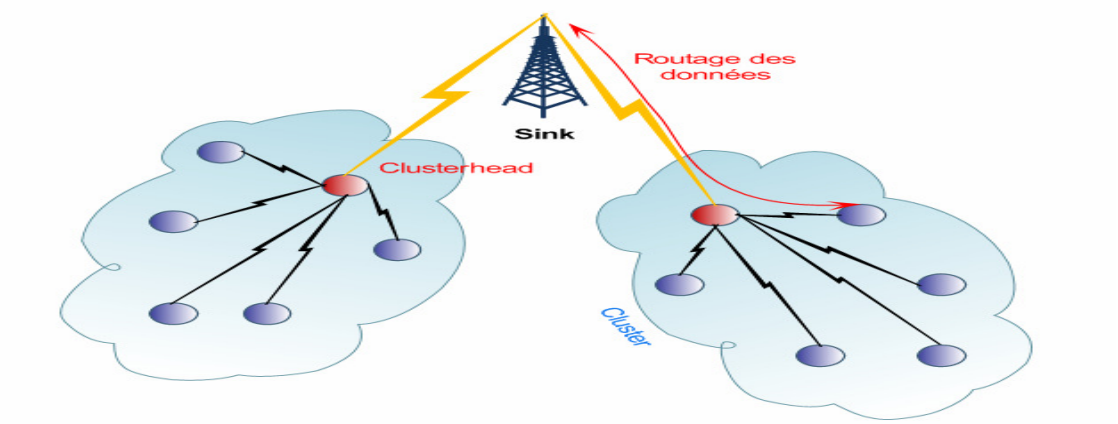


Figure 1.7 : Topologie Hiérarchique (LEACH) [7]

À titre d'exemple des protocoles utilisant une topologie hiérarchique on peut citer le protocole LEACH *(*Low-energy Adaptive Clustering Hierarchy*)*, CBRP (Cluster Based Routing Protocol).

I.7.2. Topologie plate (Flat) :

Les protocoles à topologie plate (flat) considèrent que tous les nœuds sont égaux, ont les mêmes fonctions, et peuvent communiquer entre eux sans devoir passer par un nœud particulier ou une passerelle. Seul un nœud particulier, le Sink, est chargé de la collecte des

données issues des différents nœuds capteurs afin de les transmettre vers les centres de traitement.

En cas où la destination ne fait pas partie du voisinage de la source, les données seront transmises en utilisant les sauts multiples à travers les nœuds intermédiaires comme c'est illustré dans la figure 1.8. Ce type de réseau représente l'avantage de l'existence de différents chemins d'une source vers une destination et c'est pour remédier au problème de changement brusque de topologie ou la défaillance d'un nœud intermédiaire [8].

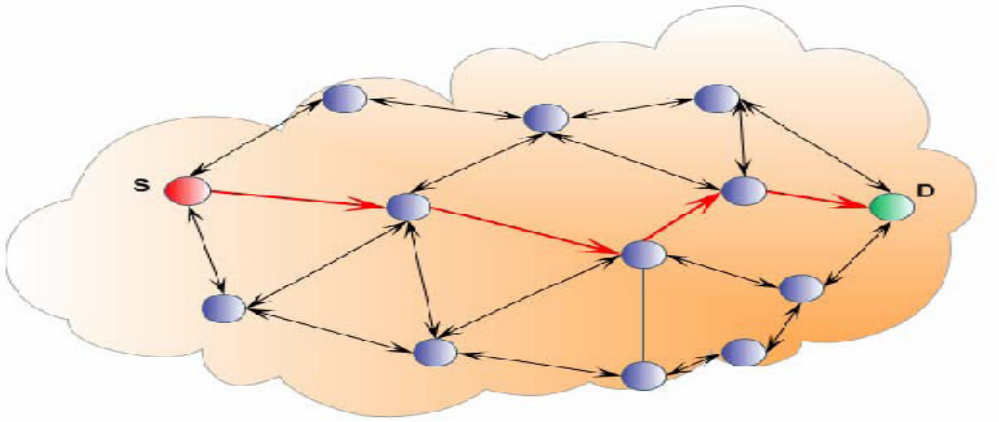


Figure 1.8 : Topologie plate (Flat) [8]

À titre d'exemple des protocoles utilisant une topologie plate on peut citer le protocole Direct Diffusion.

I.7.3. Topologie basée Localisation :

Les protocoles à topologie basée localisation suppose que :

- Le réseau est partitionné en plusieurs zones de localisation.
- Chaque zone a son identifiant.
- Chaque nœud a un identifiant EUI (End-system Unique Identifier) et enregistre dynamiquement l'identifiant de la zone à laquelle il appartient temporairement.

L'information temporaire de localisation appelée LDA (Location Dependent Address) qui est un triplet de coordonnées géographiques (longitude, latitude, altitude) obtenues, par exemple, au moyen d'un GPS avec une précision dépendant du type de l'application. Une telle topologie exige l'implémentation d'un algorithme de gestion de localisation qui permet aux nœuds de déterminer les endroits approximatifs des autres nœuds. Ce type de topologie est mieux adapté aux réseaux avec une forte mobilité.

Avant d'envoyer ses données à un nœud destination, le nœud source utilise un mécanisme pour déterminer la localisation de la destination puis inclus l'identifiant de zone de localisation et du nœud destination dans l'entête du paquet à envoyer [8].

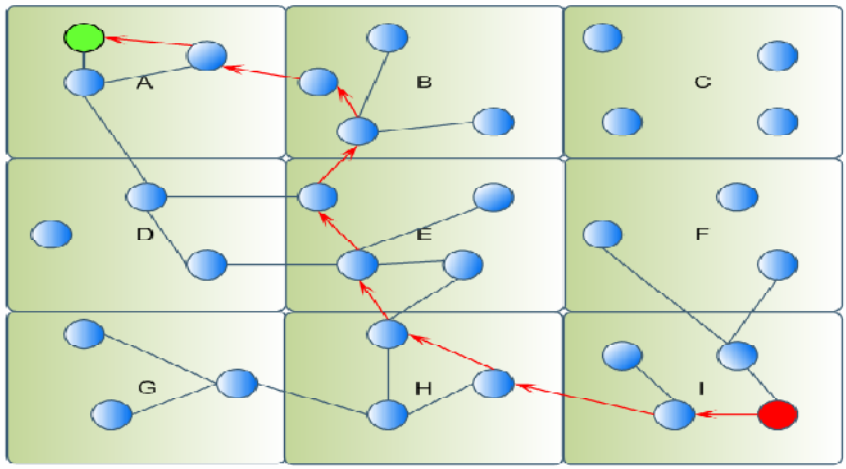


Figure 1.9 : Topologie Basée Localisation [8]

À titre d'exemple des protocoles utilisant une topologie basée localisation nous pouvons citer GEAR (Geographic and Energy Aware Routing) et LAR (Location-Aided Routing protocol).

I.8. Pile protocolaire :

La pile protocolaire utilisée par la station de base ainsi que tous les autres capteurs illustrés par la figure 1.10. Cette pile de protocoles combine routage et gestion d'énergie et intègre les données avec les protocoles réseau. Elle communique de manière efficace (en terme d'énergie) à travers le support sans fil et favorise les efforts de coopération entre les nœuds-capteurs. La pile protocolaire comprend la couche application, la couche transport, la couche réseau, la couche liaison de donné, la couche physique, le plan de gestion de l'énergie, le plan de gestion de la mobilité et le plan de gestion des tâches. [9] [10]

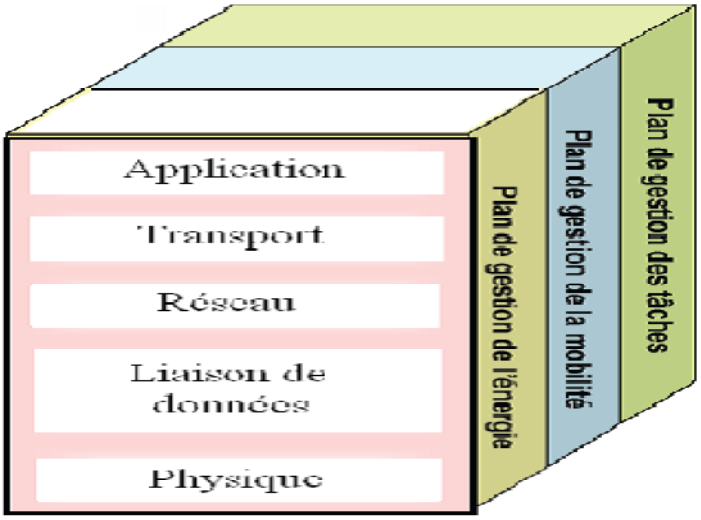


Figure 1.10 : pile protocolaire [9] [10]

- Suivant la fonctionnalité des capteurs, différentes applications peuvent être utilisées et bâties sur la couche application.
- La couche transport 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 venu de la couche réseau avant de les envoyer à la couche application
- La couche réseau prend soin de router les données fournies par la couche transport. Le protocole MAC (Media Access Control) de la couche liaison assure la gestion de l'accès au support physique.
- La couche physique 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. Ces plans de gestion sont nécessaires , de sorte que les nœuds capteurs puissent fonctionner ensemble d'une manière efficace pour préserver l'énergie, router des données dans un réseau de capteur mobile et partager les ressources entre les nœuds de capteurs. Du point de vue global, il est plus efficace d'utiliser des nœuds capteurs pouvant collaborer entre eux. La durée de vie du réseau peut être ainsi prolongée.

I.9. Les différents facteurs de conception :

La conception des réseaux de capteurs est influencée par de nombreux facteurs comme la tolérance aux pannes, les couts de production, la consommation d'´energie, l'environnement ou la topologie du réseau. Ces facteurs représentent la base de la conception de protocoles ou d'algorithmes pour les réseaux de capteurs.

Tolérance aux pannes : Les nœuds peuvent être sujets à des pannes dues à leur fabrication (ce sont des produits de série bon marché, il peut donc y avoir des capteurs défectueux) ou plus fréquemment à un manque d'énergie. Les interactions externes (chocs, interférences) peuvent aussi être la cause des dysfonctionnements. Afin que les pannes n'affectent pas la tâche première du réseau, il faut évaluer la capacité du réseau à fonctionner sans interruption.

Coût de fabrication : Les nœuds sont des produits fabriqués en série du fait de leur grand nombre. Il faut que le coût de fabrication de ces nœuds soit tel que le coût global du réseau ne soit pas supérieur à celui d'un réseau classique afin de pouvoir justifier son intérêt.

Topologie du réseau : En raison de leur forte densité dans la zone à observer, il faut que les nœuds-capteurs soient capables d'adapter leur fonctionnement afin de maintenir la topologie souhaitée.
On distingue généralement trois phases dans la mise en place et l'évolution d'un réseau :

- Déploiement : Les nœuds sont soit répartis de manière prédéfinie soit de manière aléatoire (lancés en masse depuis un avion). Il faut alors que ceux-ci s'organisent de manière autonome.
- Post-Déploiement : Exploitation : Durant la phase d'exploitation, la topologie du réseau peut être soumise à des changements dus à des modifications de la position des nœuds ou bien à des pannes.
- Redéploiement : L'ajout de nouveaux capteurs dans un réseau existant implique aussi une remise à jour de la topologie.

Consommation d'énergie : L'économie d'énergie est une des problématiques majeures dans les réseaux de capteurs. En effet, la recharge des sources d'énergie est souvent trop coûteuse et parfois impossible. Il faut donc que les capteurs économisent au maximum l'énergie afin de

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

I.10. Contraintes de conception des RCSF :

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.

- **Durée de vie du réseau :** C'est l'intervalle de temps qui sépare l'instant de déploiement du réseau de l'instant où l'énergie du premier nœud s'épuise. Selon l'application, la durée de vie exigée pour un réseau peut varier entre quelques heures et plusieurs années.
- **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.
- **Bande passante limitée :** Afin de minimiser l'énergie consommée lors de transfert de données entre les nœuds, les capteurs opèrent à bas débit. Typiquement, le débit utilisé est de quelques dizaines de Kb/s. Un débit de transmission réduit n'est pas handicapant pour un réseau de capteurs où les fréquences de transmission ne sont pas importantes.
- **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.[11]
- **Topologie dynamique :** La topologie des réseaux de capteurs peut changer au cours du temps pour les raisons suivantes :

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

- **Agrégation de donnée :** L'agrégation de données est une technique qui permet de réduire le coût global des communications dans un réseau où les messages envoyés entre les nœuds ont des contenus « similaires ». Des paquets venant de plusieurs nœuds qui arrivent sur un nœud dit « agrégateur », le long du chemin vers la station de base (puits), peuvent être agrégés (avec une fonction spécifique suivant l'application) en un unique paquet. Sur le schéma de la figure 1.11, le nœud E agrège les données de A et B, le nœud F agrège les données de C et D et le nœud G agrège les données de E et F. Finalement, l'ensemble des paquets agrégés finissent par atteindre la station de base (puits, *sink*). Comme application qui se prête bien à l'agrégation, nous pouvons par exemple en citer une où le but recherché est de récupérer la température moyenne du terrain où sont déployés les nœuds. Chaque nœud envoie la température qu'il mesure, et les agrégateurs calculent une moyenne locale des températures qu'ils reçoivent avant d'envoyer le résultat au nœud suivant.
  En revanche une application qui nécessiterait de connaître la position de chaque nœud en plus de la température bénéficierait moins de l'agrégation de données, et il faudrait alors veiller à garder la donnée de localisation de chaque nœud. Il existe différentes technique d'agrégation, différentes façon de sécuriser les données. [12]

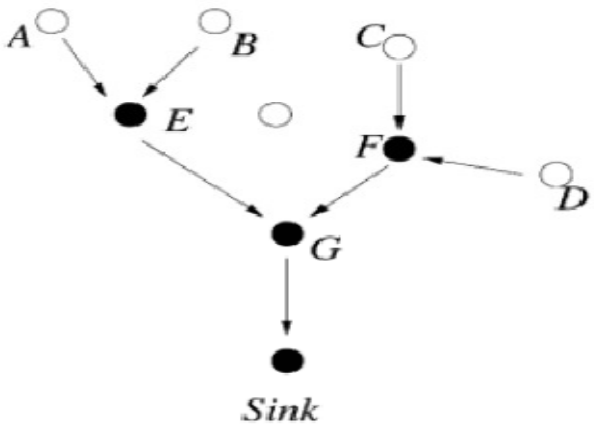


Figure 1.11: Agrégation de données. [12]

I.11. Applications des RCSF :

Les réseaux de capteurs sans fil ont été classés parmi les 21 technologies les plus importantes du 21éme siècle [13].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. Ainsi, nous classifions les applications des RCSF en quatre classes

d'applications : orientées temps (time driven), orientées événements (event driven), orientées requêtes (query driven) et hybrides [14].

I.11.1. Applications orientées temps:

Cette classe représente les applications où l'acquisition et la transmission des données capturées sont liées au temps : instant précis, période d'acquisition. Cette période d'acquisition peut être plus au moins longue selon l'application (de quelques secondes jusqu'à quelques heures voire des jours). Ainsi, la quantité de données échangées dans le réseau dépend de la périodicité des mesures à effectuer sur l'environnement local. La collecte de données environnementales peut représenter un bon exemple de cette classe d'application dans des domaines variés : agriculture, expérimentation scientifique, etc.

I.11.2. Applications 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 de 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 (barrages, ponts, voies de chemins de fer, …etc.).[15]

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. Conclusion :

Dans ce chapitre nous avons procédé à l'étude des réseaux de capteurs sans fil. Nous avons posé les briques de base et fédéré quelques concepts nécessaires à la compréhension de nos problématiques dans la suite de ce travail.

Cela fait des années que les réseaux de capteurs suscitent un engouement important dans la recherche. Nous avons remarqué à travers nos lectures que « minimiser la consommation d'énergie d'un nœud capteur » est le cheval de bataille de toutes les solutions et protocoles proposés et c'est un point que nous allons aborder dans le chapitre suivant.

CHAPITRE II :

LA COSOMMATION D'ENERGIE DANS LES RCSF

II.1. Introduction :

Les capteurs sont conçus pour fonctionner durant des mois voire des années. Ainsi, la capacité énergétique des capteurs doit être utilisée efficacement afin de maximiser la durée de vie du réseau. A noter qu'une fois qu'un nœud capteur a épuisé son énergie, il est considéré comme défaillant. Ainsi, il y a une forte probabilité de perdre la connectivité du réseau.

Dans ce chapitre, nous décrirons la problématique de la consommation d'énergie dans les réseaux de capteurs. Nous présenterons aussi les principales solutions proposées dans la littérature pour la gestion de la consommation de l'énergie.

II.2. Consommation d'énergie dans les RCSF :

L'énergie consommée par un nœud capteur est due essentiellement aux opérations suivantes : la capture, le traitement et la communication de données [16].

II.2.1 Energie de capture :

L'énergie de capture est dissipée pour accomplir les tâches suivantes : échantillonnage, traitement de signal, conversion analogique/numérique et activation de la

sonde de capture. En général, l'énergie de capture représente un faible pourcentage de l'énergie totale consommée par un nœud.

II.2.2 Energie de traitement :

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

En général, l'énergie de traitement est faible par rapport à celle nécessaire pour la communication.

II.2.3 Energie de communication :

L'énergie de communication se décline en deux parties : l'énergie de réception et l'énergie de l'émission. 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.

- **Modèle de consommation d'énergie :** Heinzelman et al. [16] proposent un modèle radio de consommation d'énergie (figure 1.4). Ainsi, les énergies nécessaires pour émettre ETx(s,d) et recevoir ERx(s) des messages sont données par :

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

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

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

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

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

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

Eelec et Eamp représentent respectivement l'énergie de transmission électronique et d'amplification (figure 2.1).

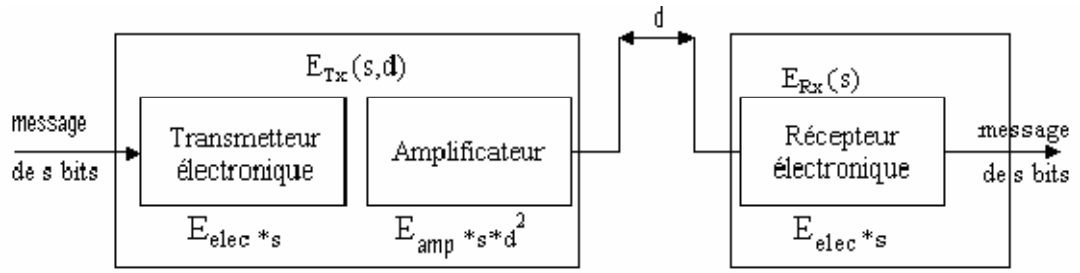


Figure 2.1 : Modèle de consommation d'énergie [16].

II.3 Notion de durée de vie d'un réseau :

Un réseau ne peut accomplir son objectif que tant qu'il est « en vie », mais pas au delà. La durée de vie prévue est critique dans tout déploiement de réseau de capteurs. Le but des scénarios applicatifs classiques consiste à déployer des nœuds dans un domaine sans surveillance pendant des mois ou des années.

La vie d'un réseau de capteurs correspond à la période de temps durant laquelle le réseau peut, selon le cas : maintenir assez de connectivité, couvrir le domaine entier, ou garder le taux de perte d'information en-dessous d'un certain niveau. La vie du système est donc liée à la vie nodale, même si elle peut en différer. La vie nodale correspond à la vie d'un des nœuds du réseau. Elle dépend essentiellement de deux facteurs : l'énergie qu'il consomme en fonction du temps et la quantité d'énergie dont il dispose.
Selon la discussion d'Akyildiz et al [67], la quantité prédominante d'énergie est consommée par un nœud-capteur durant la détection, la communication puis le traitement des données.

Il existe différentes définitions pour la durée de vie d'un réseau de capteurs (fondées sur la fonctionnalité désirée). Elle peut être définie par la durée jusqu'au moment ou le premier nœud meurt. Elle peut également être définie par le temps jusqu'au moment ou une proportion de nœuds meurt. Si la proportion de nœuds morts dépasse un certain seuil, cela peut avoir comme conséquence la non couverture de sous-régions et/ou le partitionnement du réseau.

II.4 Facteurs intervenants dans la consommation d'énergie

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

II.4.1 Etat du module radio :

Le module radio est le composant du nœud capteur qui consomme le plus d'énergie, puisque c'est lui qui assure la communication entre les nœuds. On distingue quatre états des composants radio (transmetteur et récepteur) : actif, réception, transmission et sommeil [17].

- **Etat actif** : la radio est allumée, mais elle n'est pas employée. En d'autres termes, le nœud capteur n'est ni en train de recevoir ni de transmettre. Cet état provoque une perte de l'énergie suite à l'écoute inutile du canal de transmission.
- **Etat sommeil** : la radio est mise hors tension.
- **Etat transmission** : la radio transmet un paquet.
- **Etat réception** : la radio reçoit un paquet.

Il est aussi à noter que le passage fréquent 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 appelée 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.

II.4.2 Accès au medium de transmission

La couche MAC joue un rôle important pour la coordination entre les nœuds et la minimisation de la consommation d'énergie. Dans cette section, nous allons analyser les principales causes de consommation d'énergie au niveau de la couche MAC [14].

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

- **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.

- **La surécoute :** Le phénomène de surécoute (overhearing) se produit quand un nœud reçoit des paquets qui ne lui sont pas destinés (figure 2.2). La surécoute conduit à une perte d'énergie additionnelle à cause de l'implication des autres capteurs dans la réception des données.

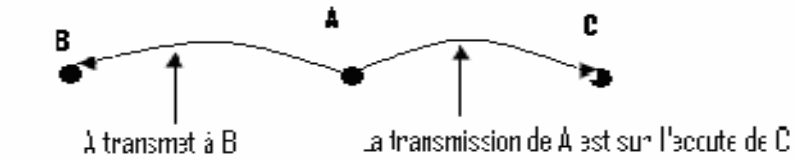


Figure 2.2: La sur écoute dans une transmission [14].

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

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

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

.

II.5 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) [18].

II.6 Routage des données :

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

II.7 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 2.3) une vue globale de différentes techniques utilisées pour minimiser cette consommation.

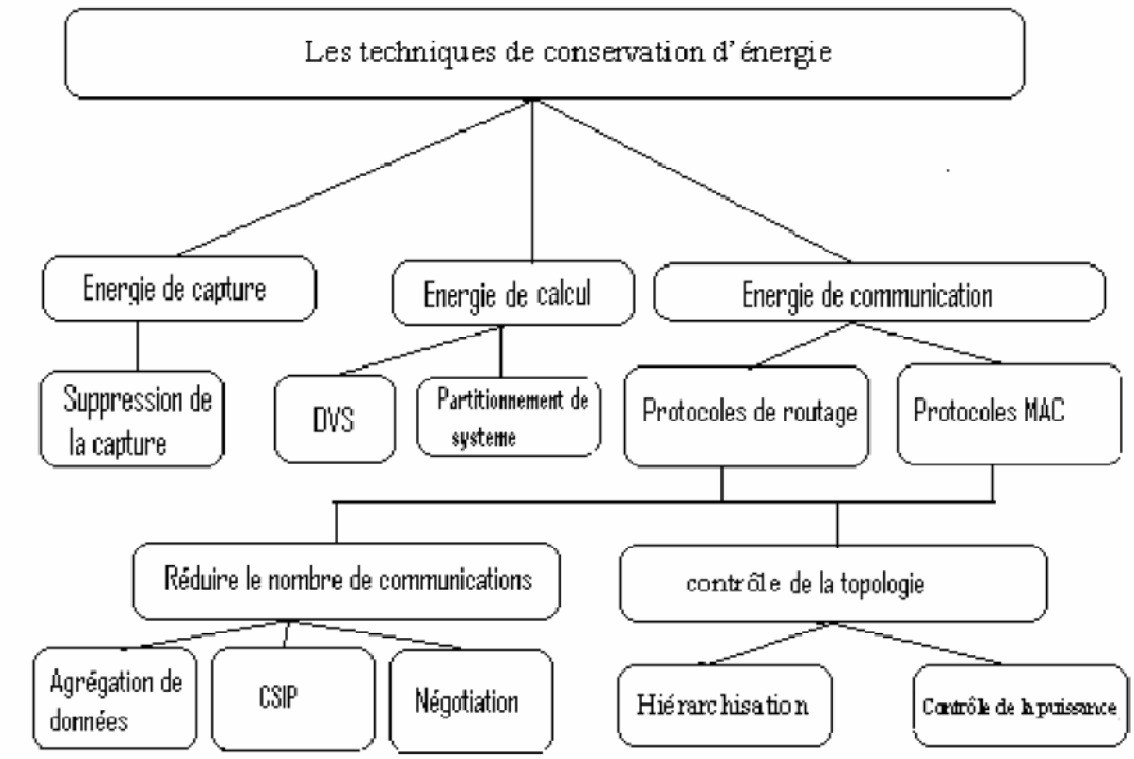


Figure 2.3 : Les techniques de conservation d'énergie. [18]

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*) [19]**,** 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 [20].

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 [21] : 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 [20] 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 [22].

- 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 [18] .

- Une autre technique a été proposée dans [23] .Cette technique 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.

Une panoplie de protocoles a été proposée dans la littérature. La grande partie de ces protocoles est destinée à la couche réseau et MAC. Les protocoles de la couche réseau dans les RCSF peuvent être divisés selon la structure du réseau en routage linéaire, routage hiérarchique et routage basé sur la localisation.

Dans le routage linéaire, tous les nœuds ont typiquement les mêmes rôles ou fonctionnalités. Cependant, dans le routage hiérarchique, les nœuds joueront différents rôles dans le réseau. Dans le routage basé sur la localisation, les positions des nœuds capteurs sont exploitées pour router les données dans le réseau. En outre, ces protocoles peuvent être classés selon leurs fonctionnements, en techniques de routage multi-trajectoires, basées sur les questions, par cohérence, par négociation ou basées sur la qualité de service .

De même, il existe plusieurs protocoles de la couche MAC qui sont développés pour les RCSF. Plusieurs classifications de ces protocoles ont été proposées dans la littérature. Dans la littérature les protocoles MAC sont classifiés en deux : les protocoles à accès centralisé et les protocoles à accès aléatoire. Dans le chapitre suivant, nous examinerons en détail les protocoles de cette couche.

II.8 Conclusion

Les réseaux de capteurs sans fil présentent un intérêt considérable et une nouvelle étape dans l'évolution des technologies de l'information et de la communication. Cette nouvelle technologie suscite un intérêt croissant étant donnée la diversité de ces applications : santé, environnement, industrie et même dans le domaine sportif.

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.

La minimisation de cette énergie est liée aux différents protocoles qui se basent sur plusieurs techniques telles que l'agrégation de données qui fera l'objet de notre prochain chapitre.

CHAPITRE III :

L'AGREGATION DE DONNEES DANS LES RCSF

III.1 Introduction :

Minimiser la consommation d'énergie revient à minimiser, entre autre, la quantité de données transmises dans le réseau. En effet, d'après les statistiques, 70% de l'énergie consommée dans un nœud capteur est due aux transmissions [68].

Dans les RCSF, les données produites par les nœuds capteurs voisins sont très corrélées spatialement et temporellement [68]. 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.

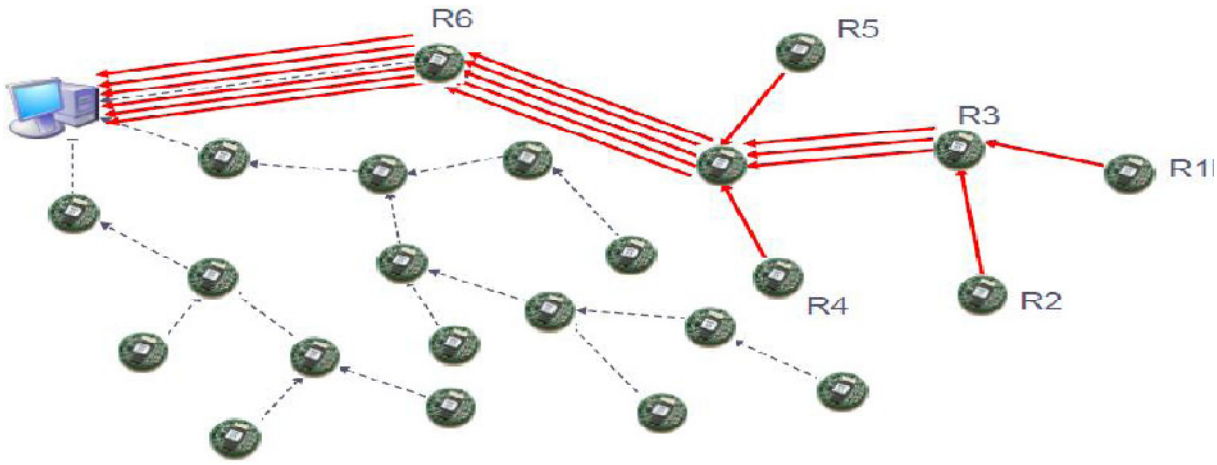
Avec cette technique, les nœuds intermédiaires agrègent l'information reçue de plusieurs sources. Cette technique est connue aussi sous le nom de fusion de données. Pour bien cerner ce point nous allons étudier dans ce chapitre les différentes techniques utilisées dans cette méthode et établir un état de l'art des protocoles s'appuyant sur cette dernière.

III.2 Agrégation de données :

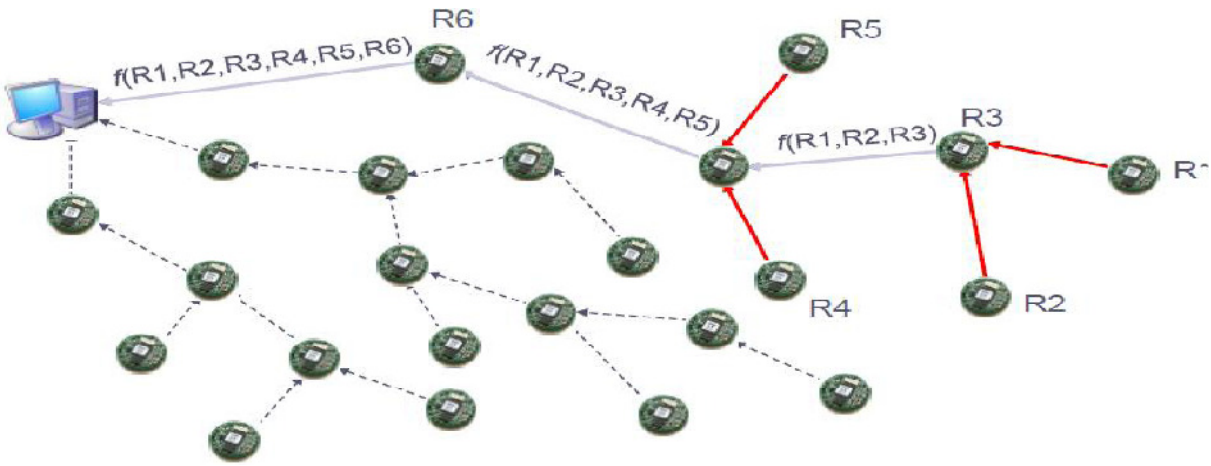
L'agrégation considérée comme approche préventive, effectue un traitement supplémentaire sur les données brutes captées depuis l'environnement. Un nœud capteur agrégateur combine les données provenant de plusieurs nœuds capteurs en une information significative ; ce qui réduit considérablement la quantité de données transmises, demande moins d'énergie et augmente ainsi la durée de vie du réseau et résout le problème d'implosion dans le routage et allège ainsi la congestion du réseau.[24]

L'agrégation de données dans les réseaux de capteurs consiste alors à remplacer les lectures individuelles de chaque capteur par une vue globale, collaborative sur une zone donnée (clustering).

On peut utiliser par exemple de simples fonctions d'agrégat telles que MIN, MAX ou MOYENNE, qui permettent à partir d'une série de n messages reçus par un « chef de zone » (capteur chef d'une zone) de ne renvoyer vers le puits qu'un seul message résumant l'information contenue dans ces n messages. [24]
Ceci réduit le nombre de messages envoyés et donc économise l'énergie.
Exemple sans agrégation : [24]



Au total, 18 messages sont envoyés sur le réseau de capteurs.
En utilisant le mécanisme d'agrégation de données, on obtient un total de 7 messages envoyés sur le réseau :



III .3 Avantages et inconvénients de l'agrégation de données :

Dans la contrainte d'énergie des nœuds capteurs sans fil, l'efficacité énergétique (la conservation d'énergie) peut être réalisée en utilisant certaines des techniques d'agrégation dans le réseau. Lors de la transmission des données, les informations provenant de diverses sources sont regroupées sur leur chemin à la station de base (puits), et ce processus est appelé l'agrégation dans un réseau. L'opérateur d'agrégation des données peut être simple (SUM, AVERAGE, MAXIMUM, MINIMUM et COUNT, etc…), ou plus compliquées, comme MEDIANE [25]**.**

L'économie d'énergie dépend du type d'opérateur d'agrégation employée. Par exemple, si l'opérateur MAX est utilisé pour l'agrégation, les résultats son mis dans un seul paquet de la même taille que celle des lectures individuels des capteurs. Si le taux de d'agrégation est n: 1, alors l'économie d'énergie sera n fois.

Supposons que l'opérateur de concaténation est utilisé pour l'agrégation, c'est à dire, les lectures de chaque capteur sont annexées par le nœud agrégateur, l'économie d'énergie a lieu seulement sur l'accès du medium. Le traitement des données a lieu dans le flanc du réseau, d'où c'est au processus d'agrégation de supprimer la transmission d'informations non désirées. Cela augmente la durée de vie des nœuds de capteurs et donc le réseau de capteurs.

Les avantages de l'agrégation de données dans un réseau sont également étendus à la station de base (puits) qui reçoit moins de quantité d'informations utiles à partir des sources du capteur et donc le puits peut effectuer moins de traitement et de filtrage pour obtenir des informations utiles à partir de ces données en consommant moins de ressources.

Même si, l'agrégation réduit la consommation d'énergie, il augmente le délai de livraison du paquet au puits. Cela est dû au fait que chaque agrégateur doit attendre un intervalle de temps prédéfini pour recueillir des données auprès de ses enfants.

Cela conduit à un retard dans la livraison des données et donc le puits ne peut pas obtenir les données fraîches. Il ya donc un compromis entre l'économie d'énergie, l'exactitude (la cohérence) des données, et la fraîcheur, c'est à dire, un nœud plus il attend, plus des lectures sont susceptible a être reçu et donc, des informations plus précise sont envoyé.

D'autre part, attendre trop longtemps peut entraîner des données périmées. Par ailleurs, si un nœud attend trop longtemps, il peut interférer avec la prochaine "vague de données".

Le tableau suivant résume tous les avantages et inconvénients de l'utilisation de l'agrégation de données [24] :

Avantages	**Inconvénients**
La réduction de la consommation d'énergie et de se fait elle augmente la durée de vie de réseau.	L'augmentation de la latence de données et le besoin d'une synchronisation stricte entre un nœud agrégateur et ses voisinages, qui pourraient devenir critiques en cas du retard des demandes sensibles.
La diminution de l'overhead (superflus, surcharge) des paquets redondants voyageant partout dans le réseau, évitant ainsi le stockage et le traitement inutile.	La réduction de l'exactitude et de l'intégrité de données en raison d'agrégation inopportune; dans ce sens, sans critères de conservation appropriés, les données livrées à la destination pourraient être incertaines et inutiles, en raison du niveau croissant d'altération.
La réduction de la charge du réseau complet ainsi la provocation d'une réduction d'effets de congestion sur les nœuds fortement chargés dans le réseau.	Le besoin de considérer l'impact de doublon en cas de l'utilisation de métrique d'agrégation plus complexe; ceci exige la distinction entre la métrique d'agrégation qui est sensible aux doublons et ceux qui ne sont pas.
Filtrage intelligent des données envoyées au sink et la livraison au centre de contrôle.	Besoin de codage approprié à cause du traitement supplémentaire exigé par les dispositifs de capteur.

III.4 Terminologie :

Des aspects différents devraient être adressés dans le processus d'agrégation de données [26] : l'interrogation par des utilisateurs distants, collection des mesures des nœuds, leur combinaison et concaténation, la compression et l'exploration à la destination finale. Par conséquence, des différents termes et leurs fonctionnalités associées doivent être discutés:

- **La fusion de données (Data fusion):** le terme de la fusion de données se réfère à la combinaison des différentes pièces de données dans une seule ; on peut imaginer, par exemple, un cas d'utilisation dans lequel l'intérêt est dans le contrôle de la température minimale dans une chambre; dans ce cas, la fusion serait aussi facile que la prise de la valeur minimale de tous les messages reçus et l'expédition d'un paquet unique. on peut voir l'agrégation de données comme un processus inclus dans la fusion de données.

- **L'interrogation de données (Data querying) :** Pour mettre en œuvre une procédure d'agrégation de données, une représentation de base de données est d'habitude employée. On voit ainsi la collection des données des capteurs comme une base de données complète, sur laquelle des opérations traditionnelles de gestion sont effectuées. Le plus approprié est l'interrogation de données, où les données sont demandées du réseau (accomplissant n'importe quels critères particuliers ou pas). Dans ce sens, il est plus approprié d'utiliser une sémantique de type SQL en décrivant les opérations d'agrégation de données exigées.

- **La collecte de données (Data gathering):** La collecte de données se réfère au processus d'envoi et le rassemblement des diverses mesures des nœuds capteur au sink. Par conséquent le terme de collecte se réfère surtout à la méthodologie de collection(ramassage), englobant ainsi les aspects de cheminement plutôt que se concentrer sur le processus d'élaboration de l'information elle-même.

- **La concaténation de données (Data concatenation):** La concaténation de données se réfère à l'élaboration à partir de plusieurs messages simples, un message plus grand (à condition que les unités de transfert maximales de la pile de protocole sous-jacente soient compatibles avec cela). Ceci implique la réduction du nombre de paquets à traiter, la consommation d'énergie et bien que d'autres effets secondaire (par exemple, la possibilité d'avoir des collisions).

- **La compression de données (Data compression):** se réfère à l'utilisation de techniques de compression pour réduire la quantité d'octets exigés pour coder les pièces différentes d'informations et, ainsi, la charge de trafic qui doit être traitée dans le réseau.

- **L'exploration ou extraction de données (Data mining):** Finalement, au sink, l'extraction de données se réfère au processus de choix des informations utiles de la quantité énorme de données rassemblées dans un réseau de capteur, les traiter et les filtrer.

Il faut mettre en évidence que l'utilisation de n'importe lequel des mécanismes mentionnés ci-dessus n'est pas exclusive et une combinaison d'entre eux pourrait être utilisée; en fait, ils sont normalement employés ensemble, comme les parties d'un même processus complet.

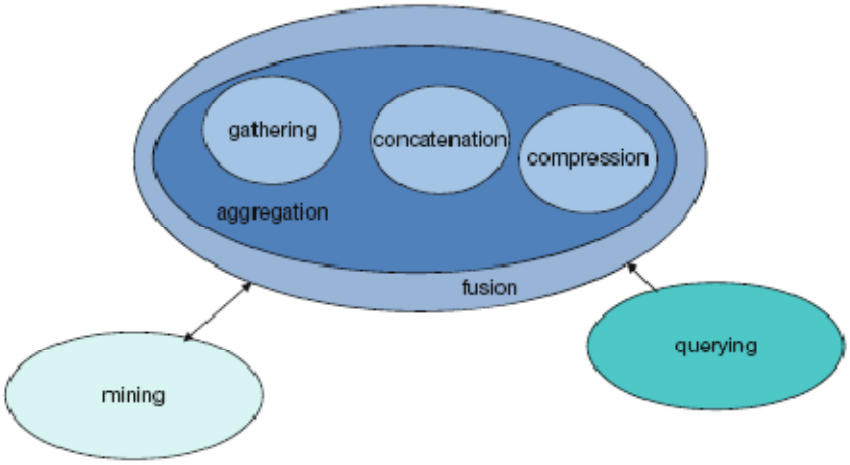


Figure 3.1 : la relation entre les terminologies de l'agrégation données [26].

III.5 Les couches utilisant cette technique :

Bien que l'agrégation de données soit une procédure complexe exigeant une action commune exécutée à travers les différentes couches de l'architecture de réseau, une première classification des techniques d'agrégation peut être faite. Plus spécifiquement, les techniques d'agrégation peuvent être distinguées sur la base de la couche architecturale spécifique principalement impliquée. En conséquence, nous pouvons distinguer une approche d'agrégation opérant sur la couche de liaison de données, d'un protocole opérant sur la couche réseau ou des couches supérieures. Cette classification peut faciliter la compréhension

des principaux éléments nécessaire pour effectuer l'agrégation à chaque couche et, d'autre part, peut mettre en évidence le besoin d'interactions de couche mutuelles.

En considérant les solutions d'agrégation proposées dans la littérature, nous pouvons principalement distinguer entre des approches travaillant spécifiquement à la couche liaison et les solutions impliquant une action de couche de réseau. Avec cela, peu de solutions impliquant une action effectuée à la couche application sont également apparu.

Quand l'agrégation de données opère dans la couche de liaison, les solutions proposées sont typiquement basée sur des paquets de données s'agrégeant au niveau de liaison en réduisant le nombre de nœuds luttant, en prenant en compte les corrélation de données possible, et en permettant seulement à un nœud d'envoyer des paquets de données chaque fois au compte d'autres nœuds. (au détriment)

Quand l'agrégation opéré dans la couche de réseau, il se focalise d'habitude sur le positionnement de nœuds agrégateur et la conception approprié des protocoles de routage qui permettent de forcer le trafic à passer par des nœuds agrégateur répartis sur tout le réseau.

Finalement, lorsque l'on considère la couche application, les techniques sont généralement liées à l'utilisation de la sémantique appropriée pour coder les données émises par les nœuds de capteur tout en envisageant aussi la possibilité d'exploiter la corrélation spatiale et temporelle des données.

III.6 Fonctionnement des agrégateurs :

La performance des procédures d'agrégation de données dépend de la manière dont les données sont collecter par les nœuds capteur, sont mises au point et propagées au puits. À ce but nous pouvons distinguer entre un processus d'agrégation de données qui réduit la quantité d'informations propagées partout dans le réseau ou non. Plus spécifiquement, quand l'agrégation est exécutée par quelques nœuds intermédiaires, les agrégateurs peuvent [26] :

- ***Combiner et compresser les paquets de données en préservant leurs parties de données utiles***. Cela signifie qu'il y a seulement une réduction de la taille d'en-tête tandis que les deux parties de données utiles sont fusionnés ensemble. Comme exemple, nous laissons supposer que le nœud 1 envoie une mesure de température, disons T1, au sink et le nœud 2 envoie la mesureT2. En conséquence le nœud 3, étant un agrégateur, émettra un paquet de données avec la partie donnée utile contenant T1

+ T2. Cette sorte d'agrégation est mentionnée comme l'agrégation sans perte puisqu'elle n'aboutit pas à des pertes d'informations. Ces résultats d'approche particulièrement sont utiles quand les lectures de données sont petites dans la taille en comparaison de la taille maximum de paquet de données permis et ainsi l'agrégation peuvent être efficacement exécutés en mêlant plus de paquets. Observez que cette approche est appropriée quand les lectures de capteur liées aux types différents de mesures sont reçues. Comme par exemple, si le nœud 1 envoie une mesure de température au skin et le nœud 2 envoie une mesure de pollution, les données ne peuvent pas être fusionnées, mais peuvent être insérées dans un paquet unique.

- ***Combiner et compresser les paquets de données en fusionnant leurs parties de données utiles.*** Dans ce cas, l'entête et la partie donnés utile du paquet sont réduits en mettant et compressant ensemble leur contenu. Comme par exemple, nous laissons supposer que le nœud 1 envoie une mesure de température, disons T1 au puits et le nœud 2 envoie la mesureT2. En conséquence le nœud 3, étant un agrégateur, émettra un paquet de données avec la partie donnée utile contenant les mesures T1 et T2 "fondu"("fusionné"). Le résultat du processus "de fusion" dépend de la fonction d'agrégation étant choisie. Comme un exemple, si la fonction d'agrégation est la MOYENNE, la moyenne des deux mesures est calculée et envoyée. En considérant cette sorte d'agrégation, il est évident qu'un processus de LOSSY (Définition du mot LOSSY, Se dit d'un logiciel de compression qui provoque des pertes de données pour parvenir à un résultat satisfaisant.) est arrivé puisque les valeurs initiales ne peuvent pas être récupérées au puits. Cependant, dans la majorité des applications de réseau de capteur, sachant que l'agrégation au puits est plus que suffisant que de garder une mise à jour de la vision du statut de réseau. Cette approche est utile en cas de l'homogénéité dans les données reçues par le nœud agrégateur.

III.7 Fonctions d'agrégat :

La performance de l'agrégation dépend aussi du type de fonction d'agrégation étant utilisée. La classification proposée dans [26] est concernée par la manière dont l'agrégation prend les mesures reçues pour produire des données agrégées ayant la valeur d'un échantillon représentatif. Les propriétés considérées pour la classification des ensembles sont :

- Sensible au doublon (duplicate sensitive) : où la sensibilité des agrégations à dupliquer les lectures étant reçues est considérer.
- Exemplaire (Exemplary) où une valeur représentative est produite à partir de toutes les valeurs, mais elle est imprévisible, ne donnant ainsi aucunes garanties sur la fiabilité de la valeur.
- Le résumé (Summary) où une valeur représentative est évaluée sur toutes les valeurs donnant ainsi une évaluation plus fiable qui provient de considérer une valeur plus stable dont la contribution est évalué sur toutes les valeurs.
- Monotonic où deux rapports partiels d'état sont tels que l'agrégation est toujours plus haut ou plus bas que les valeurs partielles d'état utilisées pour l'évaluer.

	Duplicate sensitive	Exemplary (E), Summary (S)	Monotonic	Partial state
Max, Min	No	E	Yes	Distributive
Count, Sum	Yes	S	Yes	Distributive
Average	Yes	S	No	Algebraic
Median	Yes	E	No	Holistic
Count Distinct	No	S	Yes	Unique
Histogram	Yes	S	No	Content-sensitive

Tableaux :fonctions d'agrégats [27]

Dans la Table 2, une autre dimension qui se rapporte du montant de l'état exigé pour chaque rapport d'état partiel. Par exemple, le rapport d'état partiel de AVERAGE consiste a une paire de valeurs, tandis qu'un rapport d'état partiel de COUNT constitue seulement une seule valeur.

L'état partiel des agrégations peut être dans un des états suivants :

Dans les agrégats *distributifs,* l'état partiel est simplement l'agrégation de la partition de données sur lesquelles ils sont calculés (l'ensemble des données partielles). D'où la taille des enregistrements partielle de l'Etat est le même que la taille de l'agrégat final.

Dans les agrégats *algébriques,* les rapports partielle de l'Etat sont de taille constante et ne sont pas des agrégations pour les partitions.

Dans les agrégats *holistiques,* les rapports d'état partiels sont proportionnels de la taille à l'ensemble des données dans la partition. Essentiellement pour les agrégations holistiques

aucune agrégation partielle utile ne peut être faite et toutes les données doivent être réconciliées (portées ensemble) pour être agrégé par l'évaluateur.

L'agrégat *unique* est semblable aux agrégats holistiques, à l'exception du fait que la somme d'état qui doit être propagé est proportionnel au nombre de valeurs distinctes dans la partition.

Contenu sensible où les rapports d'état partiels sont proportionnels dans la taille à une certaine propriété (par exemple propriété statistiques) des valeurs de données dans la partition.

III .8 Méthodes d'agrégation de données :

La performance du processus d'agrégation est aussi strictement rapprochée du taux d'agrégation, c'est-à-dire, le nombre de paquets étant agrégés par unité de temps. À ce but, trois méthodes d'agrégation peuvent être distinguées :

- **Périodique simple :** quand un nœud agrégateur rassemble toutes les lectures reçues pendant un temps fixe et réunit ensuite les données reçues sans aucune considération de n'importe quelles contraintes de temps. Cette approche, bien qu'elle soit simple, ne peut pas être très efficace quand seulement des paquets non significatifs sont reçus par l'agrégateur qui est alors forcé d'envoyer le paquet, même quand ce n'est pas nécessaire.

- **Périodique par saut :** Quand on considère un arbre de collection et un nœud agrégateur envoi le paquet fusionné seulement quand il a reçu un paquet de tous ses enfants. De nouveau cette approche est très simple, mais peut mener aux temps d'attente inefficaces quand quelques nœuds sont moins actifs que d'autres et envoient des données tout à fait rarement.

- **Périodique par saut ajusté :** Il utilise le même principe de base que périodiques par saut, mais la duré de temporisation d'un nœud est basée sur sa position dans l'arbre de distribution [28].

III.9 Modèle d'agrégation [29] :

III. 9.1 Modèle à un agrégateur :

Dans ce modèle, le processus d'agrégation ne s'effectue que dans un seul nœud entre les capteurs et la station de base puis l'utilisateur. De ce fait, cet agrégateur doit être assez puissant pour effectuer les calculs et les envois réguliers des agrégats. Ce modèle n'est pas très efficace en termes d'agrégation puisque le but premier de l'agrégation est d'éviter les communications ; or, un phénomène de redondance d'informations apparait sur les nœuds se trouvant entre la station de base et l'agrégateur (illustré sur la figure par les flèches à deux directions). Ce modèle n'est donc valable que pour les petits RCSF ou pour ceux ayant une station de base hors du réseau.

III. 9.2 Modèle à multiple agrégateur :

Dans ce cas, les données collectées sont agrégées plus d'une fois avant d'atteindre le demandeur. Ce modèle permet de réduire davantage les données transmises que le modèle à unique agrégateur, surtout dans les RCSF larges (comportant de nombreux nœuds) et notamment dans les réseaux où la redondance des données est élevée (comme expliquées dans le paragraphe précédent).

Les deux modèles peuvent présenter (ou non) une phase de vérification qui permet au demandeur de savoir si l'agrégat est valide ou non (ou de ne pas le savoir). Il est plus compliqué d'effectuer cette phase dans un modèle à multiples agrégateur car le demandeur doit savoir à quel nœud agrégateur les données ont été altérées. Si le modèle auquel on s'intéresse ne possède pas de phase de vérification, cela signifie que l'on ne s'intéresse pas à l'une des exigences de l'agrégation des données : l'intégrité des données.

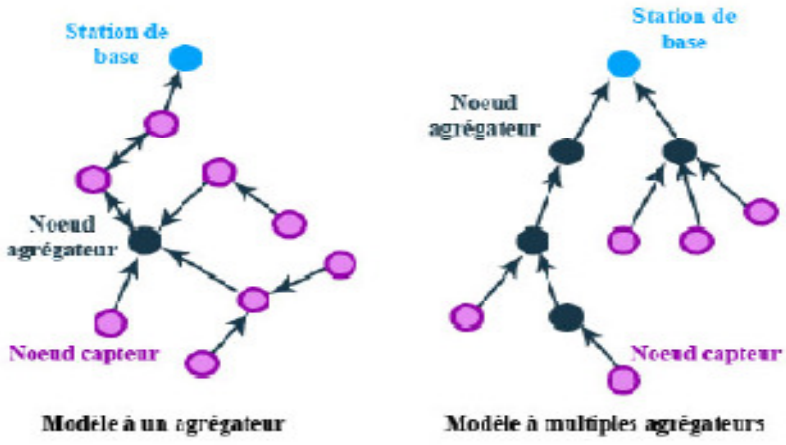


Figure 3.2 : Modèle d'agrégation [29]

III .10 Types d'agrégation de données dans les RCSFs :

Les techniques d'agrégation de données peuvent être découpées selon trois critères:

III .10.1 Agrégation basée sur l'architecture du réseau :

A)- Réseaux à plat :

Tous les nœuds jouent un rôle identique dans le réseau. La méthode d'agrégation de données est très liée au protocole de routage utilisé.

On peut alors distinguer deux modes d'agrégation de données :

- Push : les nœuds qui ont l'info qui initient sa transmission.
- Pull : les nœuds qui ont besoin de l'info qui en demandent la transmission.

1- Réseaux à plat : Mode Push :

Il définit des méta donnés spécifiques à l'application. Par exemple les capteurs d'une même zone géographique utilisent le même ID dans la méta donnée. Quand un nœud a une info il en avertit ses voisins par envoi de la méta donnée (**ADV**) .

Le ou les voisins intéressés par ce type d'info lui répondent par (**REQ**) qu'ils veulent recevoir cette info. Et le nœud la leur envoie (**DATA**). De plus chaque nœud tient le compte de sa consommation d'énergie et s'en sert pour décider d'envoyer ou pas des données. Ainsi certaines tâches peuvent ne pas être faites si l'énergie est trop faible [30].

2- Réseaux à plat : Mode Pull :

- **Les gradients** sont mis à jour sur chaque nœud qui a reçu la demande (de qui ça lui est venu) on peut aussi leur associer des infos de qualité, délai, erreurs de transmission avec ce voisin.

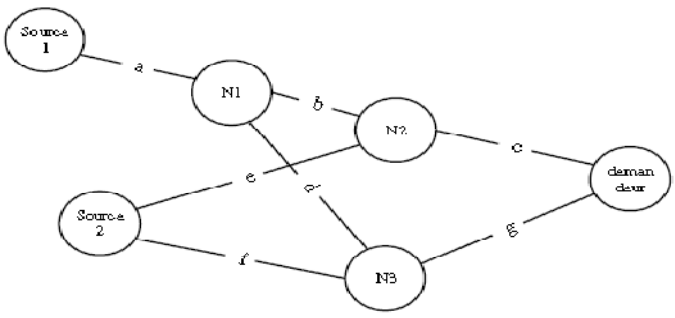
- **L'envoi de l'information** sera fait par les nœuds détenant (soit parce qu'il l'ont acquise, soit parce qu'il l'ont reçue) une information correspondant à lune des demandes qui sont dans leurs tables. Ce ne sont pas forcément les nœuds source de cette information, ce peuvent être des nœuds qui ont reçu cette info auparavant et l'ont gardée en cache.

Pour savoir s'ils doivent envoyer les nœuds tiennent compte des paramètres de la demande comme la périodicité. Ils utilisent les gradients liés à cette demande pour faire passer l'info (en fait comme ces gradients ont été établis lors de la demande ils constituent un chemin pour arriver au demandeur). Un nœud qui reçoit une telle information la fera passer en utilisant ses gradients mais il vérifiera qu'il ne l'a pas déjà fait (en tenant également compte de la périodicité) pour éviter les duplications liées au système de diffusion de la demande (on a plusieurs chemins et, a priori, on va les utiliser tous).

- **Le renforcement de chemins** peut être mis en place par un demandeur qui constate que l'info lui parvient plus vite par un chemin que par un autre. Les nœuds de ce chemin seront informés de faire plutôt passer l'info par tel voisin que par tel autre. De plus, lorsqu'il y a plusieurs sources d'info, le choix d'un chemin permettant l'agrégation de données est intéressante.

Remarque : Le protocole SPIN fonctionne avec cette logique.

Exemple [30] :



Dans ce dessin ci-contre prendre les chemins :

Source1 -> destination par a b c

Et Source 2 -> destination par f g:

Est moins intéressant que de prendre :

Source1 -> destination par a b c

Et Source 2 -> destination par e c

Car le nœud N2 peut faire l'agrégation des infos de source1 et de source2.

Il en serait de même en prenant :

Source1 -> destination par a d g

Et Source 2 -> destination par f g

Avec N3 qui fait l'agrégation.

B)- Réseaux hiérarchisés :

Lorsque l'on découpe un réseau en sous parties , chaque partie possède un nœud central qui peut faire l'agrégation de données pour sa partie.
Ensuite il fait passer l'information au demandeur par un chemin qui peut emprunter plusieurs autres nœuds centraux d'autres parties du réseau.

Remarque : les protocoles LEACH fonctionne avec cette logique.

1- Agrégation centralisé: (Agrégation par cluster)

- **Notion d'agrégation dans le cluster [31] :**

L'agrégation de nœuds en clusters permet de réduire la complexité des algorithmes de routage, d'optimiser la ressource medium en la faisant gérer localement par un chef de cluster

(le clusterhead aussi appelé caryomme), de faciliter l'agrégation des données, de simplifier la gestion du réseau et en particulier l'affectation d'adresses, d'optimiser les dépenses d'énergie, et enfin de rendre le réseau plus scalable. L'utilisation de clusters permet aussi de stabiliser la topologie et la gestion du réseau si les tailles de clusters sont grandes par rapport aux vitesses de nœuds mais cela ne fonctionne que dans le cas d'une faible mobilité.

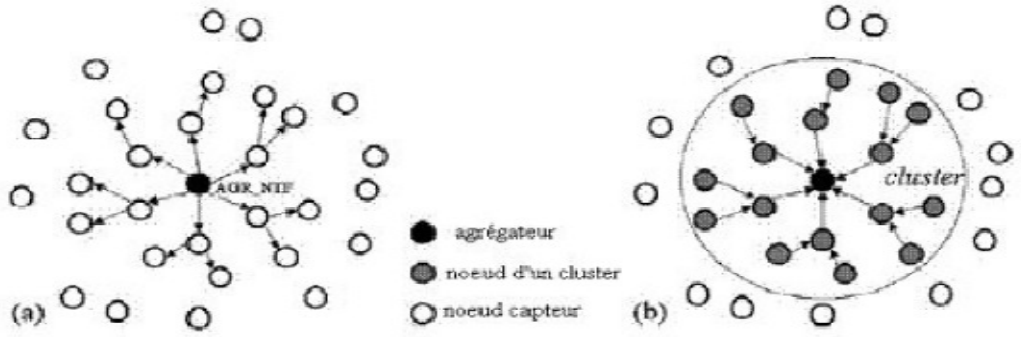


Figure 3.3 : **configuration de clusters[24]**

2- Agrégation par chaine :

Cette méthode se base sur l'établissement des liens entre nœuds sur le principe de ne communiquer qu'avec le nœud le plus proche (en utilisant la force du signal comme mesure). On constitue alors des chaînes de nœuds qui aboutissent au nœud central de la partie. Lorsque les données circulent sur cette chaîne chaque nœud qui les transmet peut y agréger ses propres donnés [30].

A la fin, le nœud central terminera l'agrégation avant d'envoyer au demandeur.

Remarque : le protocole PEGASIS fonctionne avec cette logique.

3- Agrégation distribué : (Agrégation par arbre)

Le principe est de constituer un arbre dont les feuilles sont les sources de l'information et la racine le demandeur. A chaque nœud de l'arbre on peut faire de l'agrégation de données au fur et à mesure que l'information remonte. Le principe de constitution de l'arbre est le suivant:

Le demandeur envoie un message à ses voisins [30] .

- Tout nœud qui reçoit un tel message pour la 1ère fois lance un timer. Pendant la durée de ce timer il reçoit d'autres messages de ce type d'autres voisins. Ces messages contiennent le nombre de saut depuis le demandeur et l'énergie disponible sur le nœud qui a envoyé ce message. Quand le délai est terminé, il choisit son père dans l'arbre parmi tous ceux qui lui ont envoyé ce message et le lui signale. Ce choix se fait sur le nombre de saut et l'énergie de l'émetteur.
- Enfin il diffuse à son tour à ces voisins un message du même type en augmentant de 1 le nombre de saut et en indiquant sa propre énergie.
- Chaque nœud sait alors s'il est nœud ou feuille de l'arbre en fonction du fait que, lorsqu'il a diffusé son message, il a reçu des réponses de nœuds l'ayant choisi comme père ou pas.
- Le processus continue et se termine naturellement lorsque tous les nœuds ont été atteints.

III .10.2 Agrégation basée sur les flux dans le réseau :

On part d'une vision du réseau sous forme de graphe. Chaque nœud est un nœud du réseau, chaque arc est une liaison directe possible.

- On value chaque arc par la quantité d'énergie consommée pour un paquet qui passe par cet arc.
- On cherche ensuite l'arbre permettant de transmettre les infos en en faisant l'agrégation qui consomme le moins d'énergie. Ce problème est en général NP complet => il faut trouver des heuristiques.

Ce type de solution suppose d'avoir une vision complète du réseau au moins en terme de connexions directes.

III .10.3 Agrégation basée sur la qualité de service :

Dans ce cas on s'intéresse plus à la qualité de service qu'à l'énergie. C'est-à-dire soit :

- Trouver le modèle d'agrégation qui collecte le plus de données pertinentes
- Trouver le modèle d'agrégation qui limite la congestion du réseau

Dans le premier cas on part du pourcentage (%) d'information que chaque nœud apporte à l'information finale. On pondère chaque transmission par l'énergie consommée. L'objectif est de trouver une route qui collecte le plus fort pourcentage et consomme le moins d'énergie.

Dans le second cas on utilise l'agrégation plutôt comme outil d'optimisation que pour constituer des informations complètes. En fait on agrège des informations lorsqu'elles passent par la même liaison (du nœud A vers le nœud B) ceci permet de regrouper plusieurs paquets et de les compresser pour gagner du temps et de l'énergie. Mais les paquets regroupés n'ont aucun lien entre eux sinon qu'ils passent au même endroit. C'est surtout bien si on a beaucoup de transmissions de paquets plutôt petits [30].

III .11 Protocoles d'agrégation :

Dans cette partie nous allons établir un état de l'art de quelques protocoles de routage on les classifiant selon l'architecture du réseau (Agrégation basée sur l'architecture du réseau)

III.11.1 Réseaux à plat : Nous allons d'écrire brièvement le protocole SPIN

SPIN (Sensor Protocol for Information via Negotiation)

Le protocole SPIN [32] 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 Gossiping.se protocole est proposée pour pallier au problème d'ignorance de ressources confronté dans la technique d'inondation en utilisant la négociation et l'adaptation aux ressources disponibles.

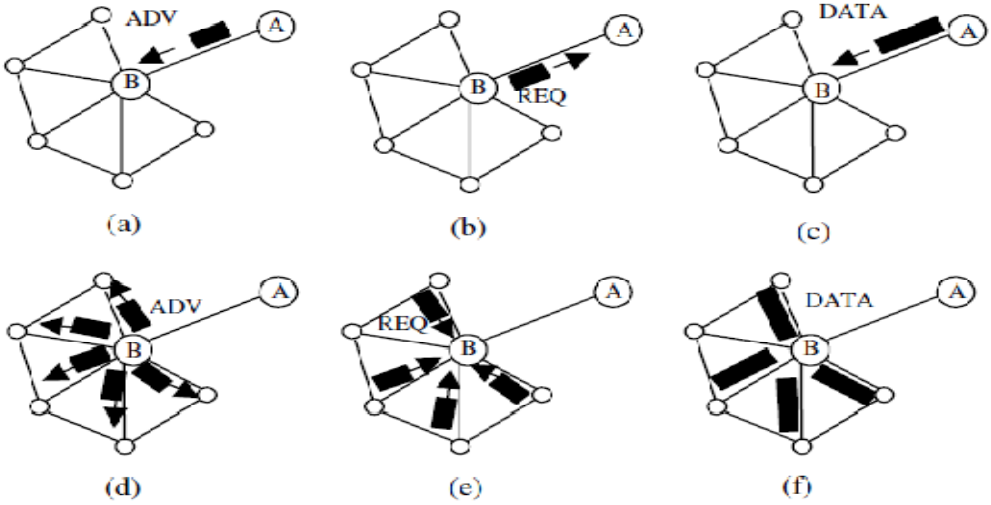


Figure 3.4 : Fonctionnement du protocole SPIN [33]

Le protocole SPIN utilise essentiellement trois types de paquets ADV/REQ/DATA. Un nœud voulant émettre une donnée commence par envoyer un paquet ADV. Ce paquet ADV consiste d'une méta-données sur les données à émettre. Les méta-données peuvent décrire plusieurs aspects comme le type des données et la localisation de son origine. Les nœuds qui reçoivent ce paquet vérifient si les données les intéressent. Si oui, ils répondent par un paquet REQ. Le nœud qui a initié la communication envoie alors un paquet DATA pour chaque réponse REQ reçue (voir la Figure 3.4). Un nœud peut parfaitement ne pas répondre aux messages ADV, par exemple dans le but d'économiser son énergie. Ensuite chaque nœud qui fait office de relais peut très bien agréger ses propres données aux données qui sont déjà contenues dans le paquet [34].

III.11.2 Agrégation centralisé : (hiérarchisé : agrégation par le cluster)

Dans ce type nous allons nous contenter des ses quatre protocoles suivant :

1- LEACH: Low-Energy Adaptive Clustering Hierarchy:

LEACH est un protocole de routage hiérarchique, employant un procédé de clustering qui divise le réseau en deux niveaux : les cluster-heads et les nœuds membres. Le protocole se déroule en rounds. Chaque round se compose de deux phases : construction et communication.

Nous allons détailler se protocole dans le chapitre suivant.

2- TEEN (Threshold sensitive Energy Efficient sensor Network protocol):

Manjeshwar et Agrawal [35] 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 nœuds 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, 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 nœud 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 nœuds écoutent le médium continûment et lorsque la valeur captée du paramètre contrôlé dépasse le seuil Hard, le nœud transmet les données i.e. un changement brusque d'un certain paramètre est survenu. La valeur captée est stockée dans une variable interne appelée SV.

Puis, les nœuds 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 important 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 nœuds ne communiqueront jamais, et aucune donnée ne sera transmise à l'utilisateur, ainsi la station de base ne connaît pas les nœuds qui ont épuisés leur énergie.

TEEN n'est pas souhaitable pour les applications qui nécessitent des envois périodiques de données.

Pour remédier à ces limitations, les auteurs ont proposé une extension de TEEN appelée APTEEN (Adaptive Threshold-sensitive Energy Efficient sensor Network protocol).

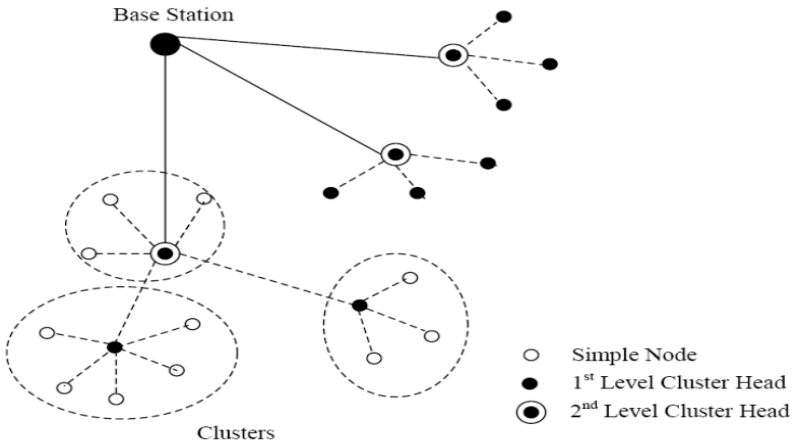


Figure 3.5 : le protocole TEEN

3- APTEEN : (Adaptive Threshold-sensitive Energy Efficient sensor Network protocol)

Est un protocole hybride qui change la périodicité et les valeurs seuils utilisées dans TEEN selon les besoins de l'utilisateur et le type d'application. Dans APTEEN [3.14], les cluster-heads transmettent à leurs membres les paramètres suivants :

– l'ensemble de paramètres physiques auxquels l'utilisateur est intéressé pour obtenir des informations (A).
– Les seuils : seuil Hard HT et seuil Soft ST.
– Un Schedule TDMA permettant d'assigner à chaque nœud un intervalle fini de temps appelé slot.
– Un compteur de temps (CT) : c'est la période de temps maximum entre deux transmissions successives d'un nœud.

Dans APTEEN, les nœuds surveillent en continu l'environnement. Ainsi, les nœuds qui détectent une valeur d'un paramètre qui dépasse le seuil HT, transmettent leurs données. Une fois qu'un nœud détecte une valeur qui dépasse HT, il ne transmet les données au cluster-head que si la valeur de ce paramètre change d'une quantité égale ou plus supérieure à ST. Si un nœud ne transmet par de données pendant une période de temps CT, il devrait faire une capture de données et les retransmettre.

APTEEN offre une grande flexibilité qui permet à l'utilisateur de choisir l'intervalle de temps CT, et les valeurs seuils HT et ST pour que la consommation d'énergie soit contrôlée par la variation de ces paramètres. Cependant, APTEEN nécessite une complexité supplémentaire pour implémenter les fonctions de seuils et de périodes de temps CT. Ainsi, l'overhead et la complexité associés à la formation des clusters à plusieurs niveaux par TEEN et APTEEN sont assez élevés.

III.11.3 Agrégation par chaine :

1- PEGASIS (Power-Efficient Gathering in Sensor Information Systems) :

Version améliorée de LEACH ça principale idée est de former une chaîne entre les nœuds de sorte que chaque nœud reçoive de et communique à un voisin proche. Les données collectées sont transmises d'un nœud à un autre qui les agrège jusqu'à ce qu'elles arrivent à un nœud particulier qui les transmet à la station de base. Les nœuds qui transmettent les données à la station de base, sont choisis tour à tour.

Contrairement à LEACH, PEGASIS évite la formation des clusters et procure à un seul nœud dans la chaîne l'envoi de données à la station de base. D'ailleurs, PEGASIS suppose que les nœuds sont capables de modifier leur puissance de transmission.

Les résultats de simulation ont montré que PEGASIS peut prolonger de deux à trois fois la durée de vie d'un réseau de capteurs relativement à LEACH en fonction du critère choisi pour évaluer la durée de vie d'un réseau i.e. quand 1%, 20%, 50% ou 100% des nœuds épuisent leurs batteries. Un tel gain de performance est réalisé par l'élimination du surcoût causé par le processus de formation de clusters dans LEACH, et par la réduction du nombre de transmissions et de réceptions en agrégeant de données. Bien que le surcoût du clustering soit évité, PEGASIS exige toujours un ajustement dynamique de la topologie puisqu'un nœud devrait connaître le niveau d'énergie de ses voisins avant de relayer ses données. Cependant, un tel ajustement de la topologie pourrait causer un surcoût important en particulier dans les réseaux les plus utilisés.

En outre, PEGASIS suppose que tout nœud communique directement avec la station de base qui gère la topologie d'une manière centralisée. Or, cette supposition est loin de la réalité car les capteurs communiquent généralement en mode multi-sauts pour atteindre la station de base. D'autre part, PEGASIS suppose que tous les nœuds maintiennent une table contenant les localisations de tous les autres nœuds dans le réseau. En résumé, PEGASIS est adapté seulement aux capteurs sans fil dont les nœuds sont immobiles. Son évaluation dans des environnements mobiles pourrait dégrader considérablement ses performances.

Une variante de PEGASIS appelée Hierarchical PEGASIS [37] a été conçue afin d'améliorer PEGASIS. Dans Hierarchical PEGASIS, la chaîne est divisée en groupes de la sorte que chaque nœud communique avec un seul nœud voisin de niveau plus bas de la hiérarchie. Les transmissions simultanées en parallèle dans des groupes différents minimisent le délai de transmission. Un autre protocole similaire à PEGASIS, appelé C2E2S, a été proposé dans [38]. Il est basé sur les clusters et les chaînes. C'est un protocole centralisé où la station de base organise le réseau en se basant sur l'information de l'énergie des nœuds.

III.11.4 Agrégation distribuée : (hiérarchisée : agrégation par arbre)

Dans les paragraphes suivants, nous discuterons sur quelques protocoles arbres.

1- COUGAR :

C'est un protocole distribué qui considère le réseau comme un énorme système de base de données répartie. L'idée principale est d'employer des requêtes déclaratives afin d'abstraire le traitement des requêtes des fonctions de la couche réseau et d'utiliser l'agrégation de données pour économiser de l'énergie. L'abstraction est soutenue par une nouvelle couche de requête entre les couches réseau et application.

Dans COUGAR, 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 cluster head ou bien des données produites par ce cluster. L'approche COUGAR fournit une agrégation partielle au niveau des cluster head . Chaque cluster head maintient une liste d'attente contenant les cluster head fils qui doivent lui envoyer les paquets. Le cluster head n'émet le paquet agrégé au prochain saut que s'il a reçu les paquets de tous les cluster-heads de la liste d'attente. Cependant, il 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 [39].

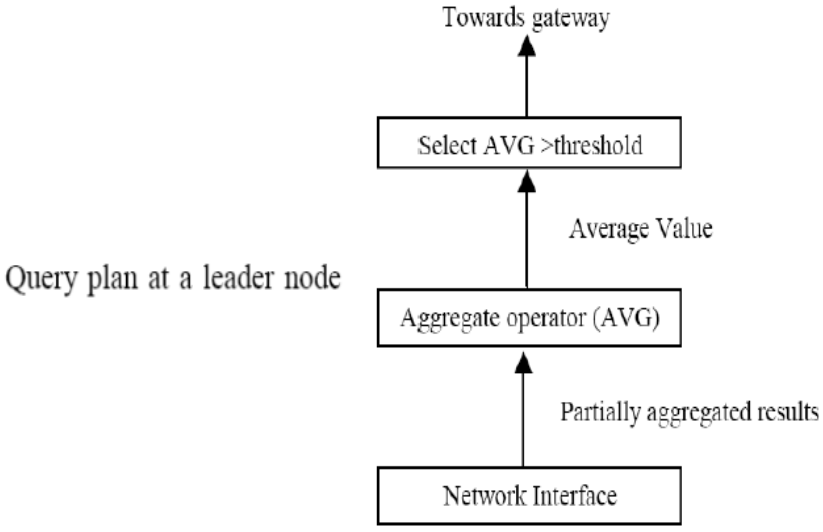


Figure 3.6 : fonctionnement de COUGAR

Inconvénient :

- Une couche de requête traditionnelle apporte une charge supplémentaire en termes de consommation d'énergie et stockage.
- Le traitement de données dans le réseau nécessite une synchronisation (attendre toutes les données avant de réaliser le calcul.
- Maintenance dynamique des nœuds leader pour éviter les failles.

2- TAG: (TINY AGgregation)

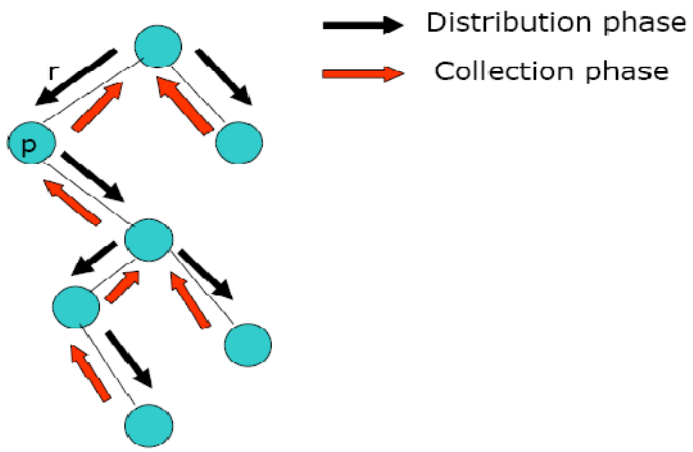


Figure 3.7 : Le protocole TAG [40]

Un service d'agrégation minuscule pour les réseaux de capteurs , c'est le tout premier algorithme proposé par Madden et al en 2004 et plus efficace en termes d'énergie. L'arbre est construit par le nœud racine qui envoie le message diffusé par le niveau 0 et son ID du capteur. Tous les nœuds entendant le message augmentent le champ de niveau, attachent leur identifiant et rediffusent à nouveau. Ils sélectionnent la même source du message que leur parent. Le processus se poursuit vers le bas de l'arbre. Il peut être utilisé comme un suivi périodique de la requête entraînée.

En mode surveillance périodique, la racine envoie la requête. Les nœuds enfants envoient leurs valeurs actuelles agrégées et rediffuser la requête au niveau suivant. Maintenant la racine reçoit les informations des nœuds enfants du premier niveau. Ce processus continu jusqu'à ou la racine reçoit les paquets du dernier niveau.

Donc TAG est constitué de deux phases: une phase de *distribution,* dans laquelle requêtes d'agrégation sont poussés vers le bas dans le réseau, et une phase de *collecte,* où

les valeurs agrégées sont continuellement acheminés en haut des enfants jusqu'aux parents. Rappelons que la requête sémantique partitionne le temps en époques de durée, et qu'il doit produire une seule valeur globale (lorsqu'il n'y a pas de groupement) qui combine les lectures de tous les capteurs du réseau pendant cette époque.

Étant donné le but d'utiliser peu de messages possible, la phase de collection doit assurer que les parents dans l'arbre de cheminement attendent jusqu'à ce qu'ils aient reçu des nouvelles de leurs enfants avant la propagation d'une valeur agrégé pour l'époque actuelle.

Cela s'accomplis en ayant des parents qui subdivisent l'époque telle que les enfants sont obligés de livrer leurs rapports d'état partiels pendant un intervalle de temps spécifié par le parent. Cet intervalle est choisi tel qu'il y a assez de temps pour le parent pour combiner des rapports d'état partiels et propager son propre rapport à son parent [41].

Quand un capteur p reçoit une demande (requête) d'agrégation, par un autre capteur ou un utilisateur, il se réveille, synchronises son horloge selon les informations de synchronisation dans le message et se prépare à participer dans l'agrégation.

Dans le schéma de routage basée sur l'arborescence, p choisit l'expéditeur du message comme son parent. En plus des informations dans la requête, r rajoutes l'intervalle quand l'expéditeur s'attend à entendre des rapports d'état partiels de p. p transmet ensuite la demande de requête r en bas du réseau, l'inclusion de cette intervalle de livraison pour que les enfants serons légèrement avant le temps que leurs parent s'attend à recevoir le rapport d'état partiel de p.

Pendant l'époque après la propagation de la requête, chaque capteur écoute s'il y a des messages de la part de ses enfants pendant l'intervalle qu'il a spécifié en expédiant la requête. Il calcule alors un rapport d'état partiel constitué par la combinaison de toutes les valeurs des enfants, qu'il a entendu avec ses propres lectures de sonde locale.

Finalement, pendant l'intervalle de transmission demandé par son parent, le capteur transmet ce rapport d'état partiel en haut de réseau.

Le TAG offre beaucoup d'avantages: économise l'énergie, minimise le nombre de transférer des messages, son utilisation permet aux nœuds de dormir pendant les temps morts économisant ainsi l'énergie [42].

3- Tina: Temporal coherence Conscious In-Network of Aggregation

L'approche consiste à envoyer les données seulement quand il ya un changement significatif dans la valeur des données dans les lectures adjacentes au fil du temps. Le concept de l'époque comme en TAG est également utilisé ici pour synchroniser la réception des paquets de nœuds enfants et l'envoi de l'agrégat. Une valeur des données peuvent être ignorés si la variation de la valeur précédente est dans la plage spécifiée appelés filtres. Cela nécessite des besoins en mémoire plus élevés pour chaque nœud, car ils ont besoin pour stocker les résultats intermédiaires des nœuds enfants, (agrégation partielle). L'avantage est la réduction significative du nombre de messages par rapport au tag [43].

4- **DQEB-Dynamic Query-Tree Protocol équilibrage de l'énergie Le DQEB** [44]

C'est un protocole qui se repose sur l'approche d'équilibrage énergétique en modifiant dynamiquement l'arborescence on se basant sur l'énergie dépensé au niveau des nœuds.

Dans cette approche, les nœuds sont organisés en grappes avec des clusters head. Chaque nœud se voit attribuer un poids qui continu d'augmenté avec la diminution de la durée de vie ou de l'énergie.

Quant l'énergie diminue, il est plus sage de remplacer le cluster head par un nœud du bas de l'arbre, et le remettre à un nœud feuille afin que l'arbre ne soit pas déconnecté. Le coût de l'énergie dépend du nombre de feuilles et de nœuds non feuilles et de l'énergie restante au niveau du nœud. Les nœuds non feuille sont souvent engagés dans la transmission et la réception de plus données en comparaison des nœuds feuille. Lorsque le poids d'un nœud non feuille descend d'un seuil, le nœud coordinateur demande à tous ses enfants pour alterner les parents, puis en utilisant une approche gourmande elle sélectionne les parents en alternance pour tous ses enfants et lui-même devient un nœud feuille.

Depuis le nœud avec moins d'énergie est devenue le nœud feuille, il va vivre un peu plus puisque il ne dispose que d'envoyer ses données. Cela augmente la durée de vie.

III.12 Sécurité de l'agrégation de données

III.12.1 Problématique de la sécurité dans l'agrégation de données

L'agrégation de données pose évidemment des problèmes liés à la sécurité dans les réseaux de capteurs. Ces problématiques de sécurité sont les mêmes que dans tous les systèmes informatiques à savoir :

- **Confidentialité des données :** se doit d'être respectée pour certaines applications (pas forcément nécessaire pour l'agriculture mais dans les domaines du transport ou le domaine médical c'est une autre histoire...).

- **Intégrité des données :** les données ne doivent pas avoir été modifiées (par accident = erreur de transmission ou intentionnellement = attaque)

- **Disponibilité des données :** technique de répartition de la consommation d'énergie (par exemple le fait que les chefs de zone changent régulièrement), diagnostic régulier du réseau afin de réagir si un nœud est compromis, etc...

- **Authentification :** permet de vérifier l'identité d'un envoyeur de données sur le réseau afin d'éviter l'injection de données par un tiers non autorisé.

- **Non-répudiation :** permet de s'assurer que le message a été émis et reçu par la personne qui a déclaré l'avoir fait. Un nœud qui envoie le résultat de ses calculs d'agrégation ne peut pas nier l'avoir envoyé (permet à la station de base de déterminer d'où vient l'erreur si erreur il y a)

III.12.2 Les principales formes d'attaques

Différentes sortes d'attaques sont potentiellement en mesure de mettre en péril l'utilisation d'un réseau de capteurs :

- **Attaque par déni de service (DoS)** : si on envoie des signaux radio pour brouiller (jamming) la transmission entre les différents nœuds du réseau (sur la même fréquence de communication), on rend indisponible le RCSF car les nœuds sont incapables de communiquer.

- **Compromission d'un nœud** : on peut avoir physiquement accès aux capteurs du fait de leur dispersion en pleine nature bien souvent. Un attaquant peut donc éventuellement extraire des informations de ces capteurs puisqu'ils sont en sa possession physique.

- **Attaque sibylline** : l'attaquant se fait passer pour plusieurs capteurs différents afin de fausser les données et les résultats issus de ces mêmes données.

- **Attaque par routage sélectif** : l'attaquant s'occupe manuellement du routage sur un des nœuds du réseau et peut donc choisir s'il route les données ou non, quelle sera la destination des nœuds routés, filtrer certaines données, etc... Il modifie donc l'agrégat qu'aurait du envoyer le nœud compromis et donc fausse les résultats.

- **Attaque par répétition** : l'attaquant écoute et enregistre le trafic réseau (sniffing) dans le RCSF puis réinjecte les anciennes données plus tard dans le réseau, ce qui peut fausser le résultat de l'agrégation de données.

- **Attaque du skin** : l'attaquant compromet un nœud du réseau et le rend le plus attractif possible afin que les paquets lui soient envoyés avant d'être envoyées vers le puits. L'attaquant peut ensuite rendre plus efficaces d'autres attaques car il faussera d'autant plus les résultats du fait que toutes les données transitent par lui.

- **Attaque furtive** : l'attaquant injecte de fausses données dans le réseau sans révéler son existence (pas de compromission d'un nœud), ce qui revient à injecter un faux agrégat et fausser les résultats.

Il semble évident que cette liste n'est pas exhaustive, mais elle présente les principales formes d'attaques menées contre les réseaux de capteurs sans fil en terme d'agrégation de données.

III .13 Conclusion:

L'agrégation de données dans les réseaux de capteur sans fil est une approche prometteuse pour neutraliser la perte de ressources causées par la redondance voyageant partout dans le réseau.

Dans ce chapitre nous avons montré que l'utilisation d'agrégation de données est un moyen pour neutraliser ces problèmes de redondance dans les réseaux de capteur. Nous avons établis une étude détaillée de l'agrégation de données, qui a permis d'expliciter ses avantages et ses inconvénients, nous avons d'écris les technique d'agrégation de données, nous avons établis par la suite un état de l'art de quelques protocoles utilisant cette technique et en dernier nous avons expliqué la problématique de sécurité dans cette technique.

Dans le chapitre prochain nous allons étudier en détaille le fonctionnement de l'un des protocoles s'appuyant sur cette technique d'agrégation que nous avons cité précédemment intituler le protocole LEACH.

CHAPITRE IV :
FONCTIONNEMENT DE LEACH

IV.1 Introduction :

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

LEACH est considéré comme étant le premier protocole de routage hiérarchique utilisant l'approche basée sur les groupes. Il est aussi l'un des algorithmes de routage hiérarchiques les plus populaires pour les RCSF, proposés dans le cadre du projet μAMPS [45]. Il combine l'efficacité en consommation d'énergie et la qualité de l'accès au média, et ce en se basant sur le découpage en groupes, en vu de permettre l'utilisation du concept de l'agrégation de données pour une meilleure performance en termes de durée de vie.

Dans ce chapitre le but principal et d'expliquer le fonctionnement du protocole LEACH, pour cela nous voir l'architecture de communication utilisée par ce protocole ensuite, nous allons entamer les différentes phases de son algorithme et nous terminerons par ces avantages et inconvénients.

IV.2 Protocoles MAC utilisés par LEACH :

Pendant son fonctionnement, le protocole LEACH appelle certains schémas des protocoles MAC qui seront détaillés dans cette section pour mieux comprendre son déroulement. Les nœuds doivent avoir une certaine capacité de calcul pour supporter différents protocoles MAC. Comme les RCSF ont des caractéristiques distinctes de tout autre type de réseaux sans fil, les protocoles MAC conçus pour ces derniers ne sont pas toujours applicables dans les RCSF. Deux versions des protocoles MAC pour l'accès au media sont alors proposées pour les RCSF : l'accès aléatoire et l'allocation fixe [46].

IV.2.1 Accès aléatoire :

Les schémas à accès aléatoire sont à base de contention. Dans ces schémas, les nœuds qui possèdent des données à transmettre doivent essayer d'obtenir l'autorisation pour l'accès au media tout en réduisant les collisions avec les transmissions des données des autres nœuds.
Le schéma d'accès multiple avec surveillance de porteuse CSMA (Carrier Sense Multiple Access) sur lequel se base le protocole LEACH est l'un des schémas d'accès aléatoire [47].

Lorsqu'un nœud veut transmettre un message, il examine le média pour vérifier s'il est libre ou occupé par un autre nœud. Dans le cas où le media est libre, ce nœud pourra émettre son message afin d'éviter les collisions. Cela dit, des nœuds peuvent émettre des données en même temps, ce qui mène à des collisions. Il est nécessaire donc que celles-ci soient détectées et que la récupération de données soit effectuée et que ces données soient retransmises. Si les retransmissions se passent encore en même temps, d'autres collisions vont se produire. Une solution à ce problème consiste à introduire que chaque nœud attende un délai aléatoire avant de retransmettre ses données, ce qui réduit la probabilité d'une autre collision. [48]

IV.2.2 Allocation fixe :

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

Étant donné que chaque nœud est attribué en exclusivité à un intervalle, il n'y a presque pas de collisions entre les données. Toutefois, les schémas à allocation fixe s'avèrent inefficaces lorsque tous les nœuds n'ont pas de données à transmettre. En effet, ces intervalles sont affectés à des nœuds qui n'ont pas besoin de les utiliser. [49]

IV.2.2.1 TDMA :

Le schéma d'accès multiple à répartition de temps ou TDMA (Time Division Multiple Access) permet de diviser le temps en intervalles (time-slot) attribués à chaque nœud (voir figure 4.1). Ainsi, un seul nœud a le droit d'accès au canal (il utilise toute la plage de la bande passante du canal), mais doit émettre ses données pendant les intervalles de temps qui lui sont accordés. [48]

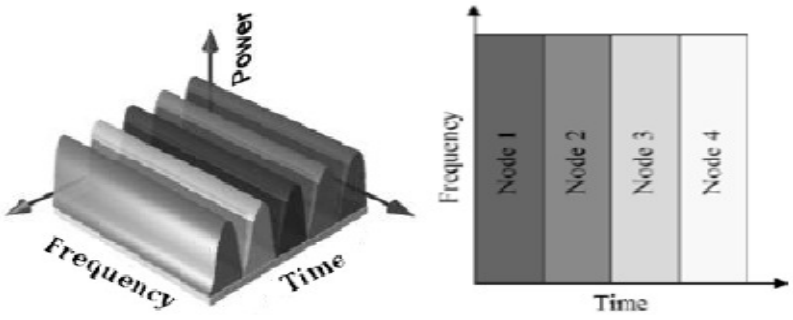


Figure 4.1 : **Diagrammes représentant le protocole MAC TDMA**. [50]

IV.2.2.2 CDMA :

Le schéma d'accès multiple par répartition en code ou CDMA (Code Division Multiple Access) permet de côtoyer plusieurs nœuds simultanément (voir figure). En effet, il ne divise ni la plage de fréquences ni l'intervalle de temps. Ainsi, des nœuds peuvent émettre leurs données continuellement et selon une large plage de fréquence. Le protocole CDMA utilise des techniques d'étalement de spectre afin d'éviter les collisions entre les transmissions simultanées des nœuds. [50]

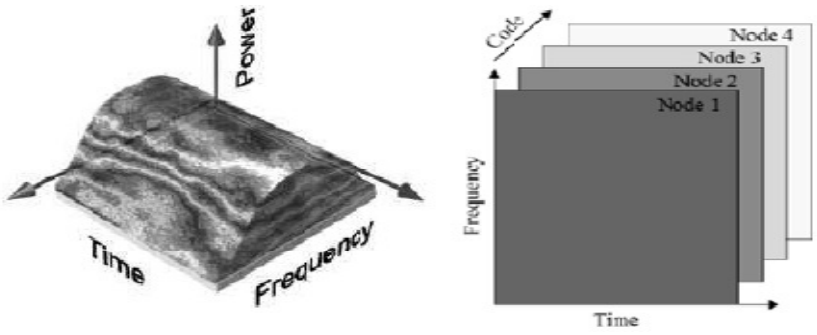


Figure 4.2 : Diagrammes représentant le protocole MAC CDMA. [50]

IV.3 Architecture de communication de LEACH :

L'architecture de communication de LEACH consiste, de façon similaire aux réseaux cellulaires, à former des cellules basées sur l'amplitude du signal, et utiliser les têtes de cellules comme routeurs vers le nœud puits. Ces cellules sont appelées groupes (clusters), quant aux têtes : chefs de groupes (cluster-heads CH). Les chefs de groupes sont choisis de façon aléatoire selon un algorithme spécifique d'élection basé sur une fonction de probabilité qui prend en compte différents critères comme l'énergie disponible des nœuds.

Comme la figure l'indique, les nœuds sont chargés de collecter des données, les envoyer à leurs CH qui les agrègent et transmettent, à leur tour, les résultats d'agrégation au nœud puits selon une communication unicast (à un seul saut).

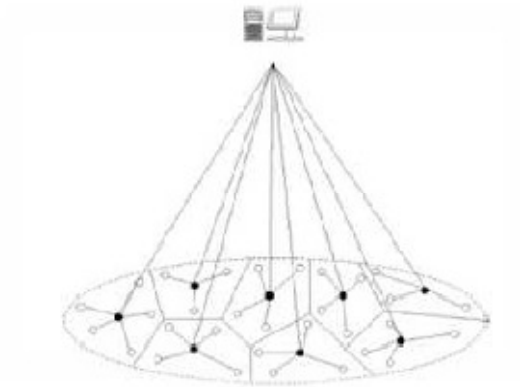


Figure 4.3 : Architecture de communication du protocole LEACH. [51]

Les CH ont pour mission d'assurer les fonctions les plus coûteuses en énergie, à savoir la communication avec le nœud puits qui est supposé éloigné, ainsi que tous les traitements de données (agrégation, fusion et transmission de données) afin de réduire la quantité des données transmises. Ce dispositif permet d'économiser l'énergie puisque les transmissions sont uniquement assurées par les CH plutôt que par tous les nœuds du réseau. Par conséquent, LEACH réalise une réduction significative de la dissipation d'énergie [52].

IV.4 Algorithme détaillé de LEACH :

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

IV.4.1 Phase d'initialisation :

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

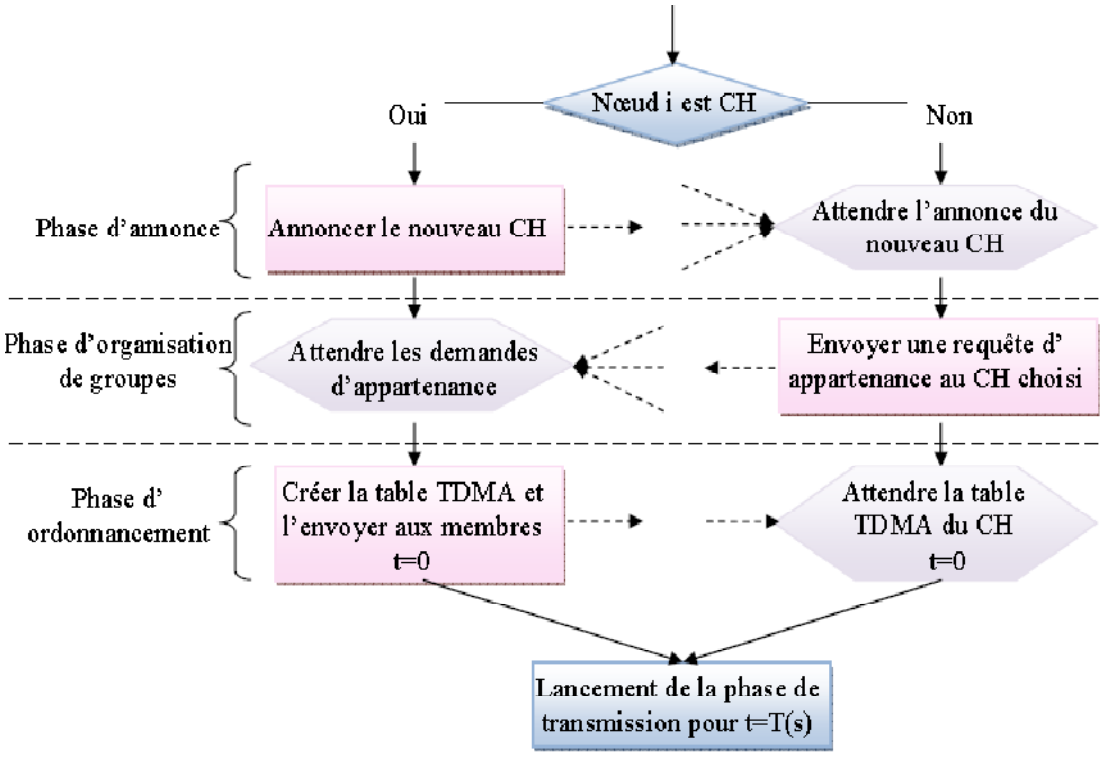


Figure 4.4 : Opérations de l'étape d'initialisation de LEACH [53].

IV.4.1.1 Phase d'annonce :

Avant de lancer cette phase, on désire avoir un certain nombre de CH. Ce nombre, que l'on note K, est fixe et il est inchangé durant tous les rounds. On estime que le pourcentage optimal du nombre de CH désirés devrait être de 5% à 15% du nombre total de nœuds [53].
Si ce pourcentage n'est pas respecté, cela mènera à une grande dissipation d'énergie dans le réseau. En effet, si le nombre de CH est très élevé, on aura un nombre important de nœuds(CH) qui se consacrent aux tâches très couteuses en ressources énergétiques. Ainsi, on aura une dissipation d'énergie considérable dans le réseau.

De plus, si le nombre de CH est très petit, ces derniers vont gérer des groupes de grandes tailles. Ainsi, ces CH s'épuiseront rapidement à cause de travail important qui leur est demandés.

Cette phase commence par l'annonce du nouveau round par le nœud puits, et, par la prise de décision locale d'un nœud pour devenir CH avec une certaine probabilité Pi(t) au début du round r+1 qui commence à l'instant t. Chaque nœud i génère un nombre aléatoire entre 0 et 1. Si ce nombre est inférieur à Pi(t), le nœud deviendra CH durant le round r+1. Pi(t) est calculé en fonction de K et de round r [54]:

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

Où N est le nombre total de nœuds dans le réseau. Si on a N nœuds et K CH, alors, il faudra N/K rounds durant lesquels un nœud doit être élu seulement une seule fois autant que CH avant que le round soit réinitialisé à 0.
Donc la probabilité de devenir CH pour chaque nœud i est :

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

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

Où Ci(t) égal à 0 si le nœud i a déjà été CH durant l'un des (r **mod** N/K) rounds précédents, et, il est égal à 1 dans le cas contraire. Donc, seuls les nœuds qui n'ont pas encore été CH, ont vraisemblablement une énergie résiduelle suffisante que les autres et ils pourront être choisis.
Le terme $\Sigma^{N}_{i=1}$ = Ci(t) représente le nombre total des nœuds éligibles d'être CH à l'instant t. Il est égal à :

$\Sigma^{N}_{i=1}$ Ci(t)= N-K*(r **mod** N/K).....(2)

Utilisant l'équation (1) et (2), le nombre de CH par round est :

$$\text{Nombre (CH)} = \Sigma^{N}_{i=1} Pi(t)*Ci(t) = (K*(r \bmod N/K))* \left(\frac{K}{N-K*(r \bmod N/K)}\right)$$

La probabilité Pi(t) est basée sur la supposition que tous les nœuds sont initialement homogènes et commencent avec la même quantité résiduelle d'énergie et meurent approximativement en même temps. Cependant, ceci pourrait être le cas juste après le déploiement, mais il n'est pas réellement valable après un certain temps. Alors, si l'énergie des nœuds diffère, il sera plus pratique que la probabilité Pi(t) soit en rapport avec l'énergie restante au niveau de chaque nœud. Cette probabilité sera donc égale à :

$$Pi(t) = \frac{Ei(t)}{Etotal(t)} K \qquad \dots (3)$$

Où Ei(t) est l'énergie résiduelle relative à chaque nœud i. Utilisant cette probabilité, le nœud avec une plus grande ressource d'énergie a une plus grande chance de devenir CH. Ainsi, le nombre de nœuds souhaités pour être CH dans chaque round est:

$$Nombre(CH) = \sum_{i=1}^{N} Pi(t) * Ci(t) = \left(\frac{E1(t)}{Etotal(t)} + \frac{E2(t)}{Etotal(t)} + \cdots \frac{En(t)}{Etotal(t)}\right) K = K$$

Les équations (2) et (3) seront égales si les nœuds commencent avec la même énergie. De plus, en utilisant l'équation (3), les nœuds requièrent des informations sur toute l'énergie disponible dans le réseau.

IV.4.1.2 Phase d'organisation de groupes :

Après qu'un nœud soit élu CH, il doit informer les autres nœuds non-CH de son nouveau rang dans le round courant. Pour cela, un message d'avertissement ADV contenant l'identificateur du CH est diffusé à tous les nœuds non-CH en utilisant le protocole MAC CSMA pour éviter les collisions entre les CH. La diffusion permet de s'assurer que tous les nœuds non-CH ont reçu le message. Par ailleurs, elle permet de garantir que les nœuds appartiennent au CH qui requière le minimum d'énergie pour la communication. La décision est basée donc sur l'amplitude du signal reçu; le CH ayant le signal le plus fort (i.e. le plus proche) sera choisi. En cas d'égalité des signaux, les nœuds non-CH choisissent aléatoirement leur CH [54].

Chaque membre informe son CH de sa décision. Une fois que le CH ait reçu la demande, il lui envoie un message d'acquittement Join- REQ.

IV.4.1.3 Phase d'ordonnancement :

Après la formation des groupes, chaque CH agit comme un centre de commande local pour coordonner les transmissions des données au sein de son groupe. Il crée un ordonnanceur (schedule) TDMA et assigne à chaque nœud membre un slot de temps durant lequel il peut transmettre ses données. L'ensemble des slots assignés aux nœuds d'un groupe est appelé frame. La durée de chaque frame diffère selon le nombre de membres du groupe.

Par ailleurs, afin de minimiser les interférences entre les transmissions dans des groupes adjacents, chaque CH choisit aléatoirement un code dans une liste de codes de propagation CDMA. Il le transmet par la suite à ses membres afin de l'utiliser pour leurs transmissions. [55]

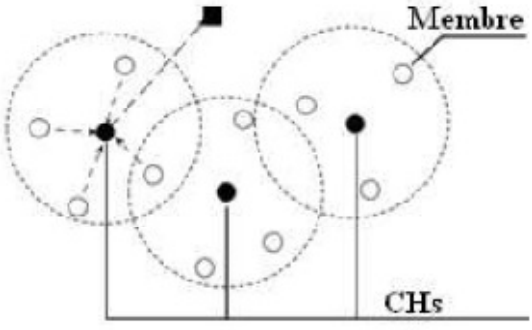


Figure 4.5 : Interférence lors d'une communication dans LEACH [56].

IV.4.2 Phase de transmission :

Cette phase est plus longue que la phase précédente, et permet la collecte de données captées. En utilisant l'ordonnanceur TDMA, les membres émettent leurs données captées pendant leurs propres slots. Cela leur permet d'éteindre leurs interfaces de communication en dehors de leurs slots afin d'économiser leur énergie. Ces données sont ensuite agrégées par les CH qui les fusionnent et les compressent, et, envoient le résultat final au nœud puits.

Après un certain temps prédéterminé, le réseau va passer à un nouveau round. Ce processus est répété jusqu'à ce que tous les nœuds du réseau seront élus CH, une seule fois, tout au long des rounds précédents. Dans ce cas, le round est réinitialisé à 0.

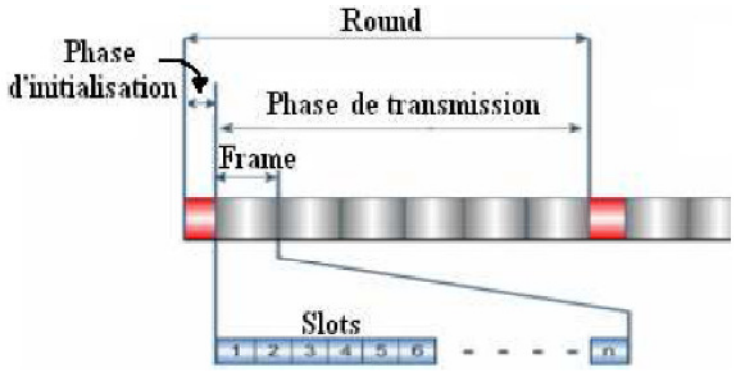


Figure 4.6 : Répartition du temps et différentes phases pour chaque round. [57]

IV.5 Avantages et inconvénients de LEACH :

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

Bien que LEACH économise la consommation d'énergie par un facteur de 8 [45] comparé à la transmission directe, grâce à l'agrégation de données et la réutilisation de largeur de bande, un nombre d'inconvénients restent plus ou moins apparents.
Dans ce qui suit, on cite quelques avantages et inconvénients du protocole LEACH.

IV.5.1 Avantages :

- **Protocole auto-organisateur basé sur le groupement adaptatif**: LEACH est complètement distribué, autrement dit, les nœuds prennent leurs décisions de façon autonome et agissent de manière locale et n'ont pas besoin d'une information globale ni d'un système de localisation pour opérer de façon efficace. De plus, la collection de données est faite périodiquement (l'utilisateur n'a pas besoin de toutes les données immédiatement). Pour exploiter cette caractéristique, ce protocole introduit un groupement adaptatif, c'est-à-dire, il réorganise les groupes après un intervalle de temps aléatoire, en utilisant des contraintes énergétiques afin d'avoir une dissipation d'énergie uniforme à travers tout le réseau. [58]

- **Rotation des rôles de chefs de groupes**: La rotation des rôles de chefs de groupes s'avère un facteur important pour l'organisation des nœuds. Ce rôle est épuisant en termes de d'énergie car les CH sont actifs tout au long de leur élection. Puisque le nœud puits est généralement loin du champ de surveillance, les CH diffusent une quantité plus importante d'énergie pour lui transmettre leurs données. Donc, si les CH sont choisis d'une manière fixe, leur énergie s'épuisera rapidement ce qui induit à leur défaillance. Par conséquent, tous les autres nœuds seront sans CH et donc inutiles. C'est pourquoi, les algorithmes de groupement (clustering) étudiés jusqu'ici adoptent la rotation du rôle de chefs de groupes. [57]

- **Faible énergie pour l'accès au média**: Le mécanisme de groupes permet aux nœuds d'effectuer des communications sur des petites distances avec leurs CH afin d'optimiser l'utilisation du média de communication en la faisant gérer localement par un CH pour minimiser les interférences et les collisions.
- **Compression locale (agrégation)** : Les CH compressent les données arrivant de leurs membres, et envoient un paquet d'agrégation au nœud puits afin de réduire la quantité d'informations qui doit lui être transmise. Cela permet de réduire la complexité des algorithmes de routage, de simplifier la gestion du réseau, d'optimiser les dépenses d'énergie et enfin de rendre le réseau plus évolutif (scalable).

IV.5.2 Les inconvénients :

- On pourra ne pas avoir des CH durant un round si les nombres aléatoires générés par tous les nœuds du réseau sont supérieurs à la probabilité Pi(t).
- Les nœuds les plus éloignés du CH meurent rapidement par rapport aux plus proches.
- L'utilisation d'une communication à un seul saut au lieu d'une communication multi-sauts diminue l'énergie des nœuds.
- Le protocole LEACH ne peut pas être appliqué à des applications temps-réel du fait qu'il résulte en une longue latence.
- La rotation des CH permet de ne pas épuiser les batteries. Cependant, cette méthode n'est pas efficace pour de grandes structures de réseaux à cause de la surcharge d'annonces engendrées par le changement des CH, et qui réduit le gain d'énergie initial.
- Il n'est pas évident que les CH soient uniformément distribués. Donc, il est possible que les CH puissent être concentrés dans une partie du réseau. Par conséquent, certains nœuds n'auront pas des CH dans leurs voisinages.
- Le protocole LEACH n'est pas sécurisé. Aucun mécanisme de sécurité n'est intégré dans ce protocole. Ainsi, il est très vulnérable même aux simples attaques. Donc, un attaquant peut facilement monopoliser le réseau et induit à son disfonctionnement.

IV.6 Conclusion :

Dans ce chapitre, nous avons présenté le protocole de routage hiérarchique LEACH qui suit une approche basée sur les groupes. Cette approche a montré son efficacité, comparée aux autres approches (par exemple, la topologie plate), en termes de consommation et de dissipation uniforme d'énergie prolongeant ainsi la durée de vie du réseau.

Nous avons vu que le protocole LEACH est soumis à certaines contraintes et suppositions qui engendrent toutefois des inconvénients. Par exemple, la communication unicast, établie entre le nœud puits et les CH et entre ces derniers et leurs membres, n'est pas toujours efficace par rapport à la communication multi-sauts.

Nous allons passer à l'implémentation de toutes les étapes de notre étude et donner des résultats démonstratifs qui les justifient. Le prochain chapitre sera consacré à la mise en œuvre de ce protocole.

CHAPITRE V :

IMPLEMENTATION ET EVALUATION DE LEACH

V.1. Introduction

L'impossibilité d'une intervention humaine dans les réseaux de capteurs mènent les chercheurs à trouvé d'autre recourt pour résoudre le problème des ressources d'énergie. Durant les dernières années, plusieurs algorithmes de routage ont été développés afin de résoudre ce problème.

Tel qu'on l'a montré au cours de l'étape précédente, l'objectif principal de notre travail est la mise en œuvre du protocole de routage LEACH.

Dans ce qui suit, nous allons modéliser un réseau de capteurs au sein de la plateforme TinyOS, ainsi nous allons aussi modéliser se protocole de routage et nous essaierons de l'évaluer selon des critères qui seront détaillés dans ce chapitre.

L'objectif de ce chapitre est donc de démontrer l'efficacité du protocole LEACH en termes de routage ainsi que d'autres métriques de performances via l'implémentation et la simulation de se protocole.

Pour cela, nous commencerons par définir les outils nécessaires pour l'implémentation et la simulation du protocole. Ensuite, nous décrirons la mise en œuvre de toutes les structures

de données et processus décrits précédemment. Nous terminerons ce chapitre par une présentation des résultats relevés lors des tests de performances de protocole LEACH.

V.2. Environnement de simulation

Dans cette section, nous présentons les outils utilisés pour la mise en œuvre de protocole LEACH. Nous commençons tout d'abord par TinyOS, le système d'exploitation conçu pour les RCSF. Nous parlons ensuite du langage de programmation NesC avec lequel nous avons programmé le code du protocole. Nous terminons cette partie par la présentation d'un simulateur des RCSF: TOSSIM qui offre deux mécanismes permettant d'émuler le réseau ; l'interface graphique TinyViz pour visualiser le déroulement de la simulation, et, le simulateur PowerTOSSIM pour simuler et évaluer la consommation d'énergie. Nous fournissons des informations plus détaillées dans l'annexe.

V.2.1. TinyOS

Suite aux différents défis des RCSF qu'on a vus dans les chapitres précédents, l'université de Berkeley, en plus de nombreux contributeurs ont développé un système d'exploitation destiné au RCSF afin de faciliter l'implémentation et l'exécution de protocoles dédiés à ce type de réseaux. L'objectif consiste à minimiser la taille du code afin de respecter les contraintes de ressources énergétiques et physiques des nœuds capteurs. Ce système est intitulé TinyOS. [59]

Il a l'avantage de permettre une programmation simple et puissante tout en gardant la portabilité du code pour les nombreuses plateformes supportées. Il est utilisé par plus de 500 universités et centres de recherche dans le monde vu la caractéristique open source qu'il détient [60]. Il respecte une architecture basée sur une association de composants. Il utilise une programmation entièrement réalisée en langage NesC.

V.2.1.1. Pourquoi TinyOS ?

Les systèmes d'exploitation pour les nœuds capteurs sont généralement moins complexes que les autres systèmes. Plusieurs systèmes d'exploitation ont été proposés pour les RCSF parmi lesquels on trouve SOS [62], Contiki[59], MANTIS[61].

TinyOS reste néanmoins le plus répandu pour les RCSFs car il répond aux exigences particulières des applications des RCSF. Il convient alors de mentionner les propriétés qui rendent TinyOS aussi populaire pour ce genre de réseaux: [64]

– Une taille de mémoire réduite.
– Une basse consommation d'énergie.
– Des opérations robustes.
– Applications orientées composants: TinyOS fournit une réserve de composants systèmes utilisables au besoin.
– Programmation orienté évènement : Généralement sur TinyOS, un programme s'exécute suivant le déclenchement des événements. Sinon, les capteurs restent en veille ce qui maximise la durée de vie du réseau.

V.2.2.2. Notions principales

TinyOS est construit autour des différents concepts décrits ci-dessous: [66]

- **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.

V.2.2. NesC

NesC 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. [64]

Il existe trois types de fichiers sources des applications NesC: les fichiers interfaces et les fichiers configurations et modules qui constituent les composants. [63]

- Une configuration définit les composants et/ou les interfaces utilisés par l'application déployée sur le capteur. Elle définit aussi la description des liaisons entre eux.

- Un module constitue la brique élémentaire du code et implémente une ou plusieurs interfaces.

- Une interface définit d'une manière abstraite les interactions entre deux composants. Elle définit un fichier décrivant les commandes et les évènements proposés par le composant qui les implémente. Une commande doit être implémentée par le fournisseur de l'interface et un évènement doit être implémenté par l'utilisateur de l'interface.

On distingue les modules et les configurations dans le but de permettre aux concepteurs d'un système de construire des applications rapidement et efficacement. Par exemple, un concepteur peut fournir uniquement une configuration qui relie un ensemble de modules qu'il ne développe pas lui même. De plus, un autre développeur peut fournir une librairie de modules qui peuvent être utilisés dans la construction d'autres applications. [24]

V.2.3. TOSSIM

Avant sa mise en place, le déploiement d'un RCSF nécessite une phase de simulation afin de s'assurer du bon fonctionnement de tous les protocoles de communication qu'il utilise.

En effet, pour de grands réseaux, le nombre de capteurs peut atteindre plusieurs milliers et entraine donc un coût financier relativement important. Ainsi, il faut réduire au maximum les erreurs de la conception. Malgré cela, il reste des facteurs réels qui ne peuvent être pris en compte par la simulation, tels que les contraintes physiques (perturbations électromagnétiques, inondations, etc.) ou les aléas (détériorations dues à un animal, etc.)

Pour arriver à simuler le comportement des capteurs au sein d'un RCSF, un outil très puissant a été développé et proposé pour TinyOS sous le nom de TOSSIM. Le principal but de TOSSIM est de créer une simulation très proche de ce qui se passe dans les RCSF dans le monde réel. Une économie d'effort et une préservation du matériel sont possibles grâce à cet outil. [65]

Pour une compréhension moins complexe de l'activité du réseau, TOSSIM peut être utilisé avec une interface graphique TinyViz. Cette dernière est équipée par plusieurs API

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. [67]

V.2.3.1. TinyViz

TinyViz est une interface graphique Java. Elle permet de donner un aperçu des capteurs à tout instant ainsi que des divers messages qu'ils émettent. Elle détermine un délai entre chaque itération des capteurs afin de permettre une analyse pas à pas du bon déroulement des actions en activant différents modes comme Radio, CPU, etc. [64]

V.2.3.2. PowerTOSSIM

Le simulateur TOSSIM n'a pas la capacité de vérifier le taux d'énergie dissipée pendant l'exécution des applications. Cependant, le besoin de vérifier la consommation énergétique dans un RCSF a un intérêt primordial. L'université de Harvard a conçu le simulateur PowerTOSSIM qui surmonte ce problème. Ce nouveau simulateur est intégré dans TOSSIM. Il permet de générer un fichier de l'extension **.**trace qui enregistre les détails de la simulation comme l'énergie consommée dans le réseau. [60]

V.3.Description de l'exemple à étudier

Dans un réseau de capteurs, les nœuds collaborent et échangent des données et des informations dont le but est d'assurer un service bien déterminé. Dans ce qui suit, nous allons développer un exemple de réseau de capteur, nous supposons que tous les nœuds d'un même réseau sont identiques.

Notre application consiste à un réseau qui permet de détecter les températures. Les nœuds sont équipés d'un capteur de température. Quand la température est détectée par un nœud, l'information sera diffusée par un envoie de paquet.
Cet exemple a été choisi dans le but d'évaluer les performances de cette application développé avec TinyOS.

Ainsi dans cette application, les nœuds mesurent périodiquement la température à partir du capteur. Ensuite la donnée capter par chaque capteur sera transmise au chef a qui il

appartient se dernier agrège toute les données qui la reçu de tous les capteurs appartenant à se cluster en utilisant la fonction d'agrégat AVERAGE (moyenne), et en dernier le paquet agréger sera transmit jusqu'à la station de base (puits) en utilisant un routage de type "multi hop''.

V.4. Implémentation du protocole LEACH

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

V.4. 1. Structures de données

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

A) Le noeud puits

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

B) Le noeud CH

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

}CLUSTER_HEAD;

C) Le noeud membre

typedef struct MEMBRE
{
uint16_t ID_MEMBRE;//l'identificateur de chaque membre qui correspond à tos_local_adress
uint16_t ID_CH; //l'identificateur du CH auquel appartiendra le noeud membre
uint8_t temp; //la température captée
}MEMBRE;

V.4. 2. Evénements

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

Evénement	**Sortie**	**Fonction**
LEACH_ReceiveMsg.receive(TOS_MsgPtr pmsg)	TOS_MsgPtr	Réception du round
ANNONCE_ReceiveMsg.receive(TOS_MsgPtr pmsg)	TOS_MsgPtr	Annonce du CH
ORGANISATION_ReceiveMsg.receive(TOS_MsgPtr pmsg)	TOS_MsgPtr	Formation de groupes
SLOT_ReceiveMsg.receive(TOS_MsgPtr pmsg)	TOS_MsgPtr	Réception des slots
Temperature_ReceiveMsg.receive(TOS_MsgPtr pmsg)	TOS_MsgPtr	Réception du CH des températures captées
AGGREGATION_ReceiveMsg.receive(TOS_MsgPtr pmsg)	TOS_MsgPtr	Réception du puits des résultats d'agrégation
ReqRelayTimer.fired()	result_t	Relai des annonces du round
RoundTimer.fired()	result_t	Envoi du nouveau round par le nœud puits

V.4.3. Déroulement

Dans cette partie, nous expliquons et déroulons les phases de l'algorithme LEACH. Nous allons simuler un réseau avec un nombre de 10 capteurs et leur positionnement illustrer dans l'image si dessous, en faisant appel à TinyViz. Un fichier de configuration est créé et permet à

TinyViz de se lancer avec des paramètres spécifiés (le nombre et l'emplacement des capteurs, la durée de la simulation et les plugins que nous souhaitons activer dès le début de la simulation comme Debug Messages).

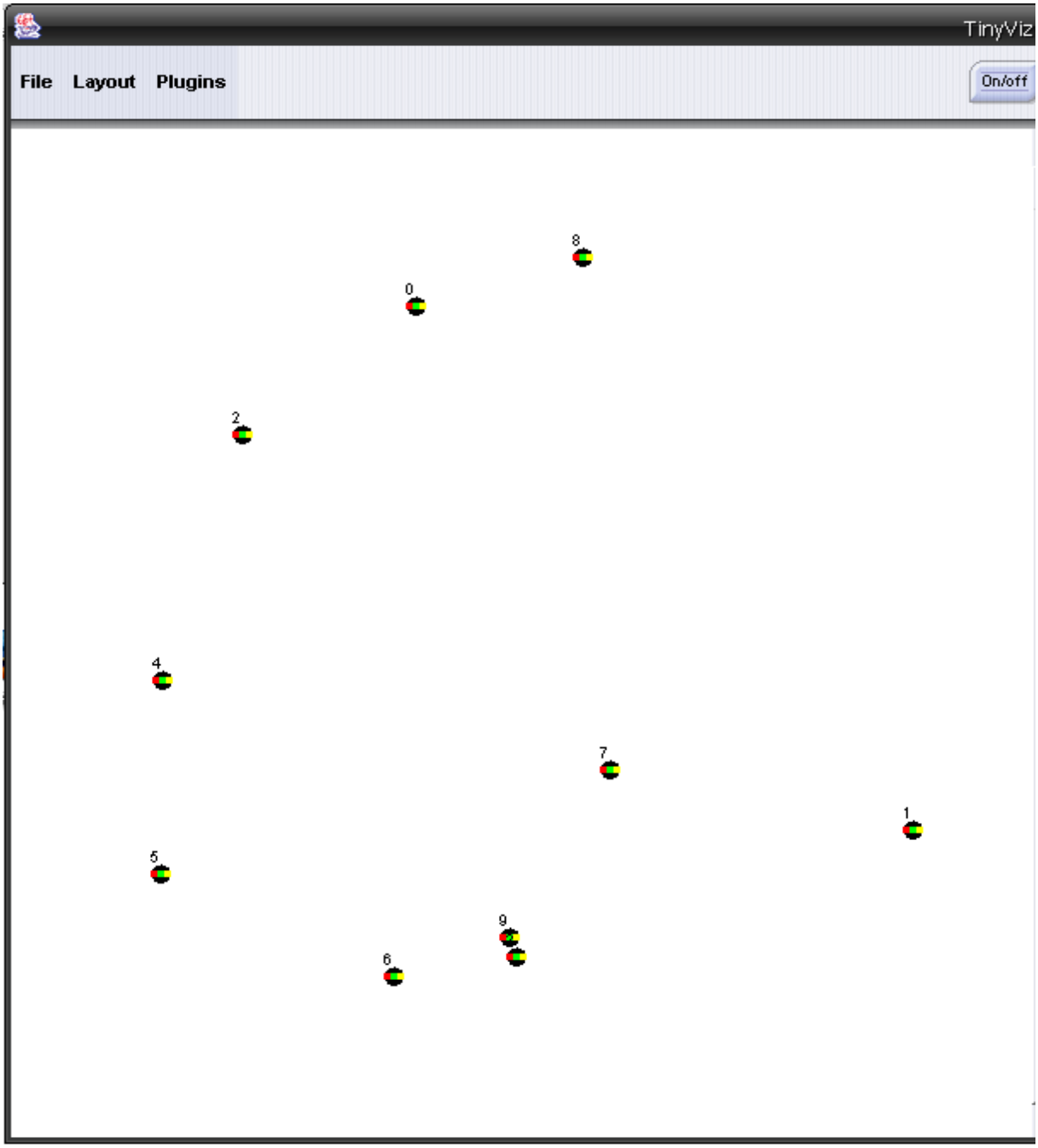


Figure 5.1 : Le positionnement des 10 nœuds sur la zone d'étude (exemple).

A) Phase d'initialisation :

Pour mieux comprendre cette étape, nous allons expliquer le déroulement de chaque phase.

A).1 Phase d'annonce :

La figure 5.2 représente les transmissions broadcast qui se passent durant différentes étapes de l'algorithme LEACH. Une transmission broadcast est repérée par un cercle bleu.

Le nœud puits envoie un broadcast aux nœuds voisins pour l'annonce du round. Ses voisins prennent le relai en envoyant à leur tour selon une transmission broadcast. De plus, nous pouvons voir que le nœud 7 est élu CH. Cet évènement est marqué par l'activation des LED rouges des CH. Ensuite, le CH 7 diffuse une annonce pour signaler son statut.

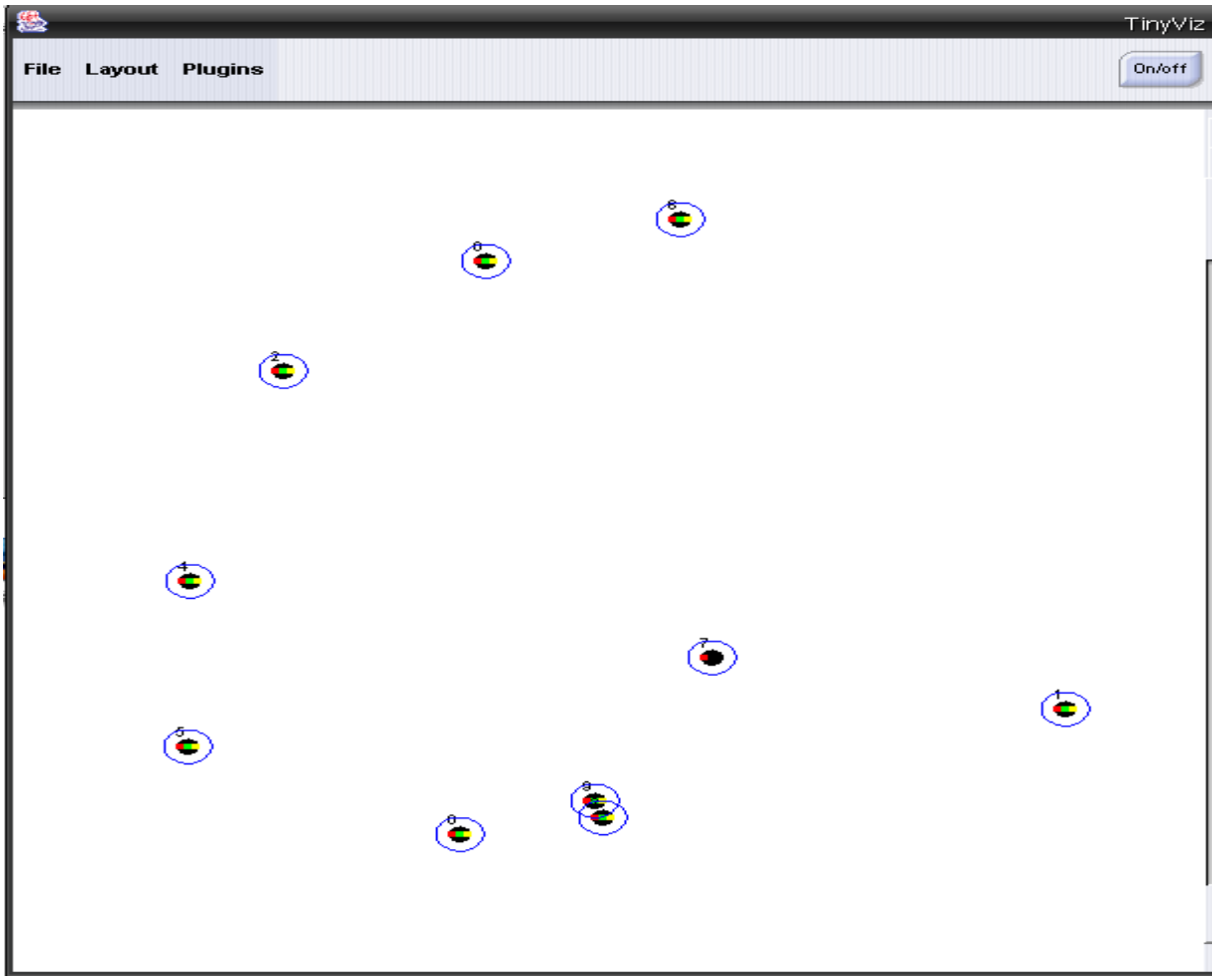


Figure 5.2 : Déclenchement round, et annonce du CH 7.

La figure suivante illustre les étapes détaillées de cette phase ainsi que les messages échangés entre le puits et les autres capteurs.

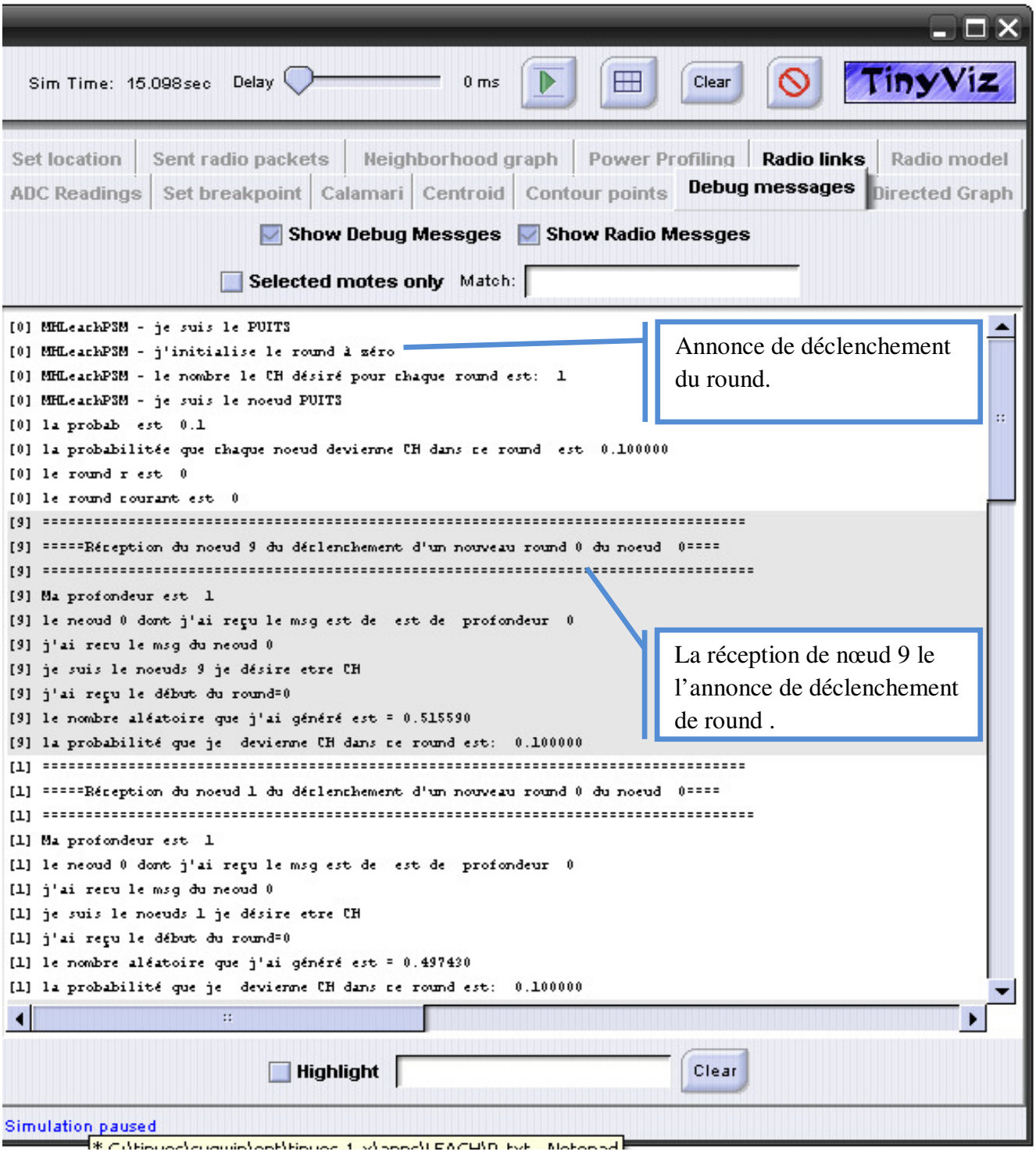


Figure 5.3 : Déclenchement du round, réception des nœuds de l'annonce de puits.

La figure si dessous indique que le capteur 7 est élu comme CH, car sa probabilité Pi est inferieur à celle que le puits a indiqué Pi(t).

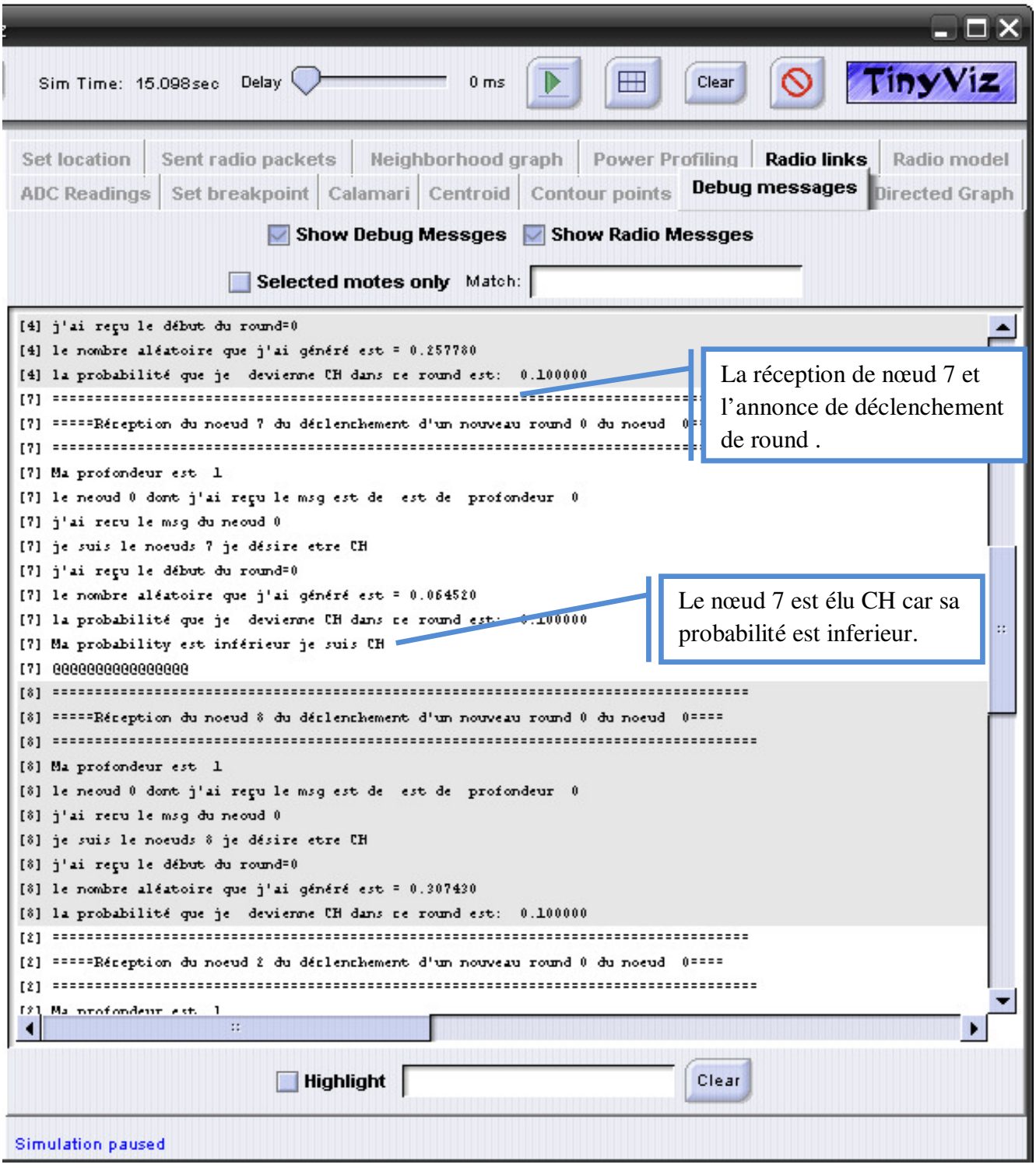


Figure 5.4 : L'élection du nœud 7 comme CH.

L'annonce du CH 7 par une transmission unicast (Une transmission unicast est repérée par une flèche).

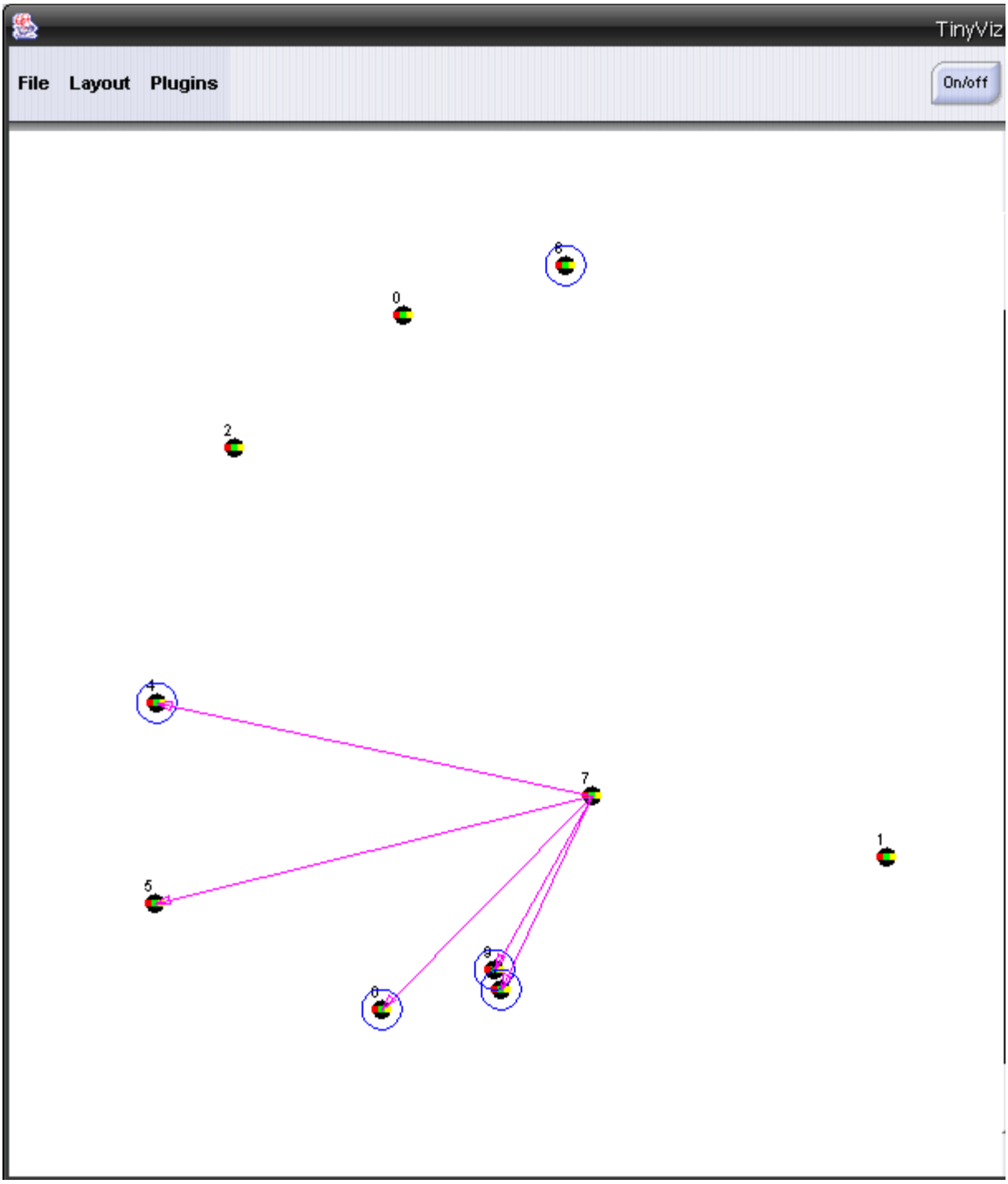


Figure 5.5 : Les transmissions unicast du CH 7.

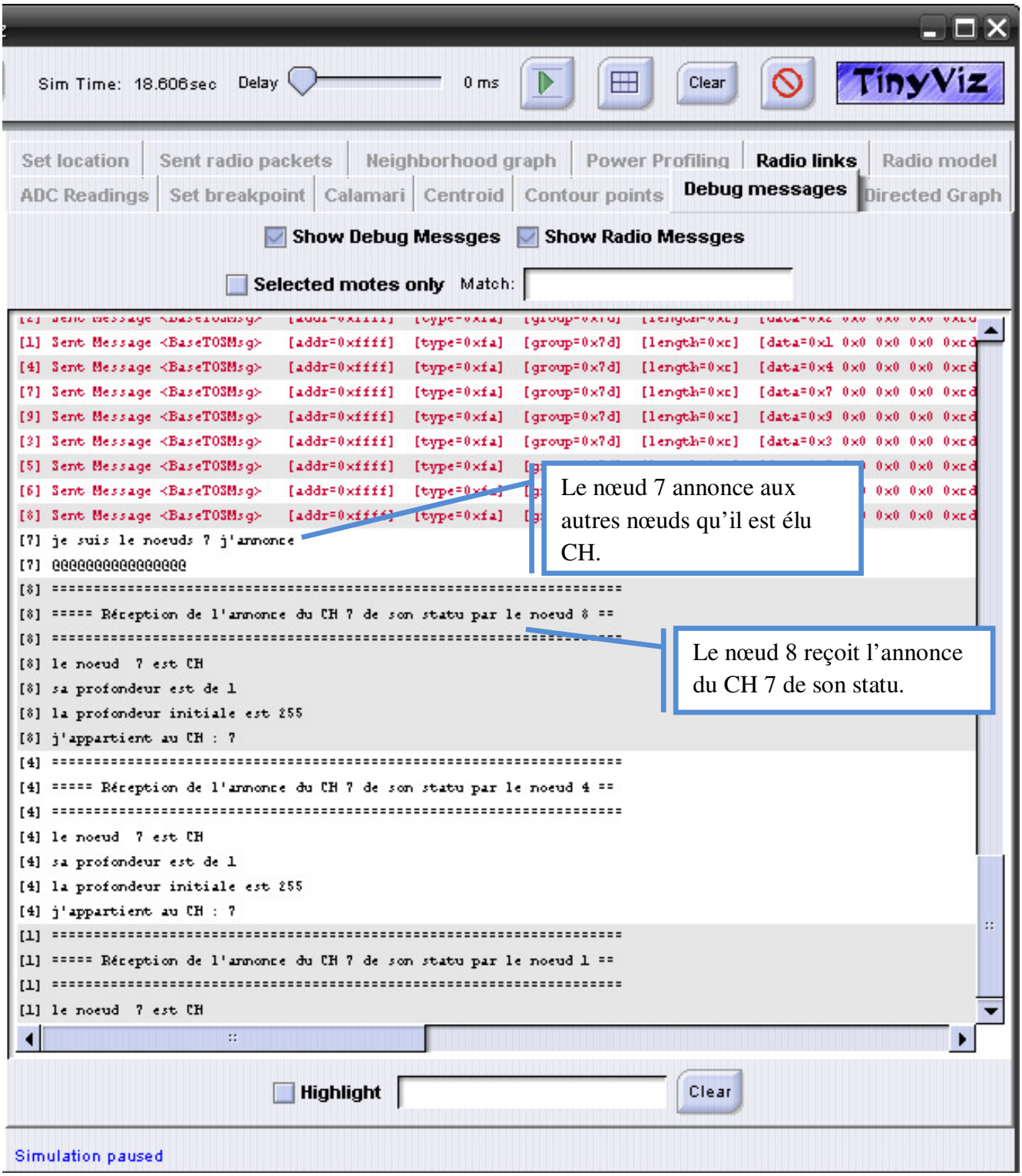


Figure 5.6 : L'annonce aux nœuds du statu de nœud 7 (CH).

A.2 Phase d'organisation de groupes :

- Les nœuds non CH envoient une requête d'appartenance au CH 7 (s'il y avait plusieurs CHs, les noeuds non CH répondent à l'annonce du CH le plus proche)
- La réception du CH 7 des demandes d'appartenance.

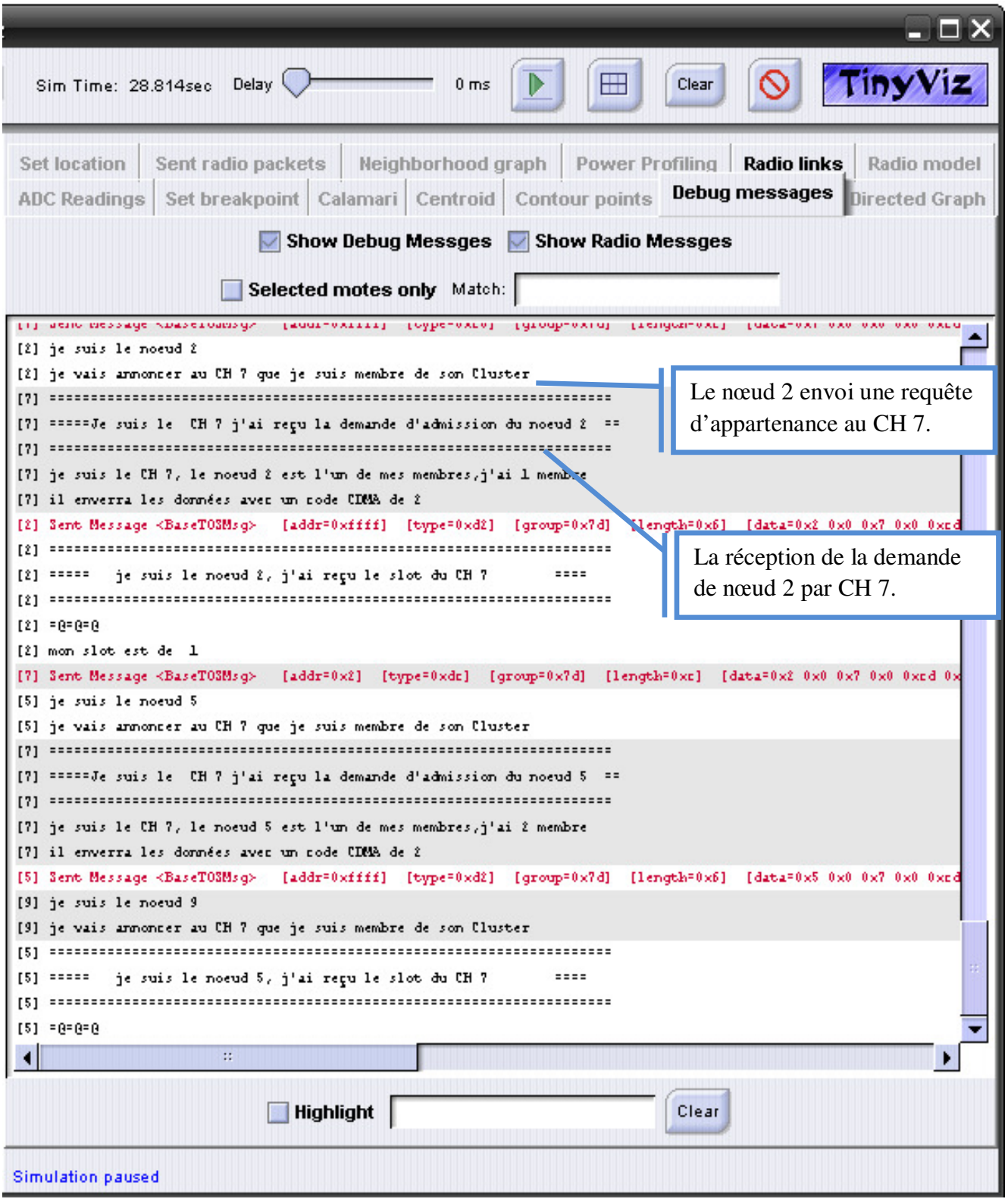


Figure 5.7 : Formation de groupes.

A.3 Phase d'ordonnancement :

- Après la formation du groupe le CH 7 crée une table TDMA est assigne a chaque nœud membre un slot de temps durant lequel il peut transmettre ses données.

- La réception des nœuds membres de leur slot.

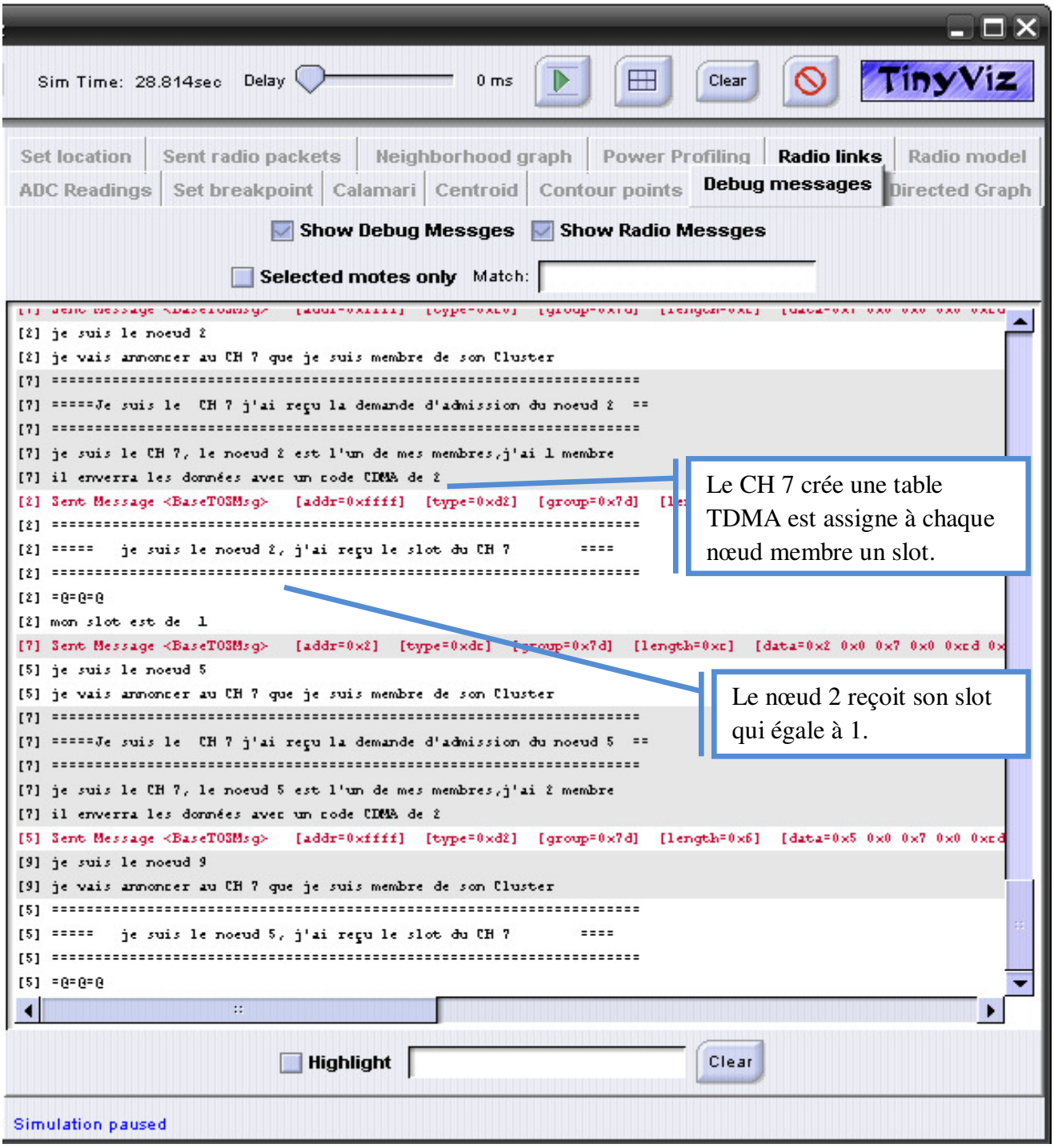


Figure 5.8 : Phase d'ordonnancement.

B). Phase de transmission:

- Chaque membre capte la température et attend le début de son slot pour qu'il puisse l'envoyer à son CH (Nous avons utilisé une application qui retourne la température sur une zone donnée).
- Le CH 7 collecte les données captées par les nœuds membres, ces données sont ensuit agrégées par le CH 7 qui fusionne et compresse ces températures en utilisant la fonction d'agrégat AVERAGE (effectue la moyenne des températures).

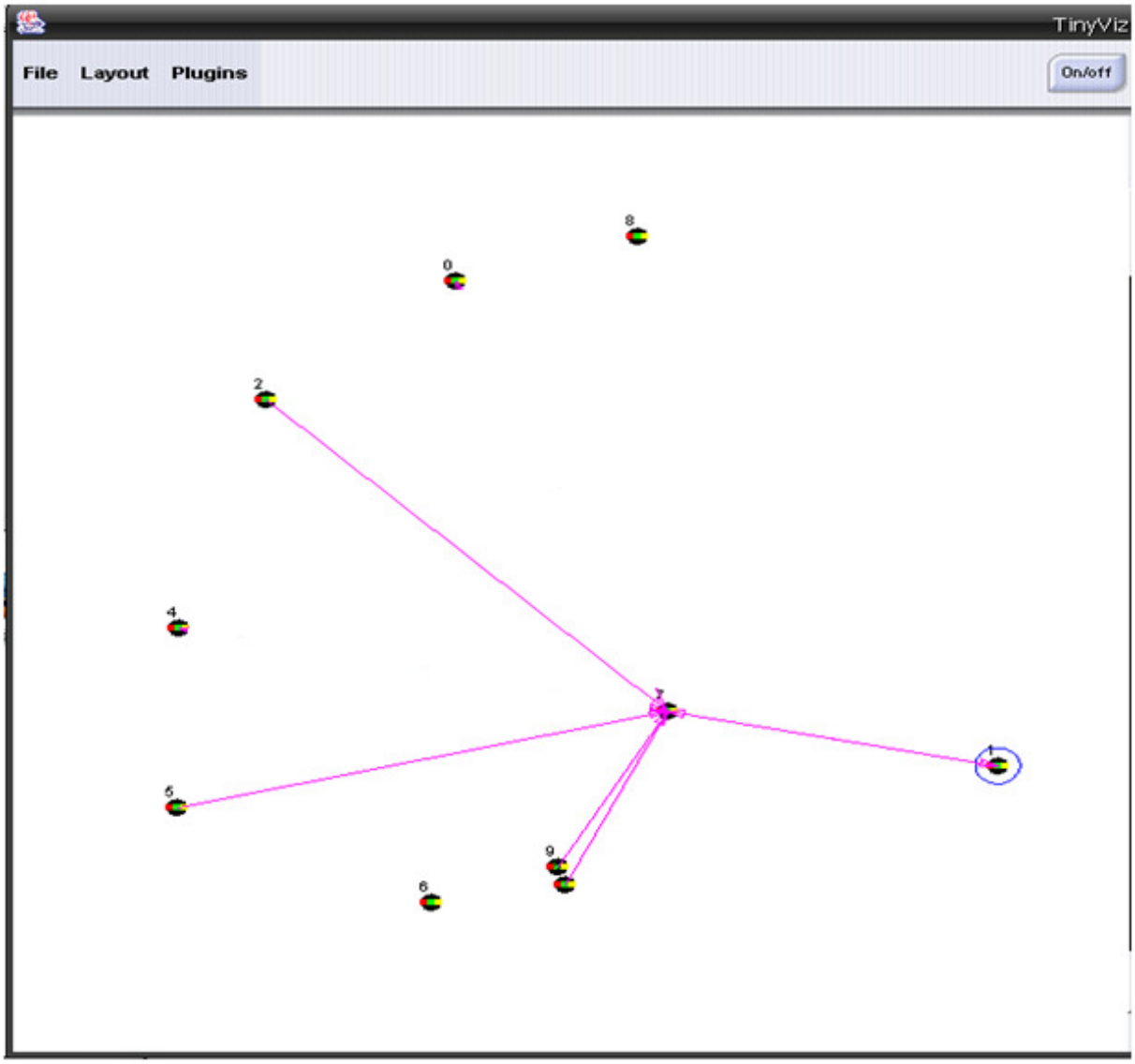


Figure 5.9 : L'envoi des températures au CH 7.

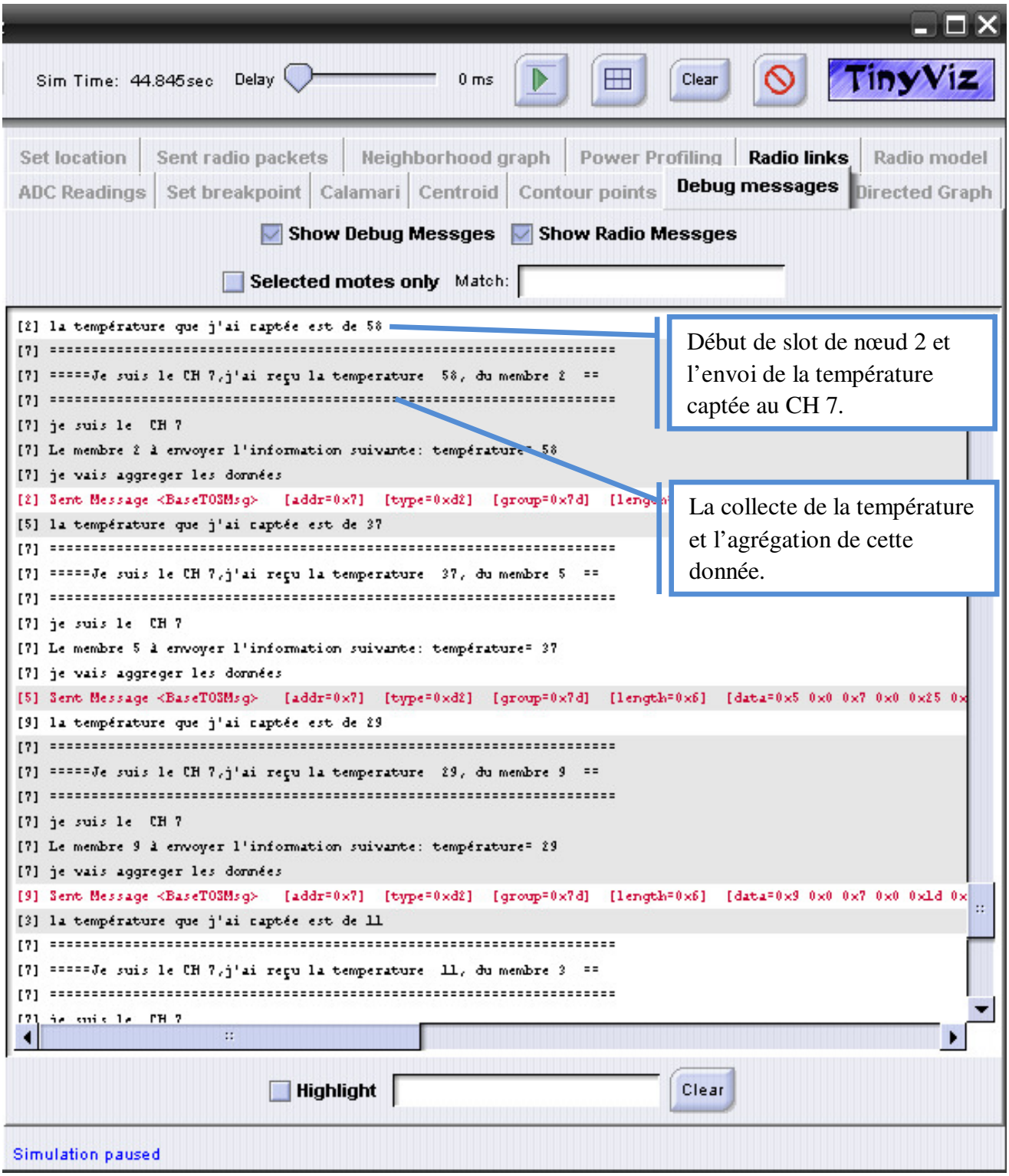


Figure 5.10 : La collecte et l'agrégation des données par le CH 7.

- L'envoi du résultat d'agrégation (la température moyenne) au nœud puits.
- Le déclenchement d'un nouveau round.

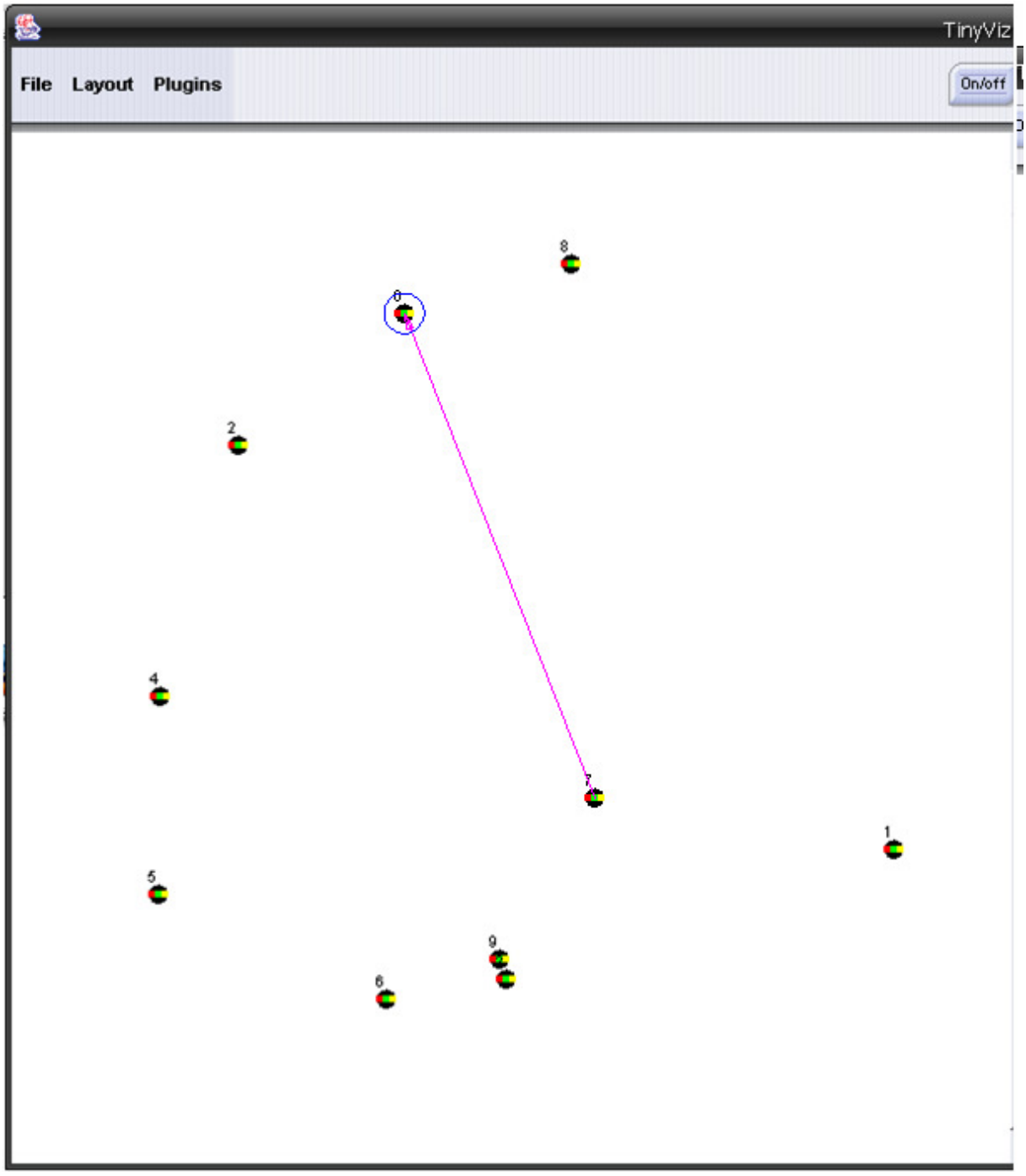


Figure 5.11 : Envoi du résultat d'agrégation des températures au nœud puits.

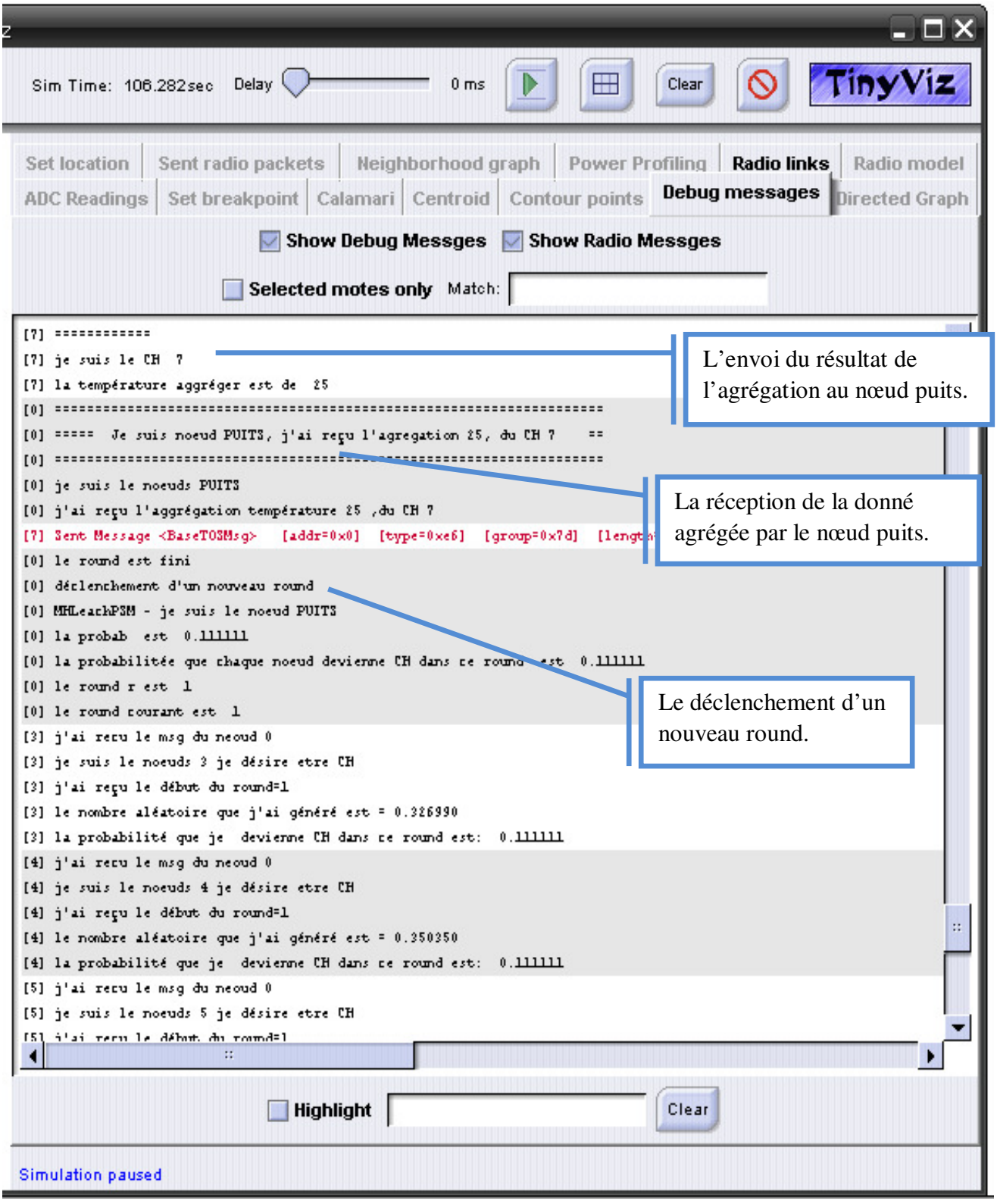


Figure 5.12 : La réception de la température finale et le déclenchement d'un nouveau round.

V.5. Simulation et évaluation de performances

Pour évaluer notre système, nous devons se rappeler toujours les buts de notre réseau, les résultats attendus et les avantages de la technologie utilisée pour élaborer le réseau utilisé. Les plus importants à prendre en compte pour l'évaluation de notre exemple est la durée de vie, le taux de perte des données et le délai de bout-en-bout.

V.5.1. Métriques à évaluer

Pour pouvoir évaluer les performances de LEACH, il est commode de mesurer certaines métriques qui sont :

V.5.1.1. Consommation énergétique

Nous nous sommes intéressés essentiellement à la consommation d'énergie des nœuds puisqu'elle constitue un paramètre primordial pour la détermination de la durée de vie d'un RCSF. En effet les nœuds ne sont pas entretenus pendant des mois et des années pour des raisons de sécurité et d'environnement. Le premier facteur limitant cette métrique est l'alimentation en énergie. Chaque nœud est conçu pour contrôler sa propre alimentation afin de maximiser la durée de vie de tout le réseau.

Pendant le fonctionnement d'un nœud, la communication radio, la gestion des données, les traitements des données,…, consomment beaucoup d'énergie. Donc il est trop important d'évaluer l'utilisation de la batterie. Pour se faire, nous prenons comme critère, l'énergie moyenne consommée par chaque nœud du réseau.

V.5.1.2. Perte de paquets

Le choix de cette métrique, comme étant un critère de performance, revient à sa nécessité dans certaines applications où les données échangées sont très critiques. Pour la mesurer, nous calculons la moyenne des taux de perte de paquets de températures entre les membres et leurs CH, et de paquets d'agrégation de ces températures entre les CH et le nœud puits. Il sera donc préférable d'avoir un taux de perte de paquet faible pour une bonne performance de réseau.

V.5.1.3. Délai de bout-en-bout

Le choix de cette métrique, comme étant une mesure de performance, revient à sa nécessité dans certaines applications temps-réel où on est obligé d'obtenir l'information le plutôt possible afin de prendre les mesures nécessaires.

Certains auteurs utilisent comme critère le temps moyen pour qu'un paquet soit acheminé d'une source jusqu'au nœud puits. Cependant, les CH dans les protocoles LEACH ne permettent pas d'utiliser ce critère.

En effet, les températures captées par les membres ne sont pas envoyées au nœud puits, mais, elles sont agrégées par les CH. Rappelons que ces derniers attendent que tous les membres achèvent l'opération de captage avant de passer à la phase d'agrégation. Donc, le critère que nous utilisons est l'EED moyen de tous les paquets transitant dans le réseau.

V.5.2. Résultats et interprétations

Dans cette partie, nous évaluant les métriques citées précédemment sur le protocole LEACH.

A. Consommation d'énergie des CH (Chefs, clusters head) et des membres sur un échantillon de 50 nœuds :

Dans ce test, nous avons mesuré le taux de consommation d'énergie des CH par rapport aux nœuds membres pour se protocole.

	LEACH
Consommation énergétique des MBR (joule)	15,611
Consommation énergétique des CH (joule)	19,836
Energie additionnelle des CH par rapport aux membres	21,29%

Comme l'illustre le résultat le tableau précédent, la moyenne de la consommation d'énergie des CH dans le protocole LEACH est plus élevée que celle des membres avec un taux de 21,29%.

Cette hausse enregistrée dans la consommation d'énergie est induite par les tâches coûteuses en termes d'énergie qu'effectue le CH lors de son élection.

B. Consommation d'énergie par nœud sur un échantillon de 50 nœuds

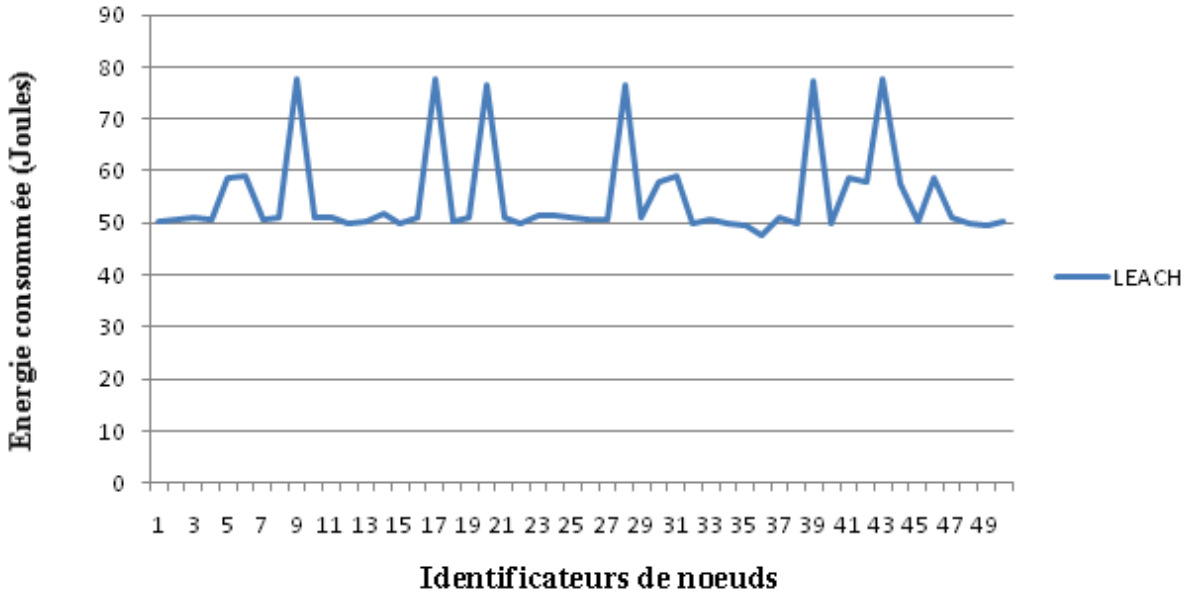


Figure 5.13 : Energie consommée par un nœud

Nous pouvons vérifier, en analysant le résultat du graphe VI, que les sommets représentent l'énergie consommée par des nœuds qui ont été élus CH durant la simulation.

Nous pouvons bien constater que les nœuds CH consomment plus d'énergie que le nœud membres, pour les raison expliquer précédemment.

C. Variation de la consommation d'énergie au nombre de nœuds du réseau

Nombre de nœuds	**50**	**100**	**150**	**200**
Moyenne de consommation d'énergie dans LEACH	16,315	16,438	16,243	16,487

Comme l'illustre le résultat du tableau, nous remarquons que la moyenne d'énergie consommée dans le réseau est indépendante du nombre de nœuds déployés à cause de la topologie hiérarchique du protocole LEACH qui le rend très scalable. En effet, quand la taille du réseau augmente, le nombre de CH augmente. Donc, les nouveaux nœuds vont être affectés aux nouveaux CH et regroupés indépendamment des groupes déjà existants dans le réseau. Donc, malgré l'augmentation du nombre de nœuds déployés, la taille de tous les groupes est la même. Ainsi, tous les CH effectuent le même taux de tâches. Ainsi, LEACH maintient la consommation d'énergie des nœuds quelque soit la taille du réseau.

V.5.2.2. Perte de paquets

Pour tester le taux de pertes de paquets, il est nécessaire de calculer le ratio des paquets perdus et des paquets envoyés. Voici le tableau de résultats de ce test:

	Nombre de paquets perdus sur le nombre de paquets envoyés			
	50	**100**	**150**	**200**
LEACH	19/286	41/585	63/857	91/1149

Nous aurons ainsi, les taux de pertes de paquets suivants :

	50	**100**	**150**	**200**
LEACH	6,64%	7,00%	7,35%	7,91%

Comme l'illustre les deux tableaux, nous remarquons que les taux de pertes de paquets échangés sont tolérables pour ce protocole. Ils varient entre 6,5 % jusqu'à 8,5% selon le nombre de nœuds déployés ; à chaque fois que le nombre de ces derniers augmente, la perte de paquets augmente. Cela est engendré par l'augmentation de collisions.

V.5.2.3. Délai de bout-en-bout

Pour évaluer l'EDD, nous avons eu recours au fichier trace généré par le simulateur PowerTOSSIM. En effet, ce fichier nous a permis de récupérer les temps des émissions et des réceptions de paquets de températures. Voici le tableau de résultats de ce test:

Nombre de nœuds	**50**	**100**	**150**	**200**
LEACH	97,316	100,988	100,678	99,993

Comme l'illustre le tableau, nous remarquons que les EDD sont indépendants du nombre de nœuds déployés. En effet, les communications entre les membres et leurs CH et entre les CH et le puits sont des communications à un seul saut. Bien que la taille du réseau augmente, il n'existe pas de longues routes qui peuvent influencer sur les EED. Cela rentre dans la cadre de la forte scalabilité du protocole LEACH.

V. Conclusion :

Dans ce chapitre, nous avons présenté l'implémentation ainsi que l'évaluation de protocole LEACH. Le système d'exploitation TinyOS est utilisé. Il consiste une programmation entière en langage NesC et une simulation avec TOSSIM.

Par ailleurs, nous avons constaté que les tests de performances effectués sur la consommation d'énergie, la perte de paquets de données échangés et le délai de bout-en-bout, ont montré que le protocole LEACH répond bien aux critères de performances souhaités. En effet, ce protocole ne permet pas de surcharger les nœuds capteurs, ni de dégrader les performances du réseau.

Mais l'inconvénient de ce protocole c'est que certains nœuds fonctionnent beaucoup plus que les autres nœuds et surtout ce sont les clusters. Même avec le concept de ce protocole clustering dynamique a pour désavantage le fait que ces nœuds meurent beaucoup plus vite que d'autres donc ce qui peut établir un déséquilibre dans le réseau ou bien un manque dans la précision et l'exactitude des informations reçues à la base station.

Conclusion générale

Tout au long de notre projet, nous avons constaté que la réalisation d'un RCSF pose de grands défis auxquels il faut répondre : la conservation d'énergie de ces réseaux est l'un des défis les plus importants à considérer. Les réseaux de capteurs sans fil sont des réseaux d'énergie limitée. Comme la plupart de l'énergie est consommée pour transmettre et recevoir des données, le processus d'agrégation des données devient un enjeu important et une optimisation est nécessaire.

Un efficace protocole d'agrégation des données doit non seulement assurer la conservation de l'énergie, mais aussi éliminer les redondances dans les données et donc de fournir seulement les données utiles. Il existe plusieurs protocoles pour l'agrégation de données qui utilise différentes approches pour fournir l'efficacité énergétique.

Dans ce projet, nous nous sommes intéressés à l'agrégation de données qui est l'un des services piliers sur lesquels se base le fonctionnement d'un RCSF. Pour cela, il nous a fallu étudier les différente technique de conservation de données ceci nous a permis de bien situé notre problématique, nous avons détaillé et explicité cette technique ainsi que nous avons cité certains protocoles de routage s'appuyant sur cette dernière.

Pour bien comprendre cette notion d'agrégation de données il nous a fallu d'étudier en détaille le protocole LEACH qui constitue un protocole de routage qui utilise l'agrégation de données dans son processus de routage, ensuite nous avons procéder a son implémentation on utilisant la plateforme TinyOS qui se base sur le langage Nesc. Dans le but de valider l'implémentation que nous avons mise en place, nous avons utilisé comme application le calcul de la moyenne des températures sur une zone donnée.

En dernier, pour évaluer se protocole nous avons définis les métriques les plus importante telle que la durée de vie et le taux de perte des données.

Nous avons atteint les objectifs fixés par notre projet. En effet étudier l'agrégation de données, nous avons implémenté un protocole sur la plateforme TinyOS, et nous avons pu évaluer se protocole selon des critères d'évaluation.

Ce projet nous a permis de se familiarisé avec les réseaux de capteurs sans fil et en particulier à la couche de routage (réseau) et l'agrégation de données dans les réseaux de capteurs sans fil. En effet, au bout de ces quelques mois, notre expérience dans le domaine des réseaux de capteurs sans fil s'est enrichie et notre savoir s'est approfondi. On a pu mettre à profit les connaissances acquises en matière de simulation. En effet, on a appris à implémenter et simuler un protocole de routage sous TinyOS. De plus, on c'est initié au domaine de la recherche.

En guise de perspective, nous envisageons d'implémenter un autre protocole qui prend en charge la contrainte d'énergie et surtout la solution de l'agrégation de données, l'évaluer et le comparer avec le protocole LEACH. Nous envisageons aussi d'introduire des mécanismes de sécurité pour pallier au problème de sécurité rencontré dans l'agrégation de donnée dans les RCSF.

Références

[1]: Noury, Norbet. Du signal à l'information : le capteur intelligent Exemples industriels et en médecine. Grenoble : TIMC-IMAG, 2002.

[2]: CAYIRCI, E. (2004). **"Wireless sensor networks".** In : D. Katsaros et al. (éd), Wireless information highways (pp. 273-301). Hershey : Idea group inc.

[3]: Cheng, K. Field and Wave Electromagnetics. s.l. : Addison-Wesley, 1989. p. 639.

[4]: CrossBow. Sensor Boards. [En ligne] https://www.xbow.com/Products/productdetails.aspx?sid=158.

[**5] :** Kamel BAYDOUN **« Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs**» Thèse, Université de Franche-Compte, 2009

[6]: Adel CHOUHA **« Traitement et transfert d'images par réseau de capteur sans fil»** These ,Université de hadj-lakhder batna 2010

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

[8]: 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

[9] : F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A survey on sensor networks. IEEE Communications Magazine, vol 40, pp. 102-114, August, 2002.

[10] : Yazeed Al-Obaisat, Robin Braun "On Wireless Sensor Networks: Architectures, Protocols, Applications, and Management" Institute of Information and Communication Technologies University of Technology, Sydney, Australia.

[11]: Eiko Yoneki, Jean Bacon, "A survey of Wireless Sensor Network technologies: research trends and middleware's role," Technical Report, no: 646, *UCAM ¡CL¡TR¡*646, [Available from the World Wide Web http://www.cl.cam.ac.uk/TechReports/*UCAM¡CL¡TR¡* 646.pdf].

[12]: H. Cam, S. Ozdemir, P. Nair, and D. Muthuavinashippan, "ESPDA: Energy-Efficient and Secure Pattern Based Data Aggregation for Wireless Sensor Networks," in press, IEEE Sensor, Toronto, Canada, 2003.

[13]: http:// busnissweek.com/datedtoc/1999/9935.htm.21 ideas for the 21st century.

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

[15] : A. Manjeshwar and D. P. Agarwal. **"**Apteen : A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks**"**, Parallel

anddistributed processing Symposium. Proceedings International, *IPDPS*, pp. 195-202, 2002.

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

[17] : M. Younis and T. Nadeem. **"**Energy efficient MAC protocols for wireless sensor networks**",** *Technical report, university of Mryland baltimre County,* USA, 2004.

[18]: H. Namgoog, D. Lee, and D. Nam. **"**Energy efficient topology for wireless microsensor networks**"**. *ACM, PE-WASUN*, October 2005.

[19]: S. Ziane and A. Mellouk. **"**A swarm intelligent scheme for routing in mobile ad networks**"**. *Systems Communications, IEEE*, Aug 2005.

[20]: Paolo Santi. **"**Topology Control in Wireless Ad Hoc and Sensor Networks**",** Hardcover, july 2005.

[21] : S. Kumar, D. Shepherd, and F. Zhao. **"**Collaborative signal and information processing in micro-sensor networks**"**. *IEEE Signal Processing Magazine*, March 2002.

[22]: S.Narayanaswamy, V. Kawadia, R.S. Sreenivas, and P.R. Kumar. **"**Power control in ad-hoc networks : Theory, architecture, algorithm and implementation of the Compowprotocol**"**, *European Wireless Conference,* 2006.

[23]: W.Ye, J.Heidemann, and D. Estrin. **"**Medium access controle with coordinated adaptative sleeping for wireless sensor networks**"**. *IEEE/ACM trans.Netw*, vol. 12,no.3,pp 493-506, Jun 2004.

[24] : Yacine CHALLAL « Réseaux de capteurs sans fils » université de technologie Compiègne 2008.

[25] : Alam, N., Clouser, T., Thomas, R., Nesterenko, M.: Emuli: model driven sensor stimuli for experimentation. In: Proc. 6th ACM Conf. Embedded Networked Sensor Systems (SenSys 2008), ACM Press, New York (2008) 423–424

[26] : Bandyopadhyay, S., Gianella, C., Maulik, U., Kargupta, H., Liu, K., Datta, S.: Clustering distributed data streams in peer-to-peer environments. Information Science 176(14) (2004)

[27] : Dey, A.: Context-aware computing: The cyberdesk project. In: Proceedings of the AAAI Spring Symposium on Intelligent Environments, pp. 51–54 (1998)

[28] : Alwan, M., Rajendran, P.J., Kell, S., Mack, D., Dalal, S., Wolfe, M., Felder, R.: A smart and passive floor-vibration based fall detector for elderly. In: Proceedings of the 2nd International Conference on Information and Communication Technologies, pp. 1003–1007 (2006)

[29] : Alexandre Brianceau, Jérémie Christin « Sécurité de l'agrégation de donnée dans les réseaux de capteurs» Université de Versailles Saint-Quentin-en-Yvelines .

[30] : M. Dalmau « Réseaux de capteurs » Université de Bayonne

[31] : Alexandre Delye de Mazieux, Vincent Gauthier, Michel Marot, and Monique Becker. État de l'art sur les réseaux de capteurs. INT 05001RST, avril 2005.

[32] : W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. 5th annual ACM/IEEE international conference on Mobile computing and networking . August 1999, pp. 174 - 185.

[33] : Kemal Akkaya, Mohamed Younis. A survey on routing protocols for wireless sensor networks. Ad Hoc Networks. May 2005, Vol. 3, 3, pp. 325-349.

[34] : Heinzelman, M. et Perillo, W. Wireless Sensor Network Protocols. [éd.] CRC Hall. 2005.

[35] : 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, pp. 2009-2015.

[36] : Arati Manjeshwar, and Agrawal, Dharma P. APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks. IPDPS '02, 16th International Parallel and Distributed Processing Symposium. 2003, p. 48.

[37]: M. Lehsaini, H. Guyennet, M. Feham. A novel cluster-based self-organization algorithm for wireless sensor networks. IEEE International Symposium on Collaborative Technologies and Systems (CTS 2008). May 2008, pp. 19-26.

[38]: Mohamed, LEHSAINI. Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique. s.l. : UFC, Juillet 2009. Thèse de doctorat.

[39]: Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks", in SIGMOD Record, September 2002.

[40]: Samuel Madden, Michael Franklin, Joseph Hellerstein,WeiHong UC Berkeley UsinexOSDI'02.

[41]: Samuel Madden, al UC Berkeley 2002.

[42]: Intanagonwiwat C., Estrin D., Govindan R., Heidemann J., Fabio Silva, Directed diffusion for Wireless Sensor Networking, Networking, IEEE/ACM Transactions on , Volume: 11 , Issue: 1 , Feb. 2003.

[43]: S. Madden, M. Franklin, J. Hellerstein, TAG: a Tiny AGgregation Service for Adhoc Sensor Networks, OSDI December 2002.

[44]: J. Gehrke , Yong Yao, The Cougar Approach to In-Network Query Processing in Sensor Networks,SIGMOD, 2002.

[45]: Mounir Achir, **« Technologies basse consommation pour les réseaux Ad Hoc**», Thèse, Institut National Polytechnique de Grenoble, 06 Juillet 2005.

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

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

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

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

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

[51]: Mounir Achir, **« Technologies basse consommation pour les réseaux Ad Hoc**», Thèse, Institut National Polytechnique de Grenoble, 06 Juillet 2005.

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

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

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

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

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

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

[58]: Srajan Raghuwanshi, **«An Energy Efficient Cross Layer Design Scheme for Wireless Sensor Networks**», Master's Thesis, Virginia Polytechnic Institute and

State University, 29 Août 2003

[59]: Adam Dunkels, Björn Grönvall, Thiemo Voigt, **« Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors »**, 29th Annual IEEE International Conference on Local Computer Networks, Pages: 455–462, Swedish Institute of Computer Science, 2004.

[60]: Borrong Chen, Geoff Werner Allen, Mark Hempstead, Matt Welsh, Victor Shnayder, **« Simulating the Power Consumption of LargeScale Sensor Network Applications»**, Proceedings of the 2nd international conference on Embedded networked sensor systems, Pages: 188 – 200, Harvard University, 2004.

[61]: Cormac Duffy, Cormac J. Sreenan, John Herbert, Utz Roedig, **« A Performance Analysis of MANTIS and TinyOS »,** Technical Report CS-2006-27-11, University College Cork, Ireland, November 2006.

[62]: C. Han, E. Kohler, M. Srivastava, R. Kumar, R. Shea, **« A Dynamic Operating System for Sensor Nodes»,** Proceedings of the 3rd International Conference on Mobile Systems, Applications and Services (Mobisys), Page(s): 163-176, University of California, Los Angeles, June 2005.

[63]: David Gay, Philip Levis, **« TinyOS Programming »**, Livre, ISBN: 0521896061, Nombre de Pages: 264, Presse de l'université de Cambridge, 28 Juin 2006.

[64]: H. Alatrista, J. Mathieu, K. Gouaïch S. Aliaga, **« Implémentation de protocoles sur une plateforme de réseaux de capteurs sans fil »**, TER master 1 informatique, Université de Montpellier II, 29 Avril 2008.

[65]: Mathieu Badnet, Nicolas Belloir **«Réseaux de capteurs : Mise en place d'une plateforme de test et d'expérimentation »**, Master Technologie de l'Internet 1ère année, France, 2005/2006.

[66]: Sylvie Tixier, **« TinyOS »**, Mini rapport, LIF12, Université Lyon 1, 6 Décembre 2007.

[67]: Wassim Znaidi, **« Modélisation formelle de réseaux de capteurs à partir de TinyOS »,** Projet de fin d'études, Ecole Polytechnique de Tunisie, 2006.

[67]: Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Eredal Cayirci. A Survey on Sensor Networks. IEEE Communications Magazine, 40(8) :102–114, August 2002. 5,7

[68]: Gianluigi.Ferrari(ed) **« Sensor networks, where theory Meets practice »** Springer

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 sur 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 XP.

A.6.1. Procédure d'installation sous Windows XP : 

Ce guide propose l'installation du principal outil nécessaire au bon fonctionnement du système, notamment Cygwin (couche d'émulation de l'API Linux) qui permet d'avoir une

interface Unix sous Windows. Cygwin est un environnement d'émulation Linux qui permet d'avoir un shell et de compiler et exécuter les programmes Linux (On dispose ainsi de gcc, apache, bash, etc.).

Fig. A-2 : Cygwin.

1- Télécharger le fichier tinyos-1.1.0-1is.exe de la source http://www.tinyos.net/dist 1.1.0/tinyos/windows/ .
2- Exécuter ce fichier pour installer la version 1.1.0 sous windows XP. L'installation se fait automatiquement. Un raccourci de Cygwin est sauvegardé sur le bureau.
3- Accéder à **C:\tinyos\cygwin\opt\tinyos-1.x\doc\tutorial\verifyhw.html** et suivre les étapes que contient cette page afin de vérifier si l'installation est bien réussie.

A.6.2. Procédure de désinstallation :

Cygwin ne possède pas de désinstalleur intégré, mais, ce logiciel étant propre, il n'éparpille pas ses fichiers sur le disque. Il est facile à désinstaller à la main. Si des services (tels que Apache ou sshd) ont été installées, il et très important de les arrêter et les désinstaller avant de désinstaller Cygwin.

1- Pour arrêter un service taper : **cygrunsrv -E nomDuService**

   Ou bien passer par le panneau de configuration.
   Puis supprimer le service : **cygrunsrv -R nomDuService**

2- Supprimer le répertoire **c:\cygwin** et tout ce qu'il contient.

3- Supprimer le sous-répertoire qui se trouve juste en dessous de setup.exe: il contient tout ce que l'installeur Cygwin a téléchargé. Ce répertoire porte un nom long qui correspond au miroir qui a été utilisé pour télécharger Cygwin.
Par exemple : **http%3a%2f%2fcygwin.cict.fr**

4- Prendre Regedit et supprimer les 2 entrées suivantes en base de registre :

HKEY_LOCAL_MACHINE\Software\Cygnus Solutions

Et

HKEY_CURRENT_USER\Software\Cygnus Solutions

5- Retirer les raccourcis que Cygwin a créés sur le bureau et dans le menu Démarrer.

6- Eventuellement, retirer le chemin **c:\cygwin** ou **c:\cygwin\bin** qui a été ajouté à la variable d'environnement PATH.
(Clic-droit sur le **poste de travail > Propriétés > onglet > « Avancé » > Variables d'environnement**).

7- Si des services (tels que ssh, NFS …) ont été installés, les scripts d'installation Cygwin ont probablement créé des utilisateurs spéciaux dans Windows pour faire tourner ces services (par exemple, l'utilisateur « sshd_server »pour le serveur ssh). Il faut également supprimer ces utilisateurs en passant par le panneau de configuration (ou bien en tapant **control userpasswords2**).

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

C:\tinyos\cygwin\opt\tinyos-1.x\doc\tutorial

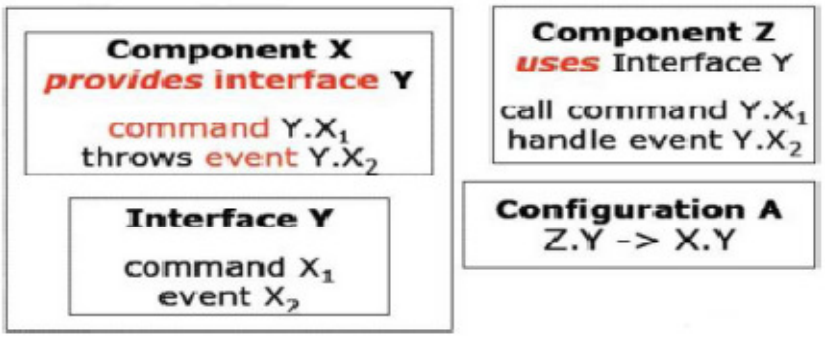


Fig. B-1 : Architecture générale d'une application NesC.

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.

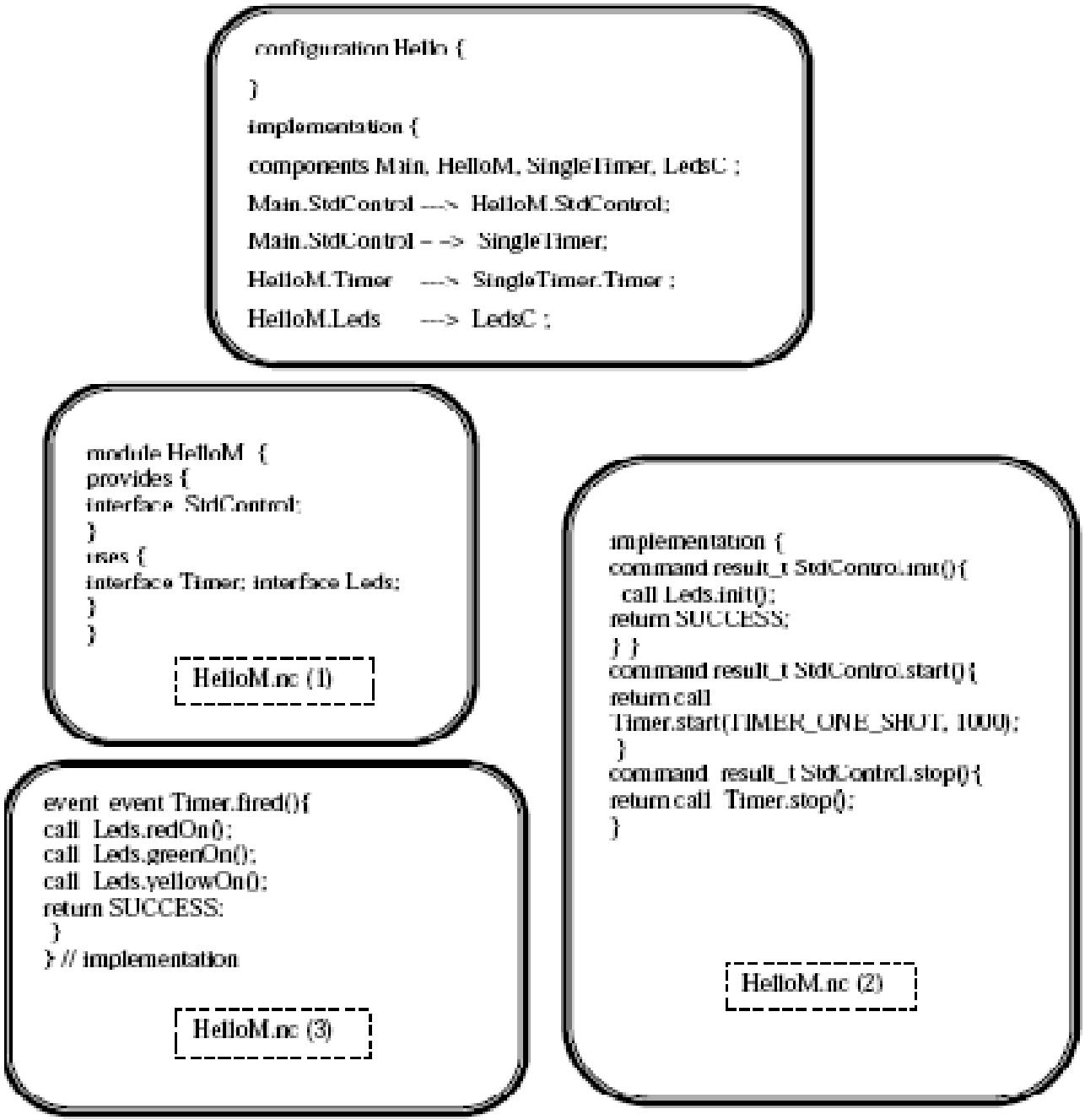


Fig. B-2 : Exemple illustratif d'une application NesC.

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

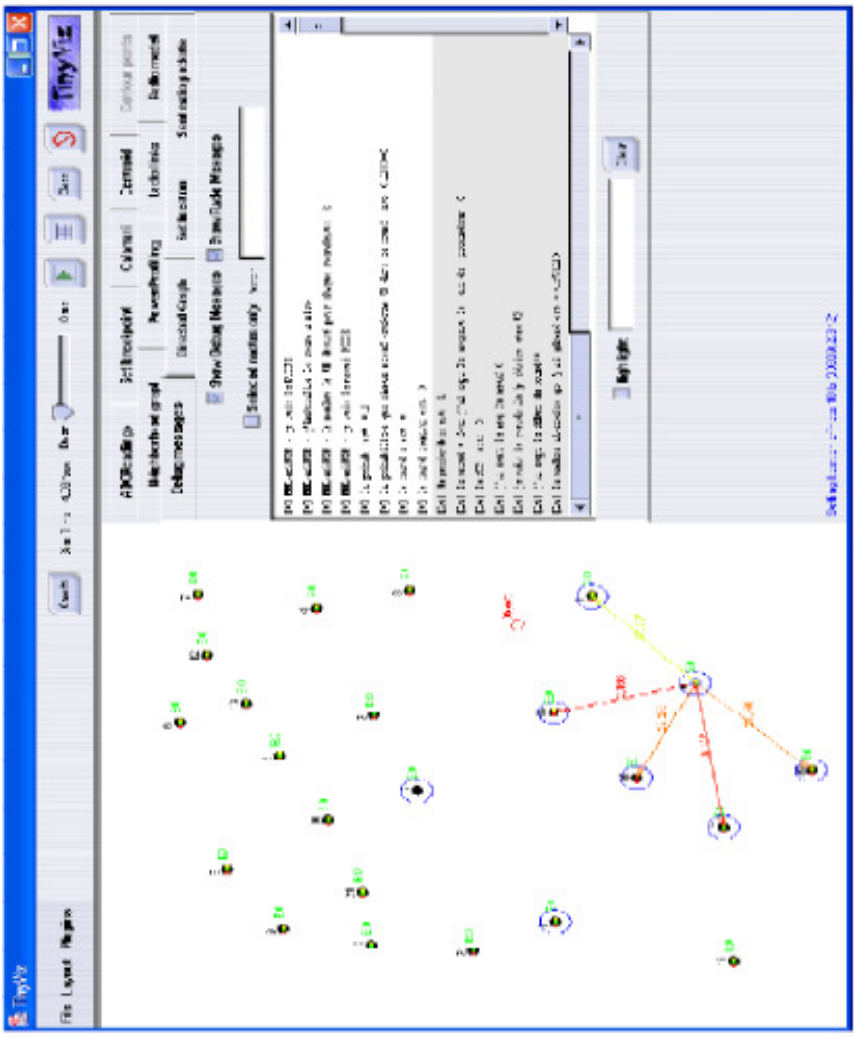


Fig. C-1: fenêtre graphique de TinyViz.

Une fois TinyViz est lancé, on peut visualiser une fenêtre comme celle illustrée dans la figure C.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.

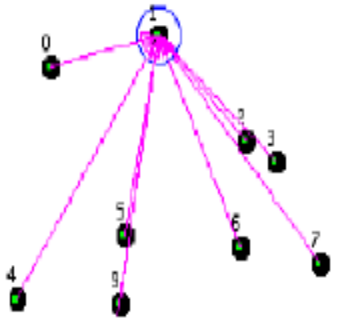


Fig. C-2 : Echange de messages entre les nœuds.

C.2. Installation de TinyViz :

Les concepteurs développent au fur et à mesure l'outil TinyViz sans mettre à jour les fichiers sources déjà existants dans les anciennes versions. Cela ne permet pas de lancer TinyViz dans des conditions normales. Pour pouvoir le lancer, il est nécessaire de passe par les étapes suivantes:

1- Installer TinyOS-1.0

2- Accéder à: **cd /opt/tinyos-1.x/tools/java**

ET taper : **make**

3- Installer les mises à jour de NesC1.1.1 and TinyOS1.1.15.

Pour se faire, rechercher sur le net http://www.tinyos.net/dist-1.1.0/tinyos/windows/ ces mises à jour en téléchargeant le **rpm** et le mettant dans **C:\tinyos\cygwin\home\PLANETE PC**

Et taper dans le **shell**:

rpm -ivh --ignoreos nesc-1.1.2b-1.cygwin.i386.rpm

rpm -ivh --ignoreos --force tinyos-1.1.15Dec2005cvs-1.cygwin.noarch.rpm

4- Aller à **opt/tinyos-1.x/tools/java/net/tinyos/sim** et vérifier si ces fichiers sont présents: **SimObjectGenerator.java** et **MoteSimObjectGenerator.java .** S'ils existent, alors les supprimer de ce répertoire.

5- Editer le **makefile** qui est dans **C:\tinyos\cygwin\opt\tinyos-1.x\tools\java\net\tinyos\sim** et écrire cette instruction : **net/tinyos/message/avrmote/*.class**

(Voir ** makefile pour vérifier là où il faut insérer cette instruction)

6- Aller à **shell** et taper:

cd /opt/tinyos-1.x/tools/java/net/tinyos/sim

make clean

make

7- Accéder à l'application qui va être simulée. On prend par exemple, l'application Blink. Accéder au **shell** et faire:

cd opt/tinyos-1.x/apps/blink

make pc

 # tinyviz

export PATH="$TOSROOT/tools/java/net/tinyos/sim:$PATH"

TinyViz -run build/pc/main.exe 20 ///Insérer le nombre de nœuds. Par exemple 20

**** makefile:**

Voici le makefile à éditer. Dans le niveau indiqué (la ligne avec un fond coloré), insérer l'instruction **net/tinyos/message/avrmote/*.class** si elle n'existe pas:

```
SUBDIRS = event plugins packet lossy script
ROOT = ../../..
PLUGINS_SRC = $(wildcard plugins/*.java)
PLUGINS = $(PLUGINS_SRC:.java=.class)
INITIAL_TARGETS = msgs jython ../sf/old/nido/NidoSerialDataSource.class
OTHER_CLEAN = msgs-clean plugins-list-clean jarclean
# Uncomment this line to make jarfile mandatory
FINAL_TARGETS = jarfile
include $(ROOT)/Makefile.include
../sf/nido/NidoSerialDataSource.class: ../sf/old/nido/NidoSerialDataSource.java
(cd ../sf/nido; $(MAKE))
msgs:
(cd msg; $(MAKE))
msgs-clean:
(cd msg; $(MAKE) clean)
# Make sure that jython gets built
jython: $(ROOT)/org/python/core/parser.class
$(ROOT)/org/python/core/parser.class:
(cd $(ROOT)/org/python && $(MAKE))
(cd $(ROOT)/org/apache && $(MAKE))
# Create a list of default plugins
plugins/plugins.list: $(PLUGINS)
echo $(PLUGINS) > plugins/plugins.list
plugins-list-clean:
rm -f plugins/plugins.list
# This is ugly. The only way to embed a jar file inside another is to
# unpack it and repack them together into a single flat file.
jarfile: plugins/plugins.list
@echo "Creating simdriver.jar..."
(cd $(ROOT); \
jar cmf net/tinyos/sim/simdriver.manifest \
net/tinyos/sim/simdriver-tmp.jar \
net/tinyos/sim/*.class \
net/tinyos/sim/event/*.class \

net/tinyos/sim/lossy/*.class \
net/tinyos/sim/msg/*.class \
net/tinyos/sim/packet/*.class \
net/tinyos/sim/plugins/*.class \
net/tinyos/sim/script/*.class \
net/tinyos/sim/script/reflect/*.class \
```

```
net/tinyos/sim/ui \
net/tinyos/sim/plugins/plugins.list \
net/tinyos/sf/*.class \
net/tinyos/util/*.class \
net/tinyos/packet/*.class \
net/tinyos/message/*.class \
net/tinyos/message/avrmote/*.class \
org/apache/oro/text/regex/*.class \
org/python/compiler/*.class \
org/python/core/*.class \
org/python/modules/*.class \
org/python/parser/*.class \
org/python/parser/ast/*.class \
org/python/rmi/*.class \
org/python/util/*.class)
rm -rf jarbuild-tmp
mkdir jarbuild-tmp
(cd jarbuild-tmp; jar xf ../simdriver-tmp.jar; jar xf ../$(ROOT)/jars/oalnf.jar; rm -rf METAINF;
jar cmf ../simdriver.manifest ../simdriver.jar .)
rm -rf simdriver-tmp.jar jarbuild-tmp
jarclean:
rm -f simdriver.jar
```

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.

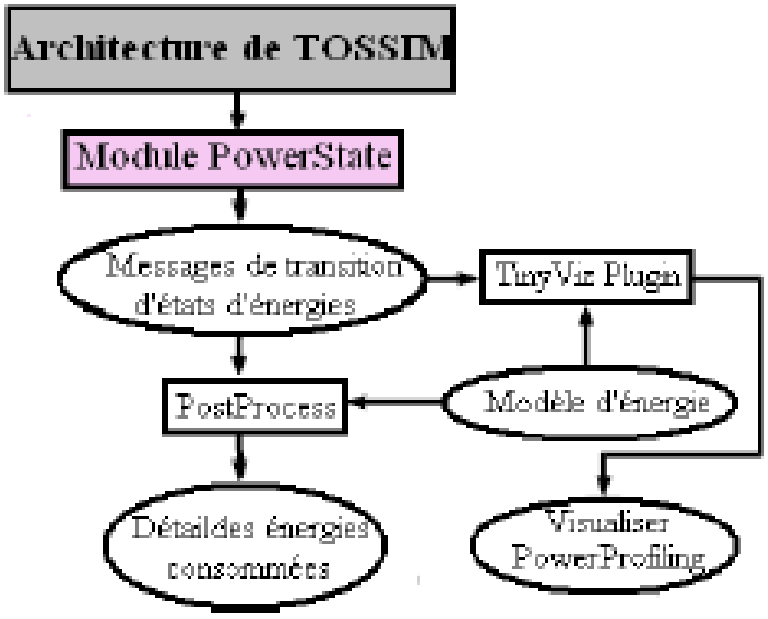


Fig. D-1: Architecture de PowerTOSSIM.

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

4- Exécuter **postprocess.py** sur la trace de simulation en spécifiant les paramètres --sb et --em **/opt/Tinyos-1.x/tools/scripts/PowerTOSSIM/postprocess.py --sb=0 --em /opt/Tinyos-1.x/tools/scripts/PowerTOSSIM/mica2_energy_model.txt NomApp.trace**

Le parameter --sb spécifie si les nœuds 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/PowerTOSSIM/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