République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Réseaux et systèmes distribués (R.S.D)*

Thème

Implémentation et évaluation des schémas de routage sur une plateforme réelle de réseaux de capteurs sans fil

Réalisé par :

- **MAAROUF Samia**
- **OUADAH Souhila**

Présenté le 25 Juin 2014 devant le jury composé de MM.

- *Mr BENMAMMAR Badr* *(Président)*
- *Mr LEHSAINI Mohamed* *(Encadreur)*
- *Mme LABRAOUI Nabila* *(Examinateur)*
- *Mr BELHOUCINE Amin* *(Examinateur)*

Année universitaire : 2013-2014

Remerciements

En préambule à ce mémoire nous remercions ALLAH qui nous aide et nous donne la patience et le courage durant ces longues années d'études.

Nous souhaitons adresser nos remerciements les plus sincères tout d'abord au corp professoral et administratif de la faculté des sciences pour la richesse et la qualité de leurs enseignements et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée ainsi qu'aux personnes qui nous ont apporté leur aide et ont contribué à l'élaboration de ce mémoire et qu'à la réussite de cette formidable année universitaire

Nous tenons à remercier sincèrement "Monsieur M.Lehsaini " qui, en tant que encadreur 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 lui ce mémoire n'aurait jamais vu le jour.

Nous expirons notre gratitude à Mr BENMAMMAR, Mme LABRAOUI et Mr BELHOUCINE d'avoir honoré notre jury de soutenance.

Dédicace

Je dédie ce modeste travail :

A mes très chers parent

Mon très cher père pour sa patience et tous ses efforts

A ma mère pour m'avoir épaulé, encouragé à reprendre les études et motivé dans les moments les plus difficiles

A ma grand-mère MEDDOUR KHADOUDJA que Dieu le Tout Puissant la garde En bon santé

A mes sœurs :SOUAAD, Nabila , ZAKIA ,AMINA

A mon frère :SALIM pour tous leur encouragements

A mon futur marie NADJI MOHAMED AMINE pour son soutien, sa compréhension et ces encouragements
A toute ma famille : MAAROUF ,BENCHOUREF , NADJI

A mon encadreur Monsieur M .Lehsaini pour l'excellence de son accompagnement et la confiance qu'il nous a accordé.

A mon binôme SOUHILA avec qui j'ai partagé de belles années d'études et avec qui j'ai eu l'honneur de les finir.

A tous mes professeurs ainsi que tous les étudiants de la promotion RSD-2014, qui nous ont accueillis si généreusement et aidés tout au long années universitaire

A mes amies : IMENE , MANEL,FATIMA ,HALIMA, HOUDA, HADJER.....

Et toute personne que je connais et qui me sont chers et tous ceux qui m'aiment.

MAAROUF Samia

Dédicace

Toutes les lettres ne sauraient trouver les mots qu'il faut, tous ces mots ne sauraient exprimer la gratitude, l'amour, le respect, et la reconnaissance à mes très chers parents qui m'ont offert sans condition leur soutien morale et financier et à qui je dois ce travail.

A mes chères sœurs Naziha, Amina, Hidayet,

et surtout Israa .

A mon frère Fethallah pour tous leur encouragements.

A mon adorable neveu Houssem Eddine et son père Hamza

A toutes mes amies avec lesquels j'ai partagé mes moments de joie, bonheur et folie : Samia, Imene, Fatima, Halima, Chahra ,Soumia,Wafaa,Asma……

A toute ma famille "Ouadah", "Medroumi"

A mon encadreur " Mr LEHASAINI".

A Belkaide Seyf Eddine pour son soutien, sa compréhension, son attention, sa patience et ces encouragements

Que toute personne n'ayant aidé de prés ou de loin, trouve ici l'expression de ma reconnaissance.

OUDAH Souhila

Table des matières

Table des figures

Table des tableaux

Introduction générale

Introduction Générale

Les technologies sans fil offrent de nouvelles perspectives dans le domaine des télécommunications et des réseaux informatiques. Grâce aux progrès faits, il est apparu un nouveau type de réseaux ad-hoc, qui sont les Réseaux de Capteurs Sans fil (RCSF). Ce sont des réseaux sans infrastructure fixe, ils peuvent être déployés de façon rapide dans des zones sensibles et/ou difficilement accessibles. Leur mission est le plus souvent de surveiller une zone, de prendre régulièrement des mesures et de faire remonter des alarmes vers certains nœuds du réseau, appelés nœuds collecteurs, capables de relayer l'information à grande échelle vers un centre de contrôle distant.

Les caractéristiques intrinsèques de cette nouvelle génération de micro-capteurs (miniaturisation, capacité de traitement, communication sans fil, diversité des capteurs (optiques, thermiques, multimédias, etc.), faible coût, etc.) ont ouvert de nouvelles perspectives applicatives très larges et très variées pour les réseaux de capteurs dans de nombreux domaines (militaires, domotiques, environnementales, etc.). Toutefois, ils soulèvent, dans les mêmes proportions, de nombreuses problématiques de recherche tant par les applications potentielles qu'ils laissent entrevoir que par les diverses contraintes qu'ils imposent. Il est communément admis que les techniques et approches développées aussi bien dans le cadre des réseaux filaires que dans celui des réseaux sans fil ne sont pas directement «transposables» dans le domaine des réseaux de capteurs. En définitif, les RCSF restent un domaine ouvert pour la communauté scientifique et ce quasiment à tous les niveaux : auto-configuration, localisation, couverture, déploiement, communication, topologie dynamique, collecte et dissémination des données, requêtes et traitement, etc.

La majorité des travaux sur les réseaux de capteurs vise à réduire la consommation énergétique, ou du moins sa rationalisation. En effet, les réseaux de capteurs sont destinés, le plus souvent, à relever des informations dans des environnements hostiles ou difficilement accessibles sans aucune intervention humaine. Il est donc difficilement envisageable de trouver une autre source d'énergie que celle des batteries. C'est pour cette raison qu'on les considère comme des dispositifs autonomes. Leur durée de vie est par conséquent égale à la durée de vie de leur batterie. En outre, la consommation énergétique est donc la contrainte clef dans les réseaux de capteurs puisque l'énergie est considérée comme une ressource

précieuse. Dans cette optique, les travaux ont concerné toutes les couches de la couche physique jusqu'à la couche application.

Dans ce travail, nous nous intéressons à plusieurs modèles d'interaction entre les capteurs et la station de base. Ces modèles sont représentés par des schémas de routage différents. Dans cette optique, nous avons évalué les performances de plusieurs modèles d'acheminement des données vers la station de base sur une plateforme réelle de capteurs TelosB pour tirer profit du modèle adéquat à ce type de réseaux et comparer avec les résultats de simulation présentés dans la littérature. Dans le premier modèle, nous avons considéré un schéma de routage basé sur une architecture à plat. Dans le deuxième modèle, nous avons évalué un schéma de routage basé sur une architecture clustérisée. Dans ce modèle, nous avons considéré deux scénarios : dans le premier scénario il y avait la formation des clusters dans lesquels les clusterheads communiquent directement avec la station de base comme dans LEACH (Low Energy Adaptive Clustering Hierarchy) [1] alors que dans le deuxième scénario l'acheminement des données pourra se faire soit directement quand la station de base est proche du clusterhead correspondant soit via un chemin de CH-à-CH comme dans CDS (Connected Dominating Set) [2,3]. Dans le deuxième modèle, nous assistons à une agrégation de données.

Ce manuscrit est organisé en trois chapitres. Dans le premier chapitre, nous présentons des généralités sur les réseaux de capteurs sans fil. Le deuxième chapitre est une présentation des outils matériels et logiciels nécessaires à la réalisation de notre application. Le troisième chapitre constitue le cœur de notre travail. Dans ce chapitre, nous présentons une architecture clustérisée permettant de réaliser une application de l'agrégation et nous évaluons les performances des schémas de routage décrits dans le paragraphe précédent. Cette évaluation se fait sur une plateforme réelle de réseaux de capteurs. Enfin, nous concluons notre travail en présentant les résultats obtenus et en donnant quelques perspectives.

Chapitre I

Généralités sur les RCSF

Chapitre I
Généralités sur les réseaux de capteurs sans fil

I.1 Introduction

Les progrès dans le domaine de l'électronique miniaturisée et les communications sans fil ont donné naissance à des composants capables de prélever des grandeurs environementales, physiologiques etc. Ces composants sont appelés des nœuds capteurs et ils ont la capacité de s'auto-organiser pour former un réseau de capteurs sans fil (RCSF).

Les RCSF permettent de faciliter le suivi et le contrôle à distance de l'environnement physique avec une meilleure précision. Ils peuvent aussi être déployés pour exploiter diverses applications (environnementales, militaires, médicales, etc). En outre, un réseau de capteurs est constitué généralement d'un grand nombre de noeuds capteurs car ces derniers sont sujets à pannes accidentelles ou intentionnelles. Chaque noeud est composé principalement d'un ou plusieurs capteurs, d'une unité de traitement et d'un module de communication. Ces noeuds communiquent entre eux selon une certaine topologie du réseau afin d'acheminer les informations à un centre de contrôle distant de la zone de leur déploiement. La mise en place d'un RCSF pose de nombreux problèmes parmi lesquels le routage des informations vers la station de base via les différents noeuds du réseau. Dans cette optique plusieurs contributions ont été proposées dans la littérature. Ces contributions visent à minimiser la consommation d'énergie ceci afin d'optimiser l'autonomie des noeuds qui constituent le réseau et par suite garantir une longue longétivité pour le réseau entier.

Dans ce chapitre, nous présentons les réseaux de capteurs sans fil avec un plan méthodologique que nous avons adopté. Nous commençons par une définition d'un capteur, son architecture, ses types et voir comment ces derniers sont déployés pour former un réseau de capteurs sans fil. Ensuite, la topologie, les différents facteurs de conception et les spécificités des RCSF seront étudiés,ainsi que les domaines d'application des réseaux de capteurs sans fil et une petite conclusion.

I.2 Les capteurs

I.2.1 Définition d'un capteur

Un capteur est un dispositif ayant pour tâche de transformer une mesure physique observée en une mesure généralement électrique qui sera à son tour traduite en une donnée binaire exploitable et compréhensible par un système d'information.

Parmi les différents types de mesures enregistrées par les capteurs, on peut citer entre autres : la température, l'humidité, la luminosité, l'accélération, la distance, les mouvements, la position,la pression, la présence d'un gaz, la vision (capture d'image), le son, etc. . .

La notion de capteur s'est évoluée avec le temps puisque leur domaine d'application s'est élargi. Les premiers capteurs n'étaient dédiés qu'à un unique type de mesure, les capteurs contemporains sont la combinaison de plusieurs dispositifs capables de mesurer différentes mesures physiques. En outre, à ces possibilités de mesures multiples, les capteurs actuels ont vu se gèrer des fonctionnalités qui leur permettent, en plus de l'enregistrement et de la détection d'événements mesurables, le traitement de ces données et leur communication vers un autre dispositif. On parle alors de capteur intelligent, capable à la fois de mesurer des données et de les communiquer avec d'autres capteurs au sein d'un réseau, tel qu'il est caractérisé par sa capacité à effectuer une collecte des mesures, les traiter et à les communiquer au monde extérieur [4].

I.2.2 Architecture d'un capteur

Dans cette section, nous distinguons les deux parties qui composent un capteur :

a) Architecture matérielle

La figure I-1 est l'illustration la plus générale de l'architecture d'un capteur dit intelligent.

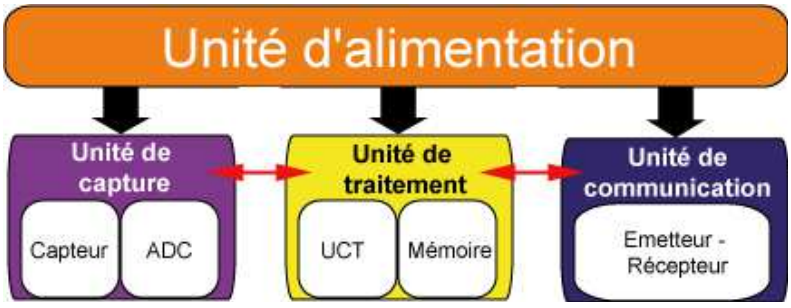

Figure I-1: Architecture d'un capteur sans fil [5]

Cette architecture s'articule autour de quatre unités :

- **l'unité de traitement :** c'est l'unité principale du capteur. Elle est généralement représentée par un processeur couplé à une mémoire vive. Son rôle est de contrôler le bon fonctionnement des autres unités. Sur certains capteurs elle peut embarquer un système d'exploitation pour faire fonctionner le capteur. Elle peut aussi être couplée à une unité de stockage, qui servira par exemple à y enregistrer les informations transmises par l'unité d'acquisition de données.
- **l'unité d'acquisition :** elle permet la mesure des grandeurs physiques ou analogiques et leur conversion en données numériques. Elle est composée du capteur lui-même et de l'ADC[1] qui permet la conversion des données. Le capteur est chargé de récupérer les signaux analogiques qu'il transmet à l'ADC qui a pour rôle de transformer et de communiquer les données analogiques en données numériques compréhensibles pour l'unité de traitement.
- **l'unité de communication :** elle a pour fonction de transmettre et recevoir l'information. Elle est équipée d'un couple émetteur/récepteur pour communiquer au sein du réseau. Il existe cependant d'autres possibilités de transmission (optique, infrarouge, etc. . .).
- **l'unité d'alimentation :** c'est un élément primordial de l'architecture du capteur, c'est elle qui fournit en énergie toutes les autres unités. Elle correspond le plus souvent à une batterie ou une pile alimentant le capteur, dont les ressources limitées en font une problématique propre à ce type de réseau puisque ces derniers sont généralement déployés dans des zones non accessibles. La réalisation récente d'unité d'alimentation à base de panneaux solaires tente d'apporter une solution pour prolonger sa durée de vie [6].

Par ailleurs, un capteur peut être doté d'autres unités. Citons, entre autres, la possibilité d'ajouter une unité de localisation, tel qu'un GPS, une unité de mobilité pour assurer la mobilité du capteur, ou une unité spécifique de capture comme une caméra pour de l'acquisition vidéo.

Dans le cas de l'utilisation d'un GPS ou d'une caméra de surveillance, il est intéressant de noter que leur utilisation dans les capteurs actuels a un coût non négligeable en termes de consommation énergétique, qui peut réduire grandement la durée de vie d'un capteur équipé de ce type de dispositifs.

[1] ADC : Analog-to-Digital Converters

b) Architecture Logicielle

La contrainte énergétique des capteurs exige l'utilisation de systèmes d'exploitation légers tels que TinyOS [7] ou Contiki [8]. Cependant, TinyOS reste toujours le plus utilisé et le plus populaire dans le domaine des RCSF. Il est libre et est utilisé par une large communauté de scientifiques dans des Simulations pour le développement et le test des algorithmes et protocoles réseau.

I.2.3 Types de capteurs

Il existe actuellement un grand nombre de capteurs, avec des fonctionnalités diverses et variées. Tous ces différents capteurs ne pourraient être décrits ici, cependant une liste exhaustive peut être trouvée sur le site The Sensor Network Museum [9].

La plupart des capteurs dépendent de l'application pour lesquels ils ont été conçus (capteur aquatique, sous-terrain, etc. . .). Il est plus intéressant de décrire les capteurs les plus utilisés et leur évolution au cours du temps.

En l'occurrence, la figure I-2 illustre l'évolution des capteurs au cours de ces 20 dernières années. Cette représentation met en avant l'importance des travaux de recherche de l'université de Berkeley dans l'essor des réseaux de capteurs, surtout sachant que l'entreprise Xbow (aussi appelé Crossbow) qui fait jusqu'à aujourd'hui office de référence dans la fabrication de capteurs est née au sein de la célèbre université californienne.

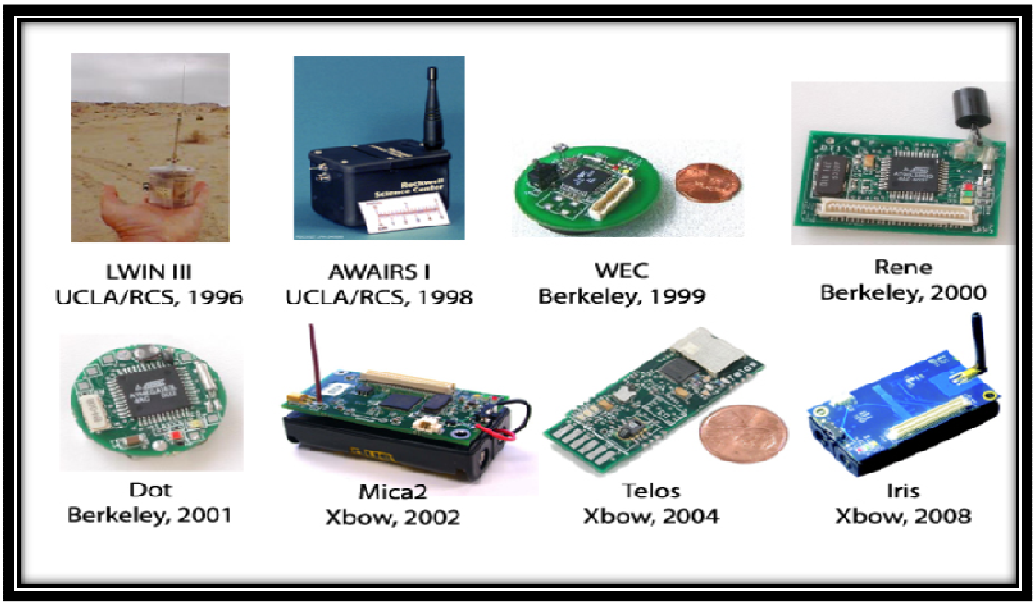

Figure 0-2: Quelques exemples de capteurs [5]

Les capteurs fabriqués par Xbow au cours des dix dernières années (famille de capteurs Mica et Telos) sont sans aucun doute les plus utilisés dans les expériences et travaux de recherche. Ces capteurs sont capables de mesurer plusieurs métriques (température, humidité, luminosité, etc. . .) et s'articulent pour la plupart d'entre eux autour du Chipcon CC2420 qui est devenu le standard au niveau des modules de transmission utilisant le protocole de communication IEEE 802.15.4.

I.2.4 Caractéristiques des capteurs

Pour illustrer les caractéristiques des capteurs, on présente un type de chaque famille : MicaZ de la famille Mica et TelosB de la famille Telos comme montre dans la figure I-3. Le tableau 1.1 résume les caractéristiques de ces capteurs.

Capteur TelosB **Capteur MicaZ**

Figure I-3: Capteurs TelosB et MicaZ [10]

Tableau I-1 : Caractéristiques de quelques exemples de capteurs

Caractéristiques	TelosB	MicaZ
Flash	48Kb	512Kb
RAM	10Kb	4Kb
Antenne	Radio Chipcon Wireless Transceiver	Antenne Chipcon CC2420
MCU	1MHz T1 MPS430	8MHz ATmega128
Dimension	65x31x6 Mm	58x32x7 Mm
2 piles AA	Radio+cpu :75mW Sleep mode : 140µW	Radio+cpu mode :63mW Sleep mode : 30µW
External flash	512kB	512kB

I.3 Réseaux de capteurs sans fil (RCSF)

I.3.1 Description des réseaux de capteurs

Un Réseau de Capteurs Sans Fil (RCSF) est un type particulier de réseaux mobiles ad hoc MANETs[2]. Il est composé d'un ensemble de dispositifs très petits, nommés nœuds capteurs, variant de quelques dizaines d'éléments à plusieurs milliers. Dans ces réseaux, chaque nœud est capable de surveiller son environnement 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 [11].

I.3.2 Architecture de communication dans les RCSF

Le processus d'acheminement de l'information des capteurs à la station de base peut prendre quatre formes. Dans les architectures à plat, les capteurs peuvent communiquer directement avec la station de base en utilisant une forte puissance (figure 1.4 (a)), ou via un mode multi-sauts avec des puissances très faibles (figure 1.4 (b)), alors que dans les architectures hiérarchisées, le nœud représentant le cluster, appelé cluster-head, transmet directement les données à la station de base (figure 1.4 (c)), ou via un mode multi-saut entre les clusterheads (figure 1.4 (d)) [12].

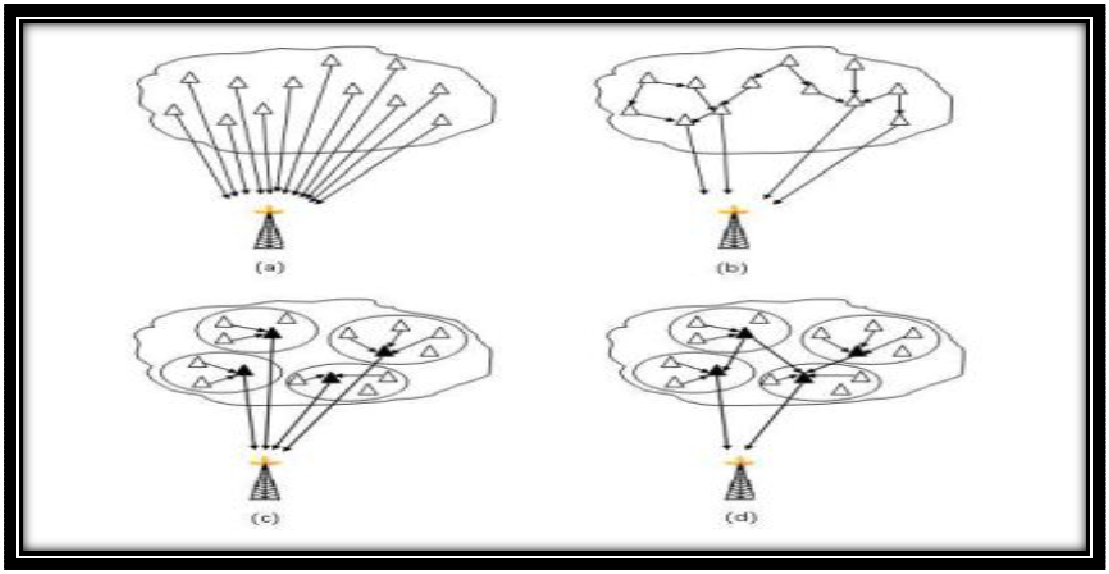

Figure 0-4: Architecture de communication dans les RCSF [12]

[2] MANETs : Mobile Ad hoc NETworks

I.3.3 Spécificités des RCSF

- **Durée de vie** : 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 à plusieurs années selon l'emplacement de la zone d'intérêt.
- **Topologie dynamique** : Le déploiement d'un grand nombre de nœuds nécessite une maintenance de la topologie. Cette maintenance consiste en trois phrases : déploiement, post-déploiement (les capteurs peuvent bouger, ne plus fonctionner,…), redéploiement de nœuds additionnels.
- **Routage des données dans un RCSF** : Pour limiter le nombre de communications coûteuses en énergie, les réseaux de capteurs sans fil requièrent des protocoles de routage efficaces. Une des solutions employées par les protocoles de routage est la clustérisation, qui divise le réseau en plusieurs clusters. Dans chacun de ces clusters, un nœud maître (clusterhead) est élu et aura pour mission de récupérer les informations des nœuds du cluster dont il a la charge pour les transmettre aux autres clusters et inversement. Le choix du nœud maître dans un protocole de routage va être fait en désignant par exemple le nœud avec l'énergie la plus importante, pour augmenter la durée de vie du réseau.

  La figure I-5 représente un exemple de réseau clustérisé où les nœuds A, B et C ont été respectivement élus clusterhead des clusters 1, 2 et 3.

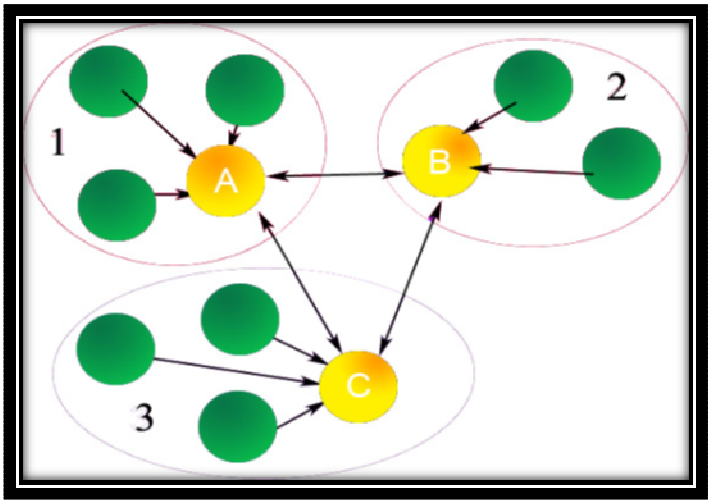

Figure 0-5: Clustérisation d'un RCSF [5]

- **Tolérance aux fautes, adaptabilité et fiabilité dans un RCSF :** Les réseaux de capteurs sont requis pour fonctionner en s'adaptant aux changements environnementaux que les

capteurs contrôlent. Les réseaux devraient être capables de s'auto-organisés en fonction des facteurs environnementaux. La fiabilité est la capacité de maintenir les fonctionnalités de réseau de capteurs sans la moindre interruption qui sera due à l'échec du nœud capteur. Ce dernier peut échouer en raison du manque d'énergie, de dommages physiques, de problèmes de communication, d'inactivité, ou d'interférence environnementale. De ce fait, le réseau devrait pouvoir détecter l'échec d'un nœud et s'organiser, se reconfigurer et récupérer des échecs de nœud sans desserrer aucune information.

- **Passage à l'échelle dans un RCSF :** Le nombre de capteurs utilisés dans les réseaux de capteurs sans fil peut varier de quelques entités à plusieurs dizaines de milliers. C'est d'ailleurs la principale utilité des réseaux de capteurs qui doivent pouvoir s'auto-organiser à une grande échelle et être efficace quel que soit leur nombre. Pour cela les protocoles des réseaux de capteurs sans fil doivent être capables de fonctionner et de s'adapter selon le nombre de nœuds.
- **Puissance de calcul dans un RCSF** : Malgré les progrès récents dans la fabrication de capteurs de plus en plus puissants, les capteurs actuels souffrent d'un manque de puissance de calcul (par exemple seulement 16 Mhz de puissance et 128 Koctets de mémoire programmable pour un capteur MicaZ [13]). Cette faible puissance ne permet pas d'utiliser des algorithmes complexes, et particulièrement des algorithmes cryptographiques gourmands en ressources CPU. De plus, la vocation des capteurs sans fil est d'être en très grand nombre et leur utilisation dans des applications avec un nombre de nœuds élevé nécessite l'utilisation de capteurs bon marché, ce qui implique des capteurs avec une puissance de calcul très faible. La faiblesse de puissance de calcul est aussi préjudiciable pour le temps de réponse du réseau. Si l'on demande à un capteur d'effectuer de nombreux calculs, la latence va sensiblement augmenter.
- **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. Or, un débit de transmission réduit n'est pas handicapant pour un réseau de capteurs où les fréquences de transmission ne sont pas importantes.
- **Sécurité** : En fonction de l'application, la sécurité peut être critique. Le réseau devrait permettre la détection des intrusions pour assurer un fonctionnement correct contre les mauvaises manipulation ou attaques. L'écoute, le brouillage, et les attaques de

retransmission peuvent entraver ou empêcher l'opération. Par conséquent, le contrôle d'accès, l'intégrité des messages, et la confidentialité doit être garanti.

- **Qualité de Service** : La qualité de service se réfère à la capacité du réseau à fournir des données fiable et à temps. Un grand nombre de service, à savoir, le débit ou la capacité de transport, ne sont pas généralement suffisant pour satisfaire un délai requis par une application, par conséquent, la vitesse de propagation de l'information peut être aussi cruciale que le débit. En plus de la capacité du réseau, de nombreux travaux importants dans les réseaux sans fil de capteurs se font pour la garantie de la qualité de service (QoS), essentiellement sur le délai. Par exemple, dans certaines les applications de contrôle en temps réel, la valeur de l'information dégrade rapidement quand la latence augmente.
- **Energie d'un RCSF :** Les capteurs sont équipés de batteries, comme par exemple des piles LR6 dans le cas des MicaZ ou TelosB. L'énergie de ces batteries est limitée (plusieurs jours à quelques mois). De plus, les RCSF quand ils sont déployés, le sont souvent dans des zones difficiles d'accès pour l'homme et les capteurs sont en général déployés pour ne plus être modifiés. Il devient alors inenvisageable de vouloir changer les batteries des capteurs. Si le nombre de capteurs dépasse la centaine d'entités, il est encore plus difficile d'intervenir pour trouver le capteur défaillant et changer sa batterie. La consommation de l'énergie des réseaux de capteurs sans fil doit être la plus faible possible. Dans ce but, les capteurs actuels ont des périodes de veille durant leur inactivité pour préserver leur batterie. Enfin les communications sont les actions qui coûtent le plus cher en termes d'énergie. Pour cela, il est donc fortement nécessaire de limiter le nombre de communications entre capteurs.

I.3.4 Agrégation de données dans un RCSF

Diffuser les données captées sur le réseau peut facilement encombrer ce dernier. Certaines applications critiques comme les contrôleurs des centrales nucléaires exigent une transmission pressante et un traitement plus rapide des données qui peuvent dégrader l'exécution et perdre la fiabilité due à la congestion ou à la latence dans le réseau. L'agrégation intelligente des données captées et l'élimination de l'information non désirée et redondante ainsi que la compression des données peuvent être une solution pour l'utilisation efficace de ressources et d'énergie et l'évitement de la congestion dans le réseau. Plusieurs algorithmes comme la diffusion dirigée [14] ont été proposés pour faciliter l'agrégation et la diffusion de données dans le contexte des RCSF.

I.3.5 Applications des RCSF

Le domaine d'applications des RCSF est très varié. Ces réseaux sont présents dans le domaine militaire, sécurité civile, médical, transport, environnemental,…etc

- **Applications militaires :** Les RCSF permettent la détection des mouvements ennemis sur un champ de bataille ou bien de tracer leurs mouvements. De façon analogue, ils peuvent permettre la détection d'intrusion ou de cambriolage dans le domaine de la sécurité civile.
- **Applications environnementales :** Les réseaux de capteurs permettent la détection des incendies, la surveillance des catastrophes naturelles, la détection des pollutions et le suivi des écosystèmes.
- **Applications agricoles :** Les capteurs peuvent être semés avec les graines. Ainsi, les zones sèches seront facilement identifiées et l'irrigation sera donc plus efficace et économique.
- **Applications médicales :** Les RCSF permettent par exemple la surveillance de l'état de santé des patients qu'ils soient au sein de l'établissement ou même ailleurs, et ce en permanence.
- **Applications de transport :** Les RCSF permettent la surveillance des réseaux ferroviaires ou routiers ainsi que d'éventuelles intrusions.
- **Applications domestiques :** En plaçant, sur le plafond ou dans le mur, des capteurs, on peut économiser l'énergie en gérant l'éclairage ou le chauffage en fonction de la localisation des personnes.

Figure I-6: Quelques domaines d'applications des RCSF [15]

I.4 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 vu la diversité de ces applications : santé, environnement, industrie et même dans le domaine sportif.

Dans ce premier chapitre, nous avons présenté les RCSF, leurs spécificités et les concepts nécessaires à la compréhension des réseaux de capteurs. Cependant, nous avons remarqué que plusieurs facteurs et contraintes compliquent la gestion de ce type de réseaux. En effet, les réseaux de capteurs se caractérisent par une capacité énergétique limitée rendant l'optimisation de la consommation d'énergie dans des réseaux pareils une tâche critique pour prolonger la durée de vie du réseau. Nous avons a mis le point sur quelques facteurs permettant l'économie de l'énergie tels que la manière d'acheminer de données de telle sorte que le nombre de messages redondants soit minimisé.

Dans le chapitre qui suit, nous présentons les outils matériels et logiciels qui permettent la mise en place de certains schémas de routage conçus pour les RCSF.

Chapitre II

Outils logiciels pour les RCSF

Chapitre II
Outils logiciels pour les RCSF

II.1 Introduction

Pour pouvoir travailler sur un réseau de capteurs sans fil, il est nécessaire d'avoir un environnement dédié à celui-ci, l'utilisation des systèmes d'exploitation traditionnels comme Windows ou Linux est impossible car ils ont un grand nombre de fonctionnalités inutiles pour un réseau de capteurs [7]. C'est pourquoi dans ce chapitre, nous allons utiliser des outils qui prennent en compte les spécificités des capteurs telles qu'une mémoire réduite, un processeur moins rapide, etc. Parmi ces outils, nous utilisons les systèmes d'exploitation conçus pour les systèmes embarqués et spécialement pour la programmation des capteurs. Parmi, ces systèmes nous optons pour TinyOS qui est le plus populaire dans la communauté scientifique. En outre, nous faisons appel à NesC [16] qui est un langage orienté composant et qui a une relation étroite avec TinyOS. En plus, pour une meilleure représentation de nos résultats nous utilisons le langage Java.

Dans ce chapitre, nous présentons brièvement ces outils dans le but de faciliter la compréhension de la partie développement de l'application.

II.2 Systèmes d'exploitation

Les systèmes d'exploitation pour les réseaux de capteurs sans fil tels que TinyOS [7], Contiki [8] et MantisOS [17] sont conçus pour répondre à ces contraintes. Les possibilités matérielles des capteurs sont limitées en mémoire, en puissance de calcul et en autonomie énergétique (batterie). Ces systèmes d'exploitation ont une « faible empreinte mémoire » lors de leurs embarquement dans la mémoire flash des capteurs.

Dans cette section, nous présentons avec plus de détails TinyOS que nous avons utilisé pour le développement de nos applications.

II.2.1 Le système d'exploitation : TinyOS

a) Présentation de TinyOS

TinyOS est un système d'exploitation open source développé et suivi par l'université de Berkeley. Ce système d'exploitation a été conçu pour les réseaux de capteurs sans fil car un capteur n'a pas assez de mémoire pour supporter un système d'exploitation comme Linux ou Windows qui prennent beaucoup de place mémoire.

TinyOS a été créé pour répondre aux caractéristiques et aux nécessités des réseaux de capteurs, telles que [18] :

- Une taille de mémoire réduite
- Une basse consommation d'énergie.
- Des opérations d'assistance intensives et robustes.
- Il est optimisé en termes d'usage de mémoire et d'énergie.

TinyOS est un système d'exploitation basé sur un fonctionnement évènementiel ; c'est-à-dire, il devient actif lorsqu'un événement se produit (par exemple la réception d'un message), dans le cas contraire les nœuds capteurs sans en état de veille. Ce processus permet de mieux économiser les ressources énergétiques des capteurs. Le langage de programmation pour la conception de ce système est NesC qui est presque similaire au langage C. Pour autant, la bibliothèque des composants de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données. L'ensemble de ces composants peut être utilisé tel quel et il peut aussi être adapté à une application bien précise [19].

b) Concepts de TinyOS

Un composant est constitué d'au moins un module utilisant et fournissant des interfaces. S'il contient plusieurs modules les liens entre eux sont décrits par un fichier de configuration. Une application complète est un composant contenant plusieurs modules liés entre eux dont un module Main qui permet de les démarrer.

TinyOS offre une centaine de composants que l'on peut utiliser pour développer des applications. Quand le programme est généré par le compilateur, seuls les composants utilisés (y compris ceux du système) sont présents. Main est lui-même connecté à certains

composants du système (comme l'ordonnanceur par exemple) qui seront donc chargés en mémoire puis lancés par Main.

Pour une telle application, plusieurs fichiers dont l'extension "**.**nc" peuvent la composer. Ces fichiers peuvent être des modules, des interfaces ou des configurations.

c) Propriétés de TinyOS

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

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

Lorsqu'un système est dit "temps réel" celui-ci gère des tâches caractérisées par des priorités et par des échéances à respecter dictées par l'environnement externe. Dans le cas d'un système strict, aucune échéance ne tolère de dépassement contrairement à un système temps réel. TinyOS se situe au-delà de ce second type car il n'est pas prévu pour avoir un fonctionnement temps réel. Il a été conçu pour réduire au maximum la consommation en énergie du capteur. Ainsi, lorsqu'aucune tâche n'est pas active, il se met automatiquement en veille. Le tableau II-1 résume les principales propriétés de TinyOS.

Tableau 0-1: Les propriétés de TinyOS [20]

Propriété	Valeur
Type	Event-driven
Disponibilité	Open-source
Langage	NesC
Préemptif	Non
Temp réel	Non
Sources	Fournies

d) Allocation de la mémoire

Il est important de préciser de quelle façon un système d'exploitation aborde la gestion de la mémoire. C'est encore plus significatif lorsque ce système travaille dans un espace restreint. TinyOS a une empreinte mémoire très faible puisqu'il ne prend que 300 à 400 octets dans le cadre d'une distribution minimale. En plus de cela, il est nécessaire d'avoir 4 Ko de mémoire libre qui se répartissent de la façon suivante :

- **La pile :** sert de mémoire temporaire au fonctionnement du système notamment pour l'empilement et le dépilement des variables locales.
- **Les variables globales :** réservent un espace mémoire pour le stockage de valeurs pouvant être accessible depuis des applications différentes.
- **La mémoire libre :** pour le reste du stockage temporaire.

La gestion de la mémoire possède de plus quelques propriétés. Ainsi, il n'y a pas d'allocation dynamique de mémoire et pas de pointeurs de fonctions. Bien sûr cela simplifie grandement l'implémentation. Par ailleurs, il n'existe pas de mécanisme de protection de la mémoire sous TinyOS ce qui rend le système particulièrement vulnérable aux crashs et corruptions de la mémoire [11].

e) Allocation de ressources

Le choix d'un ordonnanceur pour TinyOS déterminera le fonctionnement global du système et le dotera de propriétés précises telles que la capacité à fonctionner en temps réel.

L'ordonnanceur TinyOS c'est :

- **2 niveaux de priorité** (bas pour les tâches, haut pour les évènements) puisque les tâches prennent plus de temps d'exécution qu'un événement.
- **1 file d'attente FIFO** (disposant d'une capacité de 7).

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

f) Package TinyOS

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

- la version (Tinyos-1.x) : elle présente moins de risques mais elle est nettement moins récente.
- la version (TinyOS-2.x) : supporte un grand de plateformes des capteurs.

II.2.2 Autres systèmes d'exploitation pour les RCSF

Il existe d'autres systèmes d'exploitation dédiés aux réseaux de capteurs, nous citons dans ce qui suit les plus répandus :

- Contiki [8] : c'est un système d'exploitation open-source multitâche, développé pour les systèmes embarqués avec contraintes de mémoire.
- SOS [21] : système d'exploitation développé par l'université de Los Angeles en Californie écrit en langage C et qui reprend l'aspect événementiel de TinyOS.
- Mantis OS [17] : système d'exploitation dédié aux réseaux de capteurs, développé par l'université du Colorado (USA) et écrit en langage C. Contrairement à TinyOS qui est basé sur un modèle de programmation événementielle, Mantis OS s'articule autour d'un modèle commandé par l'exécution de processus.
- Nut/OS [22] : Système d'exploitation multitâche pour les systèmes embarqués avec une pile TCP/IP.

II. 3 Langage de programmation NesC

Le langage NesC utilise une architecture basée sur des composants, celle-ci vise à réduire la taille mémoire du système et les applications développées. Une application est considérée comme un ensemble de composants ayant un but précis. Chaque composant correspond à un élément matériel (LEDs, timer, ADC, etc) et peut être réutilisé dans différentes applications [22].

Un composant est constitué de trois éléments essentiels :

- **Les interfaces :** Spécifient un ensemble de fonctions à mettre en application par le fournisseur des interfaces (commandes) et par l'utilisateur des interfaces (événements). Ces fonctions sont précédées par des mots-clés respectifs "command" ou "event". 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 redéfinit son code. De plus, tous les composants possèdent l'interface StdControl car sa tâche est l'initialisation, le démarrage et l'arrêt des composants.
- **Les modules :** Définissent les éléments de base de la programmation. Ils utilisent une ou plusieurs interfaces. Par ailleurs, il est à noter que le processus d'exécution repose sur les tâches et les mécanismes d'interruption. De ce fait, les modules permettent aussi d'implémenter ces tâches.
- **Les configurations :** Constituent un ensemble de modules et d'interfaces ainsi que des liaisons entre les composants de l'application déployée dans les capteurs.

II.4 Java

JAVA a été développé par Sun au début des années 90 dans une filiation avec le langage C et surtout le langage objet C++ mais dans une optique de plus grande portabilité d'une machine à une autre et d'une plus grande fiabilité. Les programmes JAVA sont compilés en "bytecode", un langage intermédiaire indépendant de la plateforme.

Dans notre implémentation, nous avons choisi JAVA car il permet la création des interfaces graphiques grâce à sa librairie Swing et aussi parce que c'est le seul langage dont le code exécutable est portable.

II.5 MIG (Message Interface Generator)

TinyOs fournit des outils pour produire automatiquement des objets message des descriptions de paquet. Ainsi, au lieu d'analyser les formats de paquet manuellement, l'outil de MIG établit une interface Java à la structure de message. En effet, pour une séquence d'octets donnée, le générateur de code MIG devra analyser automatiquement chacun des champs des paquets et fournit un ensemble d'accédant et de mutators standards pour visualiser les paquets reçus ou produire des nouveaux paquets.

II.5 Conclusion

Dans ce chapitre, nous avons présenté le système d'exploitation TinyOS, avec ses propriétés, caractéristiques, sa structure logicielle, et nous avons parlé un peu sur l'allocation de mémoire et de ressources par la plateforme TinyOS. Nous avons également présenté le langage de programmation NesC qui est créé pour la conception des applications embarquées et plus précisément pour les réseaux de capteurs sans fil.

Dans le chapitre suivant, nous allons aborder la partie implémentation de notre application.

Chapitre III
Evaluation des schémas de routage pour les RCSF

Chapitre III

Evaluation des schémas de routage pour les RCSF

III.1 Introduction

Les capteurs sont généralement déployés dans des zones hostiles là où l'accès humain est difficile voire impossible. En outre, ces derniers disposent d'une autonomie d'énergie puisque les batteries sont généralement irremplaçables et non rechargeables. De ce fait, il faudrait instaurer des mécanismes d'acheminement de l'information des capteurs jusqu'à la station de base pour une longue durée et maximiser par la suite la durée de vie du réseau.

Dans les réseaux de capteurs, la communication de l'information à la station de base peut se faire de différentes manières selon l'architecture utilisée. Par exemple, dans une architecture à plat, tout capteur qui reçoit l'information pour la première fois, la diffuse dans son voisinage et ce processus se répète jusqu'à l'aboutissement à la station de base. Cependant, dans une architecture clustérisée, la communication de l'information se fait soit directement par les nœuds agrégateurs appelés aussi clusterheads comme dans LEACH soit d'agrégateur à agrégateur jusqu'à la station de base.

Dans ce chapitre, nous évaluons les performances de ces architectures sur une plateforme réelle de capteurs de type telosb. La première consiste en une architecture à plat dans laquelle les nœuds permettent l'acquisition de données et de relayer les données des voisins. Cependant dans la deuxième, il s'agit d'une architecture clustérisée dans laquelle il y a des nœuds qui sont désignés comme des clusterheads et qui ont comme rôle l'agrégation de données et la communication de la donnée agrégée à la station de base. Dans la troisième, il s'agit aussi d'une architecture clustérisée dans laquelle la communication de la donnée agrégée ne se fait pas directement à la station de base par le nœud agrégateur mais ce dernier pourrait être un nœud relayeur de l'information ou a besoin d'un autre nœud agrégateur pour relayer sa donnée agrégée.

Dans ce qui suit, nous détaillons les éléments matériels et logiciels qui ont une relation avec notre application.

III.2 Les choix techniques

L'implémentation de notre application a nécessité l'utilisation de différents outils matériels qui sont les capteurs telosb ainsi que des outils logiciels bien spécifiques aux réseaux de capteurs sans fil, tels que NesC qui est un langage orienté composant, TinyOs qui est un système d'exploitation dédié aux équipements à ressources limitées et Java pour une meilleure présentation des résultats recueillis par les capteurs.

III.2.1 Outils matériels

Nous avons développé notre application en utilisant : des capteurs de type TelosB, un PC portable de type HP, des piles de type AA pour alimenter les capteurs, et un câble filaire qui relie le port USB de la station de base à celui du PC. La figure III.1 présente la plateforme matérielle utilisée.

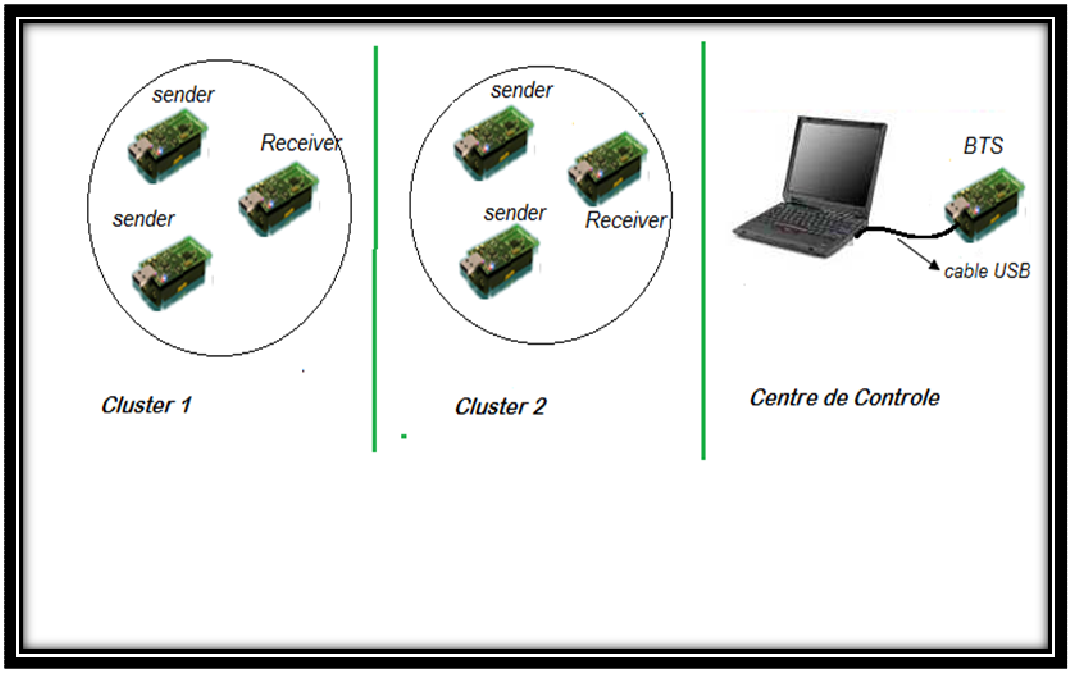

Figure III-1: Plateforme matérielle utilisée

III.2.2 Outils logiciels

Dans cette partie, nous présentons les outils logiciels nécessaires pour le développement de notre application. Pour cela, nous avons utilisé Ubuntu12.10, TinyOS2.1.2, NesC et Java.

Pour développer une application pour les réseaux de capteurs sans fil, il est nécessaire d'avoir un environnement dédié à ceux-ci l'utilisation d'un système d'exploitation léger tels

que TinyOS ou Contiki et non pas des systèmes d'exploitation traditionnels comme Windows ou Linux car ces derniers nécessitent un grand espace mémoire pour leur mise en place. C'est pour cette raison nous avons opté pour TinyOS qui est populaire pour les systèmes embarqués et les dispositifs à ressources limitées. En outre, pour l'implémentation de notre l'application nous avons adopté deux langages de programmations NesC et Java. Le premier est un langage orienté composant pour programmer les dispositifs à ressources limitées tels que les capteurs, et le deuxième est un langage orienté objet utilisé pour interpréter et bien exploiter ce qui se passe au centre de contrôle et pour communiquer les données envoyées par les capteurs.

D'autre part, pour réaliser l'interface graphique notre choix s'est porté sur JAVA car il permet la création des interfaces graphiques grâce à sa librairie Swing et aussi parce que c'est le seul langage dont le code exécutable est portable.

III.3 Les étapes de développement de l'application

Le développement de notre application s'articule autour de trois parties : l'installation logicielle, l'installation matérielle, et l'architecture de l'application.

III.3.1 Installation logicielle

Cette étape nous a permis de se familiariser avec les capteurs telosb et le langage NesC ainsi que le système d'exploitation TinyOS.

L'installation du système d'exploitation TinyOS 2.x sous Ubuntu a été la phase la plus délicate. En effet, nous avons commencé par installer TinyOS 2.1.2, NesC et le microcontrôleur MSP430 pour les capteurs de type TelosB. La procédure d'installation de TinyOs se déroule en plusieurs étapes et elle est décrite dans l'annexe. Puis, nous avons installé NetBeans 6.9.1 au niveau du poste de contrôle pour communiquer avec les capteurs et faire visualiser les valeurs des grandeurs remontées à la station de base.

III.3 .2 Installation matérielle

Une fois l'installation logicielle terminée, il a fallu installer le matériel : une station de base reliée à l'ordinateur via un câble USB, six capteurs de type TelosB pour la maquette. Chacun des TelosB communique avec la station de base via une liaison sans fil et la station de base communique avec l'ordinateur via le câble USB.

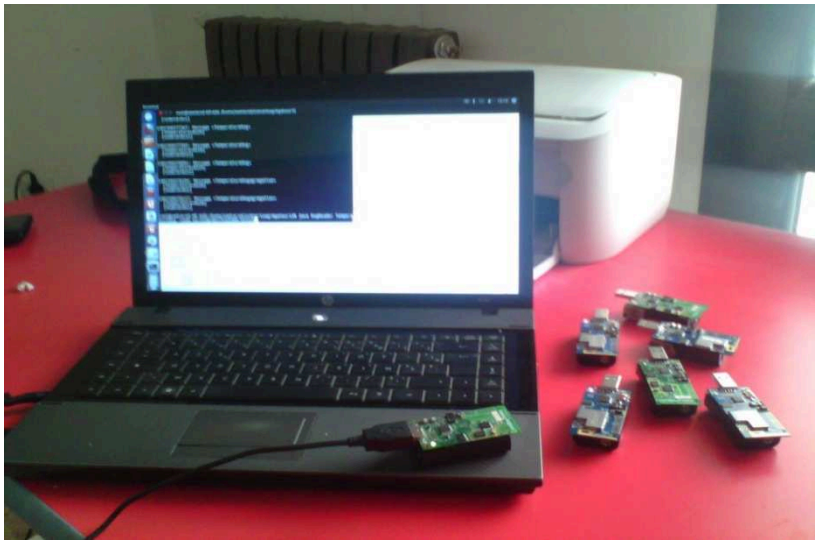

Figure III-2: Environnement du travail

III.3.3 Architecture de l'application

L'architecture consiste en trois scénarios : architecture à plat, architecture clustérisée avec des nœuds agrégateurs en contact direct avec la station de base et architecture clustérisée avec des nœuds agrégateurs qui peuvent être également des nœuds relais.

a) Scénario 1 : Architecture à plat

Les étapes suivantes expliquent le fonctionnement de l'architecture proposée :

- Les capteurs qui composent le réseau ont le même rôle. Chacun d'eux mesure périodiquement la température dans son voisinage et la communique à ces voisins. En outre, chacun de ces voisins la communique dans son voisinage s'il la reçoit pour la première fois.
- Ce processus se répète jusqu'à l'aboutissement à la station de base qui est attachée au centre de contrôle représenté par un PC.
- L'application au niveau du PC affiche les températures relayées par les Senders à chaque fois qu'une nouvelle grandeur est communiquée à la station de base.

La figure III-3 illustre une architecture à plat dans un environnement restreint là où les nœuds capteurs peuvent communiquer directement avec la station de base.

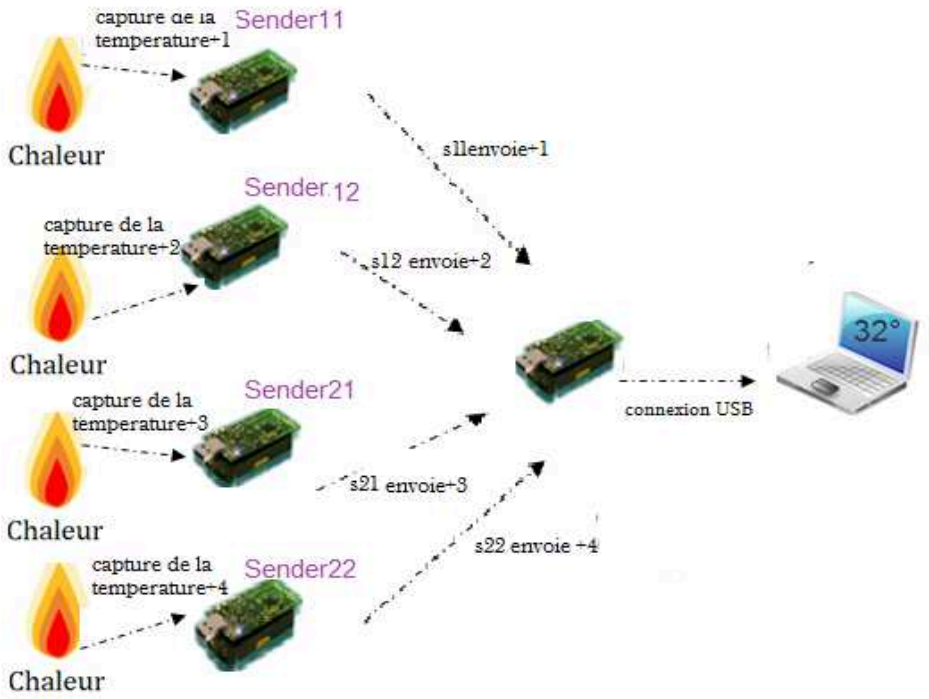

Figure 0-3: Scénario 1: Architecture à plat

b) Scénario 2 : Architecture Clustérisée 1

Cette architecture s'articule autour de deux clusters dans lesquels il y avait la désignation d'un clusterhead au niveau de chaque cluster. Son fonctionnement se résume comme suit :

- Nous avons formé deux clusters et nous avons choisi un clusterhead au niveau de chaque cluster.
- Les nœuds membres au niveau de chaque cluster mesurent périodiquement les températures dans leurs zones de couverture.
- Les nœuds membres transmettent les mesures et leurs identifiants au clusterhead correspondant qui jouera le rôle d'un nœud agrégateur dans cette l'architecture.
- Le clusterhead agrège les données reçues de la part de ces membres en calculant la moyenne de ces mesures qu'il a reçues, puis à son tour il retransmettra directement comme dans LEACH [1] la valeur de la moyenne à la station de base rattachée au centre de contrôle où réside l'application.
- L'application affiche les températures reçues des différents nœuds membres et la moyenne calculée respectivement par les deux clusterheads.

La figure III-4 illustre le fonctionnement d'un routage de données basée sur une architecture clustérisée de type 1.

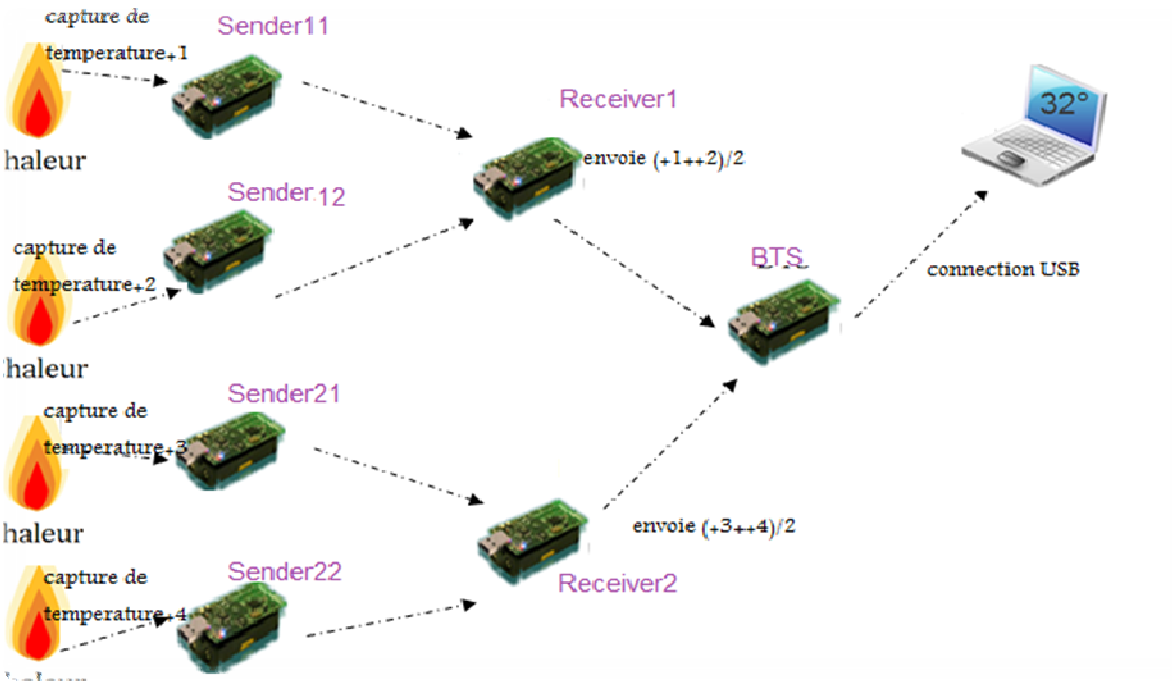

Figure 00-4: Scénario 2: Architecture clustérisée 1

c) Scénario 3 : Architecture clustérisée 2

Cette architecture est similaire à la précédente en termes de topologie c'est-à-dire il s'articule autour de deux clusters mais les nœuds agrégateurs ne communiquent pas directement avec la station de base quand ces derniers sont éloignés d'elle. Dans ce cas ils impliquent des nœuds agrégateurs qui vont jouer le rôle de nœuds relais comme dans les architectures basées sur l'approche du CDS [2,3].

Les étapes suivantes expliquent le fonctionnement de l'architecture proposée :

- Dans le premier cluster (C_1) :
  - Les nœuds membres mesurent périodiquement les températures dans leurs zones de couverture. Puis ils les transmettent à leur clusterhead correspondant (CH_1) qui agrège ces données en une seule.
  - Le clusterhead (CH_1) effectue la moyenne des mesures qu'il les a reçues et à son tour il retransmettra la valeur agrégée au clusterhead du deuxième cluster (CH_2).
- Dans le deuxième cluster (C_2) :

- Les nœuds membres mesurent périodiquement les températures dans leurs zones de couverture.
- Ils transmettent les mesures et leurs identifiants correspondants au clusterhead (CH_2) qui jouera le rôle d'un nœud agrégateur principal dans l'architecture puisqu'il se trouve plus proche de la station de base.
- Le clusterhead (CH_2) agrège les mesures qu'il a reçues (les valeurs mesurées par les nœuds membres du cluster C_2 et la valeur agrégée par le clusterhead CH_1 du cluster C_1). Puis CH_2 retransmettra la nouvelle valeur agrégée à la station de base rattachée au centre de contrôle où réside l'application.
- L'application affiche les températures reçues des différents nœuds membres et la valeur agrégée par le clusterhead CH_2.

La figure III-5 illustre l'architecture clustérisée 2 contenant des clusterheads qui peuvent être à la fois des nœuds agrégateurs et des nœuds relais.

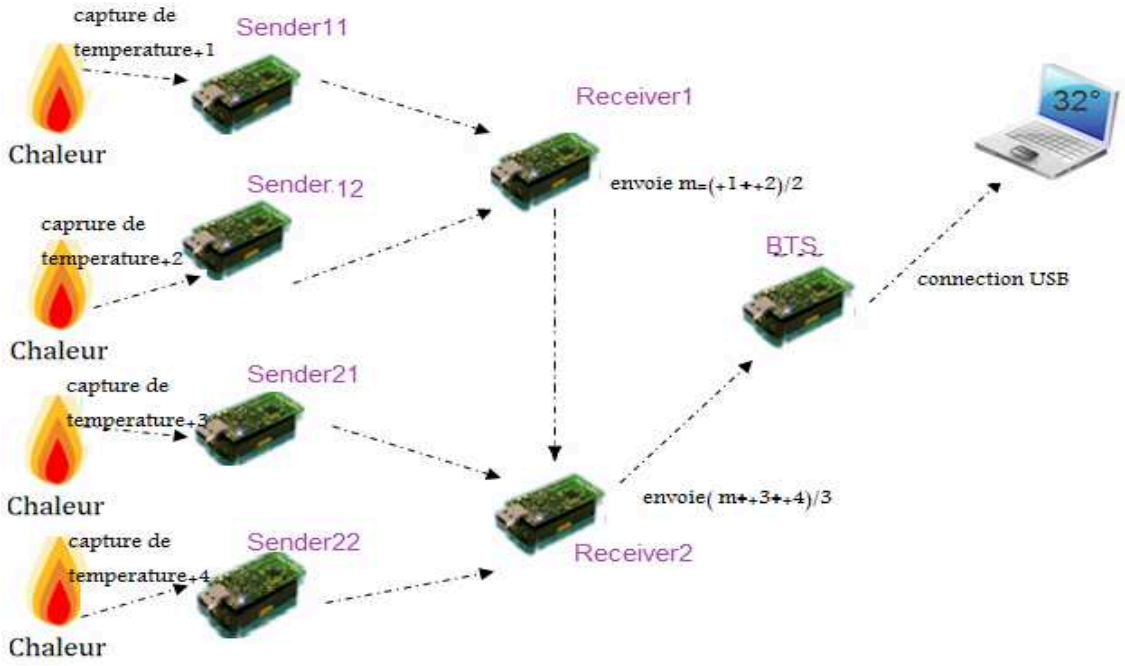

Figure III-5: Scénario 3 : Architecture clustérisée 2

III.4 Exemples d'exécution

Dans ce qui suit, nous présentons les implémentations réalisées dans le cadre de notre projet. Au début, nous avons développé une application qui permet d'une part l'agrégation de données par les clusterheads et d'autre part qui réalise un routage de type CH-à-CH jusqu'à la

station de base c'est-à-dire un clusterhead peut être agrégateur et de même relayeur de l'information agrégée des autres agrégateurs. Deuxièmement, pour mettre en valeur chaque processus de routage cité dans la section précédente, nous avons réalisé une application qui permet d'évaluer les performances de chacune de ces processus de routage.

III.4.1 Première partie : illustration de l'agrégation de données

Dans cette partie, nous avons développé une application qui concrétise un processus de routage basé sur l'architecture clustérisée avec des nœuds agrégateurs et relais :

a) Premier scénario

Dans cette partie, l'application consiste à afficher les valeurs des températures reçues des différents membres des deux clusters et la valeur agrégée de ces températures. Après l'exécution nous avons obtenu l'affichage des deux températures captées par les nœuds membres de chaque cluster. Puis, nous assistons à l'envoi d'une valeur agrégée résultante de la moyenne des deux membres du cluster. Cette valeur est transmise directement vers la station de base.

La valeur agrégée dans le premier cluster = (30°+30°) /2 qui donne 30° et la valeur agrégée dans le deuxième cluster = (33°+30°) /2 qui donne 31° comme le montre la figure III-6.

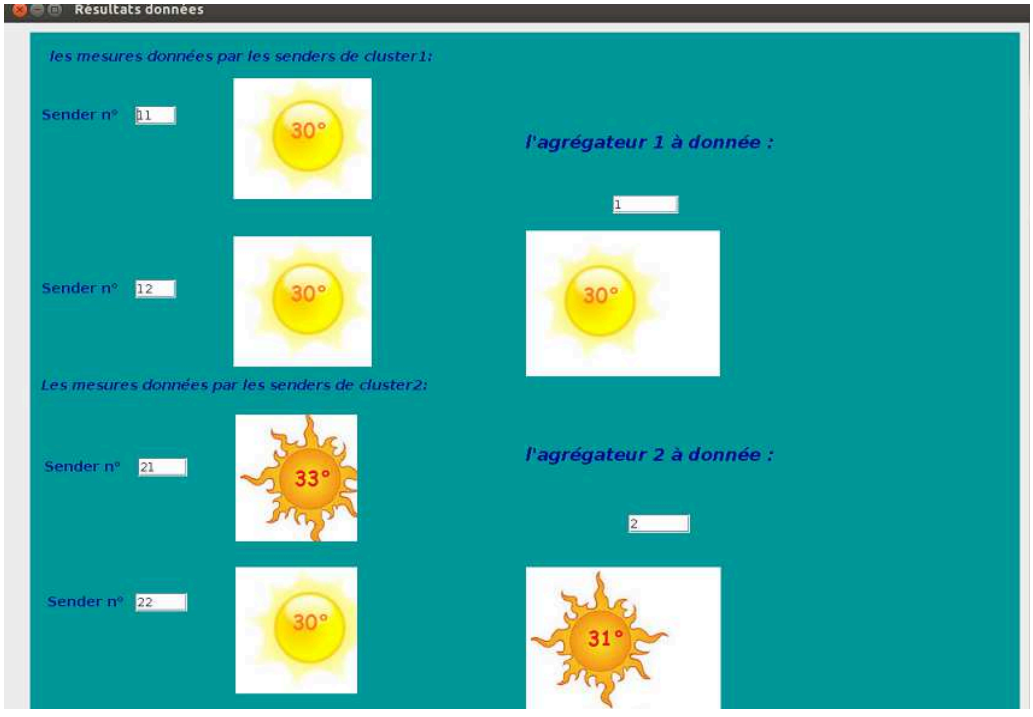

Figure 0-6: Déploiement d'une architecture clustérisée

b) Deuxième scénario

Dans cette partie, l'application consiste à afficher les valeurs des températures reçues des membres et la valeur agrégée de ces températures par chaque clusterhead dans les deux clusters. Après l'exécution nous avons obtenu l'affichage des deux températures captées par les membres du premier cluster dont les identifiants sont respectivement 11 et 12. Puis, nous assistons à l'envoi d'une valeur agrégée résultante de la moyenne des deux nœuds membres, Cette valeur est transmise vers le clusterhead CH_2 du deuxième cluster (C_2), et en même temps nous avons obtenu l'affichage des deux températures captées par les nœuds membres du deuxième cluster (C_2) qui ont pour identifiants 21 et 22. Enfin, il y a l'envoi de la valeur agrégée résultante de la moyenne des deux nœuds membres et de la valeur agrégé par le premier agrégateur (CH_1). Cette valeur est transmise vers la station de base.

Exemple :

Valeur agrégée = (32°+30°+31°) /3 qui donne 31° comme le montre la figure III-7.

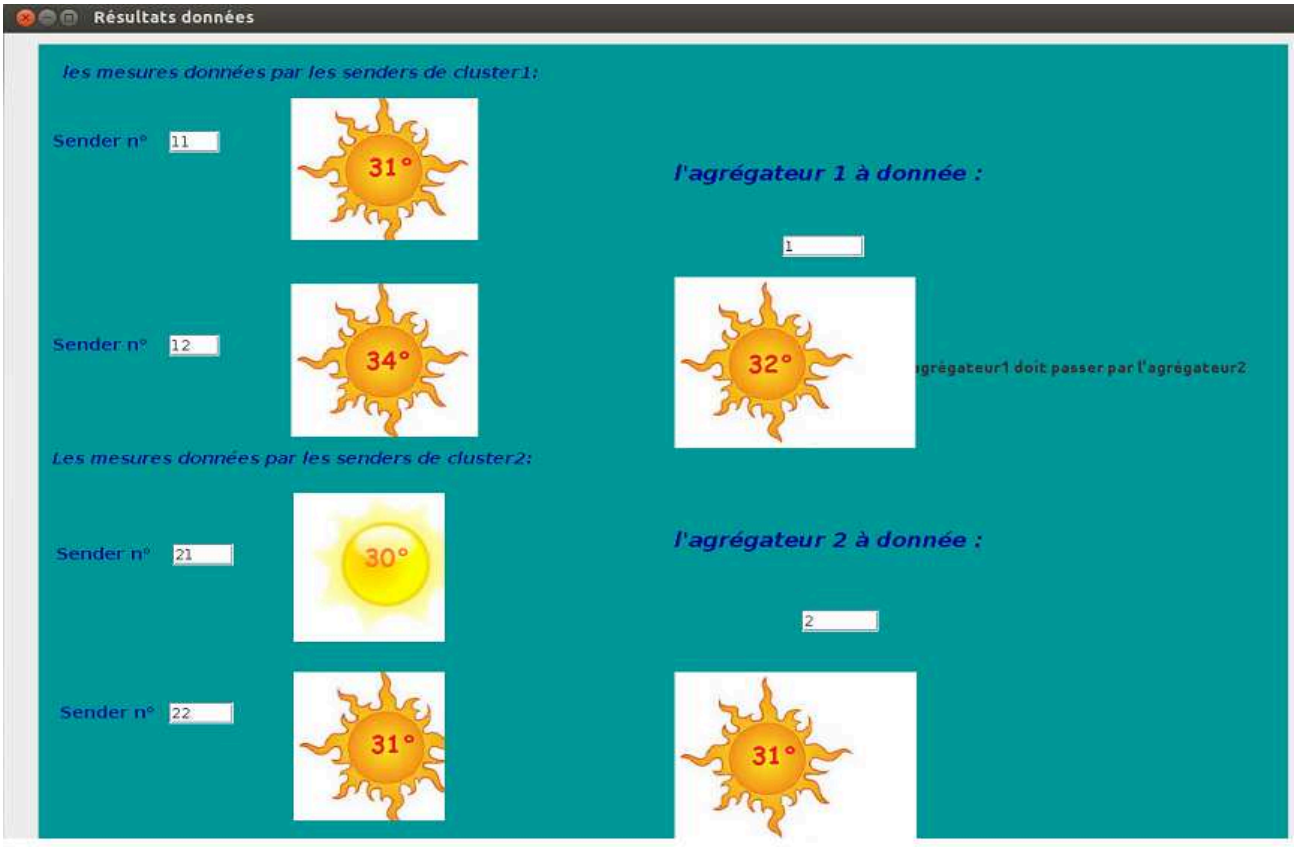

Figure III-7: Architecture clustérisée avec deux agrégateurs actifs

Dans la figure III-8 le premier clusterhead (CH_1) est désactivé. Par suite, l'application consiste à afficher les valeurs des températures reçues des nœuds membres des deux clusters mais nous assistons à l'envoi d'une valeur agrégée résultante de la moyenne des deux valeurs de températures captées par les deux membres 21 et 22 du deuxième cluster par le deuxième clusterhead qui joue le rôle d'un agrégateur et d'un relayeur de l'information agrégée par CH_1 dans le réseau considéré. Dans notre cas, nous avons l'affichage même des températures des membres du premier cluster car ces membres peuvent atteindre la station de base par un simple saut. Cependant, dans la réalité ces nœuds membres sont éloignés de la station de base et seulement leur clusterhead correspondant qui est responsable de l'envoi de leurs captures.

Exemple :

Valeur agrégée = (30°+30°) /2 qui donne 30° comme le montre la figure III-8.

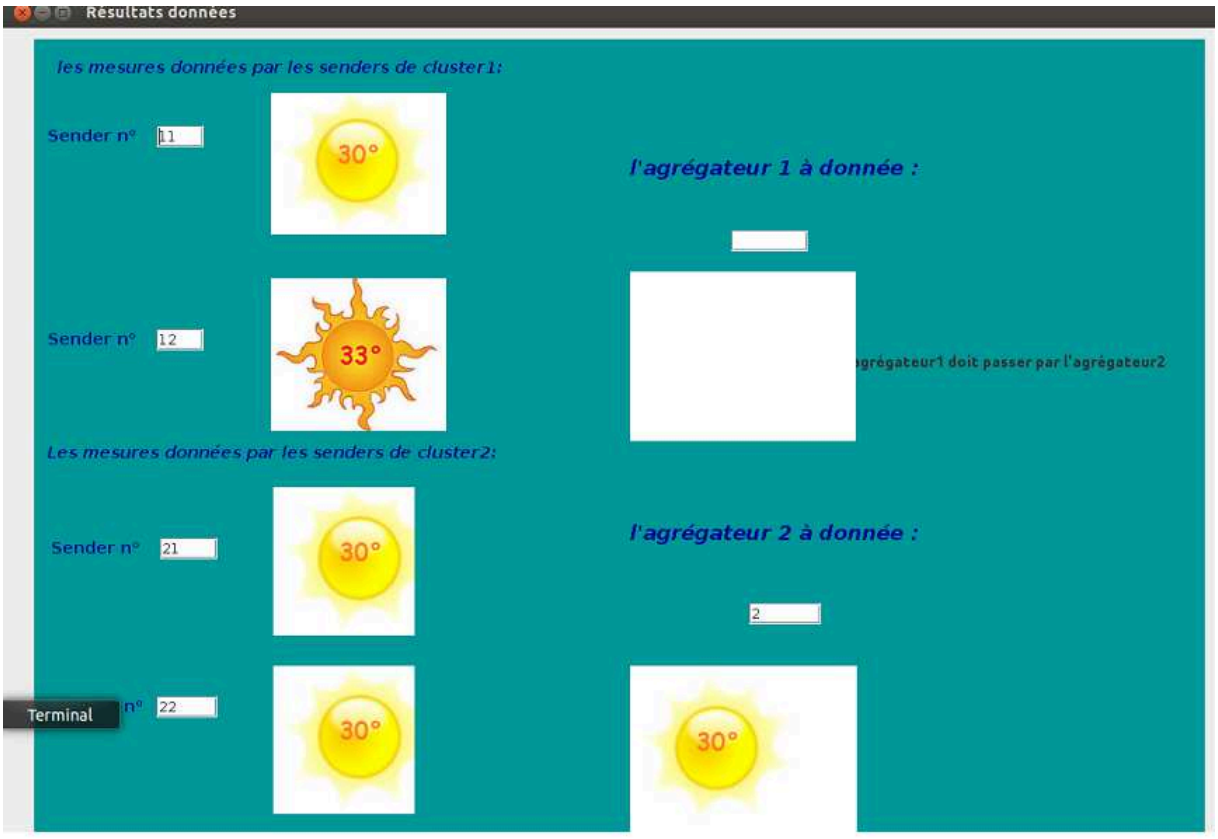

Figure 0-8: Architecture clustérisée avec un seul CH qui est actif

Pour la figure III-9, le deuxième clusterhead (CH_2) qui joue le rôle d'agrégateur principal (le plus proche à la station de base), est désactivé. De ce fait, l'application consiste à afficher les valeurs des températures reçues des membres des deux clusters mais aucune

valeur agrégée n'est envoyée, parce que la valeur agrégée par le premier clusterhead doit passer par le deuxième clusterhead qui est désactivé

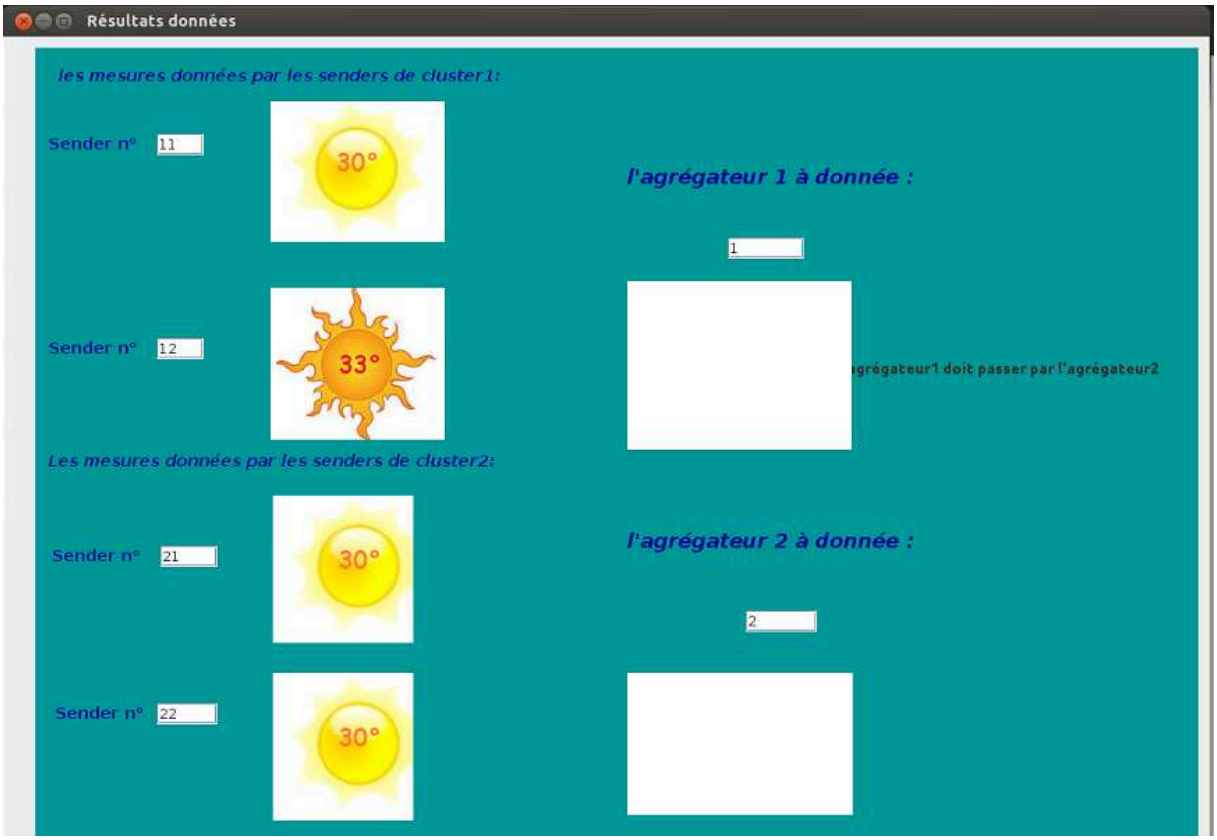

Figure III-9: Architecture clustérisée avec seulement CH1 actif

III.4.2 Deuxième partie : Evaluation des performances

Pour illustrer l'apport d'une architecture clustérisée en particulier en termes de consommation d'énergie puisque l'énergie est une ressource précieuse dans les réseaux de capteurs, nous avons mis en place deux architectures : une à plat et l'autre clustérisée. Dans la première, les nœuds envoient des paquets directement à la station de base après chaque seconde. Dans la deuxième architecture, nous avons mis en place deux scénarios et dans chacun d'eux nous avons utilisé deux clusters tels que dans chaque cluster a deux nœuds membres et clusterhead. Chaque seconde, les nœuds membres envoient des paquets au clusterhead correspondant qui joue le rôle d'un nœud agrégateur. Dans le premier scénario chaque clusterhead envoie sa valeur agrégée directement à la station de base alors que dans le deuxième scénario le clusterhead du premier cluster envoie sa valeur agrégée au clusterhead du deuxième cluster et ce dernier agrège les valeurs de ces nœuds membres et la valeur agrégée par le clusterhead du premier cluster, et envoie directement la valeur résultante à la station de base.

a) Architecture à plat

Dans une architecture à plat, tout nœud qui reçoit l'information pour la première fois la diffuse dans son voisinage et ses voisins procèdent de la même manière. Les figures III-10 et III-11 illustrent le nombre de paquets envoyés et reçus dans le réseau pendant une minute et cinq minutes et le tableau III-1 résume le nombre de paquets envoyés et reçus pendant des périodes différentes.

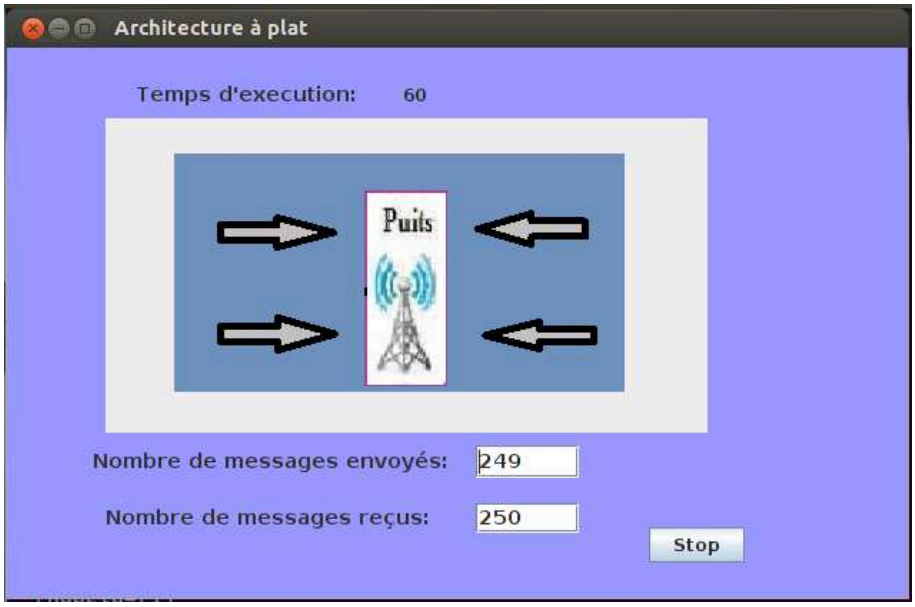

Figure III-10: Architecture à plat après une minute

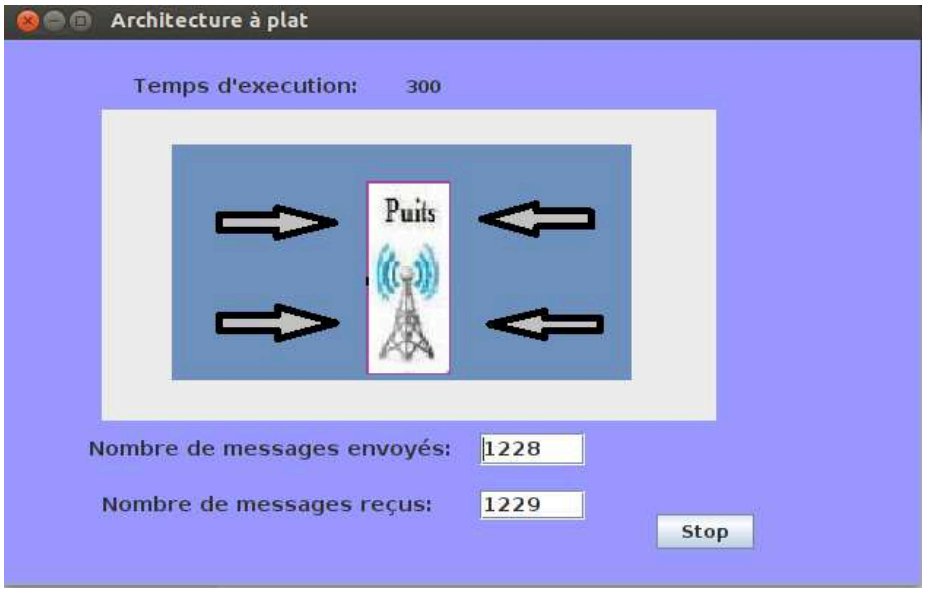

Figure III-11: Architecture à plat après cinq minutes

Tableau 0-1: Résultats de l'architecture à plat

Temps (Minutes)	*1*	*2*	*3*	*4*	*5*
Nbre de paquets envoyés	249	498	745	986	1228
Nbre de paquets reçus	250	499	745	986	1229

La première ligne de ce tableau représente le temps, la deuxième le nombre de paquets transmis des nœuds capteurs à la station de base, et la troisième le nombre de paquets reçus par la station de base.

Nous remarquons que le nombre de paquets reçus est égal au nombre de paquets envoyés puisque la taille du réseau est réduite. Cependant, si le nombre de nœuds composant est grand, le nombre de paquets reçus sera inférieur au nombre de paquets envoyés puisqu'il y aura des paquets perdus.

b) Architecture clustérisée 1

Comme nous avons expliqué précédemment, ce scénario décrit le fonctionnement de l'architecture clustérisée avec deux agrégateurs qui envoient leurs données agrégées directement avec la station de base.

Les figures III-12 et III-13 représentent respectivement l'état de l'exécution après 1 minute et 5 minutes.

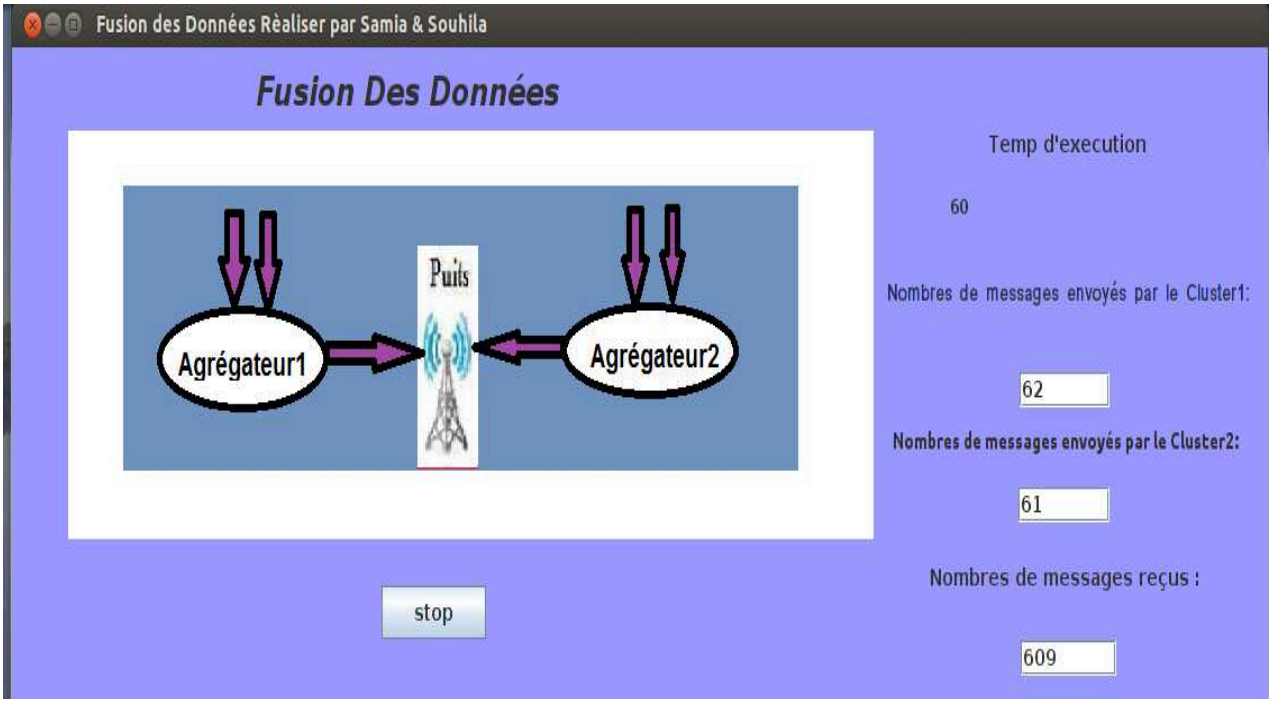

Figure III-12 : Architecture clustérisée pour état 1

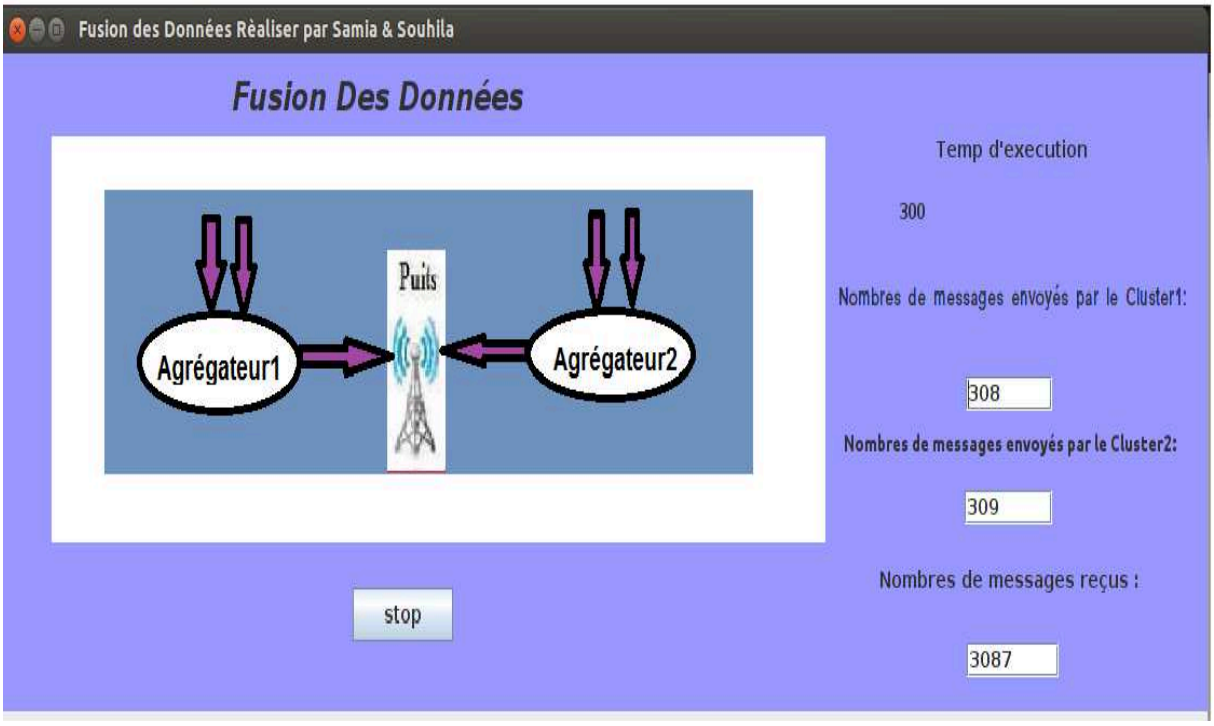

Figure III-13 : Architecture clustérisée 1 pour état 5

Le tableau III-2 récapitule les résultats obtenus dans les premiers scénarios pendant différentes périodes.

Tableau 0III-2: Résultats de l'architecture clustérisée 1

Temps (Minutes)	*1*	*2*	*3*	*4*	*5*
Nbre de paquets envoyés par CH_1	62	125	184	248	308
Nbre de paquets envoyés par CH_2	61	124	184	248	309
Nbre de paquets reçus	609	1249	1840	2475	3087

La première ligne du tableau III-3 représente le temps durant lequel nous avons supervisé la zone d'intérêt, la deuxième le nombre de paquets transmis par les nœuds membres au nœud agrégateur (CH_1) dans le premier cluster, la troisième le nombre de paquets transmis par les nœuds membres au nœud agrégateur (CH_2) dans le deuxième cluster, et la quatrième le nombre de paquets reçus par la station de base.

Nous remarquons que le nombre de paquets reçus est égal à cinq fois le nombre de paquets envoyés ce qui permet d'éviter la redondance des informations quand des neouds agrégateurs sont impliqués dans l'acheminement des données. En outre, si nous faisons référence à l'idée citée dans l'introduction qui dit que si nous réduisons le nombre de messages envoyés, la consommation de l'énergie sera minimisée.

c) Architecture clustérisée 2

Ce scénario représente une architecture clustérisée avec deux nœuds agrégateurs tels que les données du premier agrégateur (CH_1) sont relayées par le deuxième agrégateur (CH_2) et ce dernier communique directement avec la station de base.

Les figures III-14 et III-15 représentent respectivement l'état de l'exécution après 60 secondes et 5 minutes dans un réseau selon l'architecture décrite précédemment. Le tableau III-3 résume les résultats obtenus le réseau est déployé selon cette architecture.

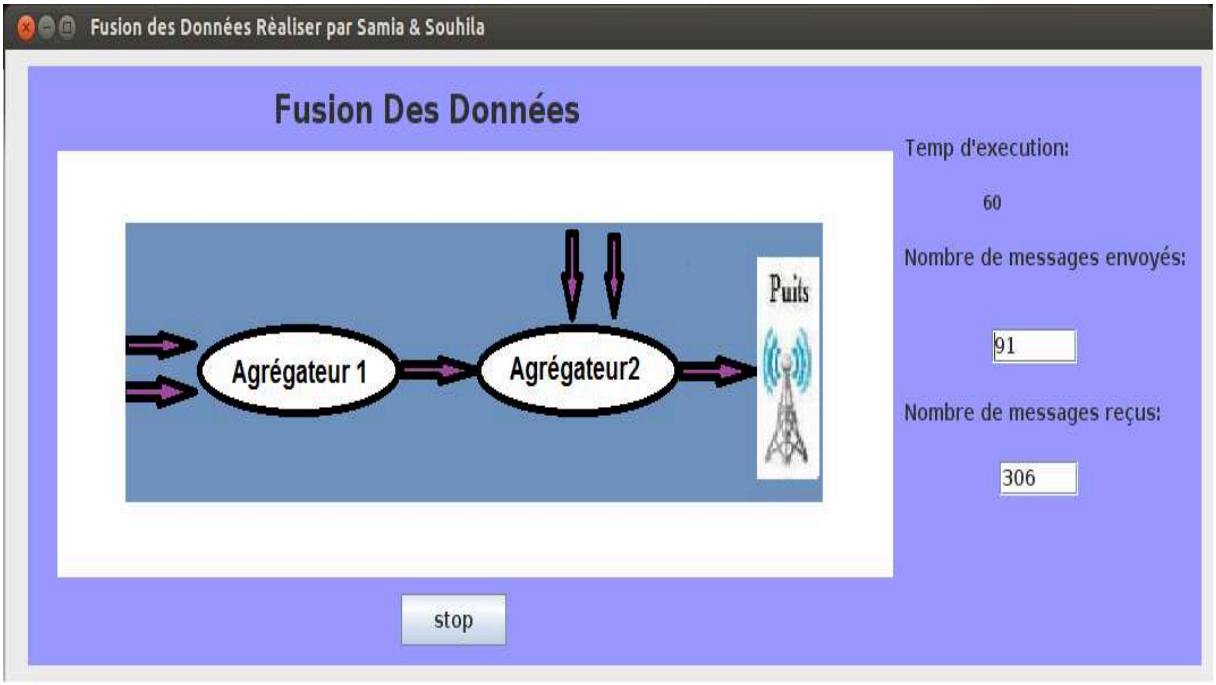

Figure 0III-14 : Architecture clustérisée 2 après une minute

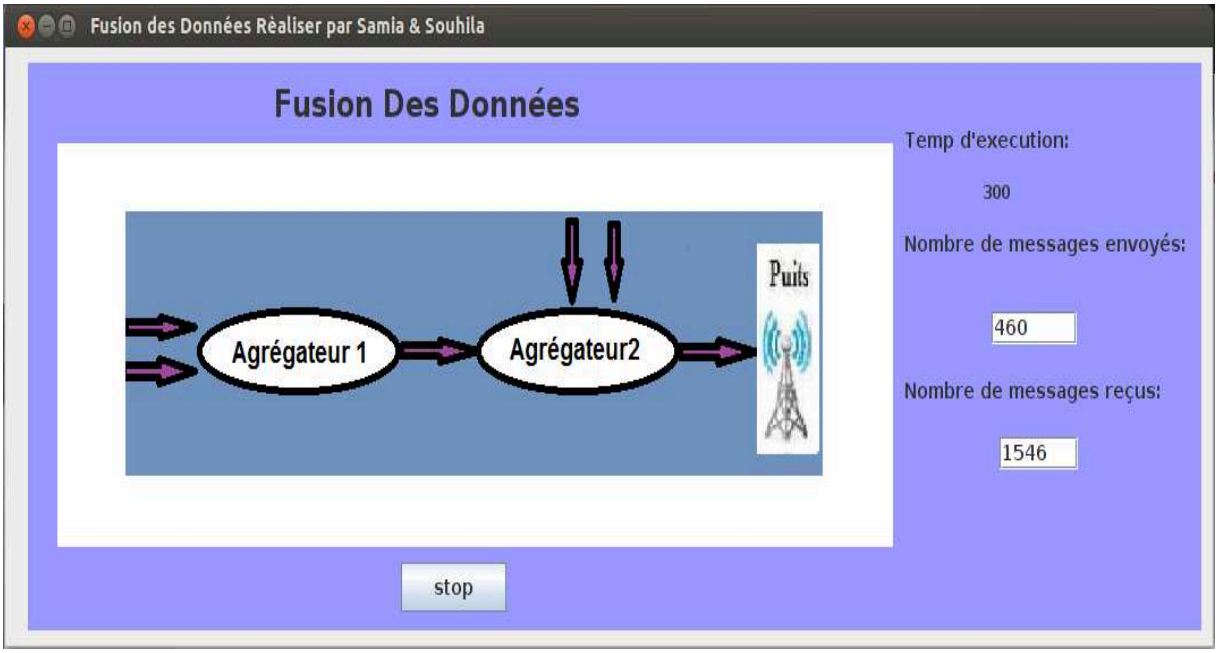

Figure 0III-15: Architecture clustérisée 2 après 5 minutes d'exécution

Tableau 0III-3 : Résultats de l'architecture clustérisée 2

Temps (Minutes)	*1*	*2*	*3*	*4*	*5*
Nbre de paquets envoyés	91	185	280	372	460
Nbre de paquets reçus	306	616	930	1238	1546

La première ligne du tableau III-3 représente le temps durant lequel nous avons supervisé la zone d'intérêt, la deuxième le nombre de paquets transmis par les nœuds membres des deux clusters, et la troisième le nombre de paquets transmis par les nœuds agrégateurs à la station de base.

Nous remarquons que le nombre de paquets reçus est supérieur au nombre de paquets envoyés. D'où il est souhaitable d'impliquer des nœuds agrégateurs pour éviter la redondance des informations et par suite la minimisation de l'énergie sera assurée. En outre, le schéma de routage joue un rôle très important en termes de réduction de nombre de messages envoyés ce qui a un impact positif sur la consommation de l'énergie.

d) Récapitulatif des résultats

Dans cette partie, nous illustrons la différence entre les trois architectures par un graphe en prenant les résultats présentés dans les tableaux III-1, III-2 et III-3.

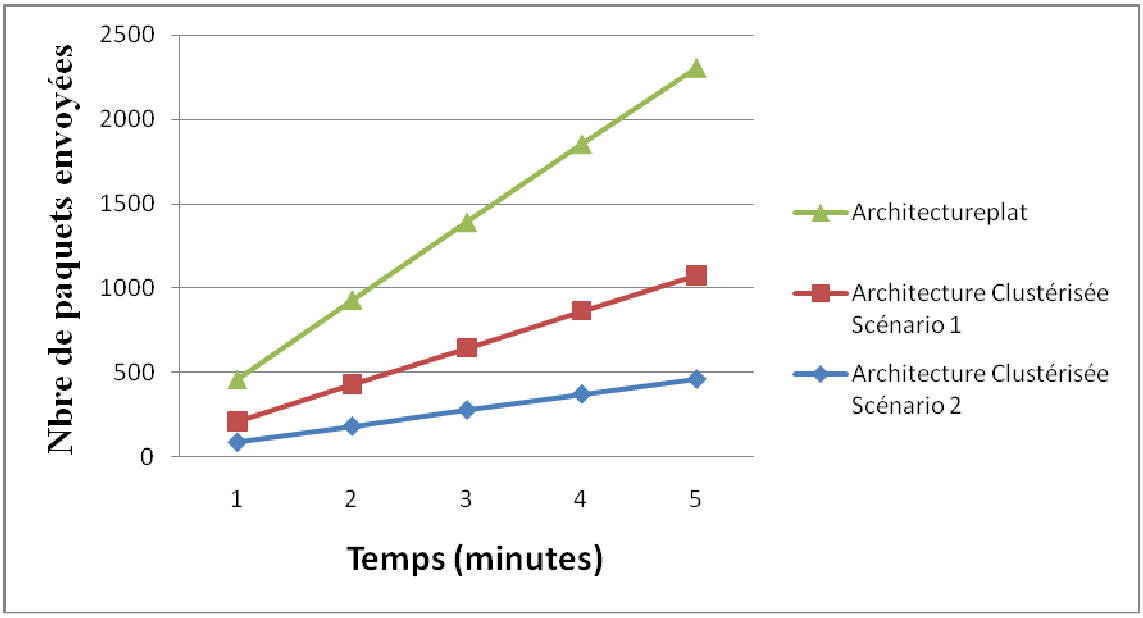

Figure III-16 : Nombre de messages envoyés en fonction du temps

La figure III-16 montre que le nombre de messages envoyés est réduit quand nous adoptons une architecture clustérisée comparativement au scénario dans lequel nous avons utilisé de l'architecture à plat. En outre, si nous faisons la comparaison entre les deux scénarios dans lesquels l'architecture clustérisée a été, nous remarquerons que le nombre de messages envoyés est réduit quand seulement un seul agrégateur fait la transmission à la station de base comparativement au scénario dans lequel les deux agrégateurs transmettront

directement les données agrégées à la station de base. Ces résultats symbolisent que la consommation d'énergie est minimisée dans le cas où nous utilisons une architecture clustérisée avec un bon acheminement des messages (les agrégateurs communiquent entre eux et un seul agrégateur fait la transmission à la station de base). Cette constatation est faite sur la base que la consommation d'énergie est calculée en fonction de la quantité d'informations échangées.

III.5 Conclusion

Dans ce chapitre, nous avons présenté la démarche à suivre pour de réaliser une application qui permet de mettre en valeur plusieurs schémas de routage dans les RCSF.

Le développement de cette application nécessite des outils logiciels bien particuliers tels qu'un système d'exploitation léger "TinyOs" et un langage orienté composant "NesC". En outre, pour montrer les bénéfices de chaque schéma de routage, nous avons évalué les performances de chacun d'eux sur une plateforme réelle de réseaux de capteurs. Les résultats obtenus ont montré que lorsqu'un nombre réduit de nœuds agrégateurs est utilisé les résultats étaient meilleurs.

Conclusion générale

Conclusion générale

Les réseaux de capteurs sans fil ont un large potentiel et constituent un sujet de recherche innovant ainsi qu'un outil convoité par plusieurs domaines.

C'est sans aucun doute, une technologie qui va nous accompagner pour les prochaines années et ainsi faire partie de notre vie quotidienne. Cependant, il y a encore beaucoup de problèmes qui doivent être abordés pour un fonctionnement efficace de ces réseaux dans des applications réelles. Parmi les problèmes fondamentaux et importants dans ces réseaux de capteurs nous citons la problématique de l'économie de l'énergie qui est une nécessité absolue à laquelle des solutions adéquates doivent être proposées.

Ce projet nous a permis d'acquérir des connaissances en programmation évènementielle. Il nous a aussi fait découvrir un nouveau langage de programmation, le NesC ainsi que la plateforme de programmation adéquate qui est TinyOs. En outre, dans ce projet on a pu bien comprendre le fonctionnement des réseaux de capteurs sans fil.

Le travail consigné dans ce mémoire a été le fruit d'une étude menée dans le contexte des réseaux de capteurs sans fil, ce qui nous a permis de découvrir les propriétés de ces derniers, de leurs contraintes et des domaines variés qui les utilisent. Nous nous sommes intéressées principalement au routage des données dans les réseaux de capteurs pour préserver l'énergie puisque le nombre de messages envoyés et reçus a un impact sur la consommation d'énergie.

Dans notre projet, nous nous sommes intéressées à la mise en place de trois schémas de routage sur une plateforme réelle de réseaux de capteurs dans le but de mettre en valeur les bénéfices de chaque schéma de routage sur la durée de vie du réseau. Au début, nous avons mis en place une architecture à plat, puis une architecture clustérisée. Dans cette dernière, nous avons considéré deux variantes : une variante similaire à LEACH et une autre basée sur l'approche CDS.

En perspectives, nous proposons de mettre en place des protocoles de routage sur des plateformes réelles de réseaux de capteurs.

Références Bibliographiques

Références Bibliographiques

[1] Heinzelman and al. “Energy-efficient communication protocol for wireless microsensor networks”, IEEE Proceedings of 33rd Annual Hawaii International Conference on System Sciences (HICSS '00), Vol. 2, pp.3005-3017, Maui, Hawaii, USA, January 2000.

[2] K.M. Alzoubi, P.J Wan, and O. Frieder. “New distributed algorithm for connected dominating set in wireless ad hoc networks”. In Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS’02), pp.3849-3855, Maui, Hawaii, January 2002.

[3] K.M. Alzoubi, P.J. Wan, and O. Frieder. “Distributed heuristics for connected dominating set in wireless ad hoc networks”. Journal Of Communication and Networks, vol.4, no.1, pp.22-29, March 2002.

[4] Vernon S. Somerset, “Intelligent and Biosensors”, Edited by Vernon S. Somerset, Intech, January 2010.

[5] David Martins, ”Sécurité dans les réseaux de capteurs sans fil Stéganographie et réseaux de confiance“, L'U.F.R. des Sciences et Techniques de l’université de Franche-Comté, 2010.

[6] Yacine Younes, ”Minimisation d’énergie dans un réseau de capteurs“, Mémoire de Master, Département d’Informatique, Université Mouloud Mammeri de Tizi-Ouzou, Septembre 2012.

[7] Tinyos. http:/www.tinyos.net/, 2010

[8] Dunkels, A., B. Grönvall, et T. Voigt. Contiki: a Lightweight and Flexible Operating System for Tiny Networked Sensors. In Proceedings of the First IEEE Workshop on Embedded Networked Sensors, pages 455-462, Tampa, Florida, USA, 2004.

[9] http://www.btnode.ethz.ch/Projects/SensorNetworkMuseum.

[10] Jun SHI, “Implémentation de mécanismes de sécurité efficaces pour les réseaux de capteurs”, Mémoire de Master, Université de Franche-Comté UFR Sciences et Techniques, Juin 2010.

[11] M. Badet, W. Bonneau. ”Mise en place d’une plateforme de test et d’expérimentation“, Projet tutoré (1ière Master Technologie de l’Internet), Mémoire de Master, Université Pau et des pays de l’Adour, 2006.

[12] M. LEHSAINI, “Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique”, Thèse de Doctorat, Université de Tlemcen et Université de Franche-Comté, 2009.

[13] http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

[14] Intanagonwiwat. C, Govindan. R, Estrin. D, ”Directed Diffusion: a scalable and robust communication paradigm for sensor networks“, ACM Press, 2000.

[15] Xue Yong, Gonzalez Andres, Aguilar Andres, Barroux Mickaël, “Agrégation de données dans les réseaux de capteurs”, Rapport de Projet SR04, Université de Technologie Compiègne, Automne 2010.

[16] http ://www.tinyos.net/tinyos-1.x/doc/nesc/ref.pdf.

[17] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han. Mantis OS : an embedded multithreaded operating system for wireless micro sensor platforms. Mob. Netw. Appl., 10(4) : 563-579, 2005.

[18] Bouzidi Zeyneb et Benameur Amina, ”Mise en place d’un réseau de capteurs sans fil pour l’irrigation intelligente“, Mémoire de Master, Université de Tlemcen, 2012.

[19] Séverine Sentilles, “Architecture logicielle pour capteurs sans-fil en réseau”, Mémoire de Master, Université de Pau et des Pays de l’Adour, Juin 2006.

[20] www.techno-science.net.

[21] C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In MobiSys '05 : Proceedings of the 3rd international conference on Mobile systems, applications, and services, pages 163_176, New York, NY, USA, 2005.

[22] ACM. Nut/OS. In http ://www.ethernut.de/en/software/index.html.

[23] M. Damou, L. Mounier, ”Simulation d’un réseau de capteurs avec TinyOS“, Rapport de recherche, Laboratoire VERIMAG, Grenoble, France.

Annexe

Annexe: Installation de TinyOs 2.1.2 sur Ubuntu

Etape1 :

- exécutez la commande ci-dessous:

  sudo gedit /etc/apt/sources.list

- coller le code ci-dessous dans le fichier ouvert (à la fin), puis enregistrer et fermer:

 # TinyOS Repository deb http://tinyos.stanford.edu/tinyos/dists/ubuntu lucid main

- mettre à jour et installer TinyOS en exécutant les commandes suivantes :

  sudo apt-get update

  sudo apt-get install tinyos-2.1.2

Etape 2 :

- exécutez les commandes ci-dessous:

  sudo chown your_user_name:your_user_name -R /opt/tinyos-2.1.2/

  sudo chown your_user_name -R /opt/tinyos-2.1.2

- Maintenant sur votre ligne de commande allez dans votre dossier installation de TinyOS

cd /opt/tinyos-2.1.2

- Créer un fichier 'tinyos.sh' avec le contenu cidessous:

```
#! /usr/bin/env bash
# www.ElectronicsPub.com
# TinyOS 2.1.2 Configuration Guide
# Here we setup the environment
# variables needed by the tinyos
# make system
echo "Setting up for TinyOS 2.1.2"
export TOSROOT=
export TOSDIR=
export MAKERULES=
TOSROOT="/opt/tinyos-2.1.2"
TOSDIR="$TOSROOT/tos"
CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java
MAKERULES="$TOSROOT/support/make/Makerules"
export TOSROOT
export TOSDIR
export CLASSPATH
export MAKERULES
```

- Ouvrir l'éditeur de texte, ouvrez ce fichier en utilisant la commande :

sudo gedit ~/.bashrc

- Maintenant, ajoutez le contenu suivant dans le fichier :

```
# Start TinyOS environment pathing
export TOSROOT=/opt/tinyos-2.1.2
export TOSDIR=$TOSROOT/tos
```

export CLASSPATH=$TOSROOT/support/sdk/java/tinyos.jar:.$CLASSPATH
export MAKERULES=$TOSROOT/support/make/Makerules
export PATH=/opt/msp430/bin:$PATH
source /opt/tinyos-2.1.2/tinyos.sh
End TinyOS pathing

- Maintenant, pour appliquer les modifications exécuter la commande ci-dessous:

source ~/.bashrc

Etape 3 :

- Exécutez les commandes ci-dessous

cd $TOSROOT/support/sdk/java

sudo tos-install-jni

make

make install

Etape 4 :

- Exécuter la commande suivante

sudo apt-get autoremove --purge msp430*

- Ajoutez les clés nécessaires à votre Ubuntu en utilisant les commandes ci-dessous

gpg --keyserver keyserver.ubuntu.com --recv-keys 34EC655A

gpg -a --export 34EC655A | sudo apt-key add –

- Ajouter les sources de paquets suivants dans votre liste de source # TinyOS MSP430

deb http://tinyprod.net/repos/debian squeeze main

deb http://tinyprod.net/repos/debian msp430-46 main

- Exécutez les commandes ci-dessous pour terminer l'installation du MSP430

sudo apt-get update

sudo apt-get install msp430-46 nesc tinyos-tools

- Ajout de PATH, pour cela exécuter la commande suivante :

sudo gedit ~/.bashrc

- Ajouter la ligne suivante

PATH=/opt/msp430-46/bin:$PATH

- On termine par appliquer les modifications, sur ligne de commande exécuter la commande suivante :

source ~/.bashrc