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

Université Saad DAHLEB -BLIDA-

Faculté des Sciences
Département d'Informatique

Mémoire de fin d'étude

En vue de l'obtention du
Diplôme de MASTER en INFORMATIQUE

Option : Ingénierie des Logiciels

THÉME

Implémentation d'un protocole de planification pour la collection de données agrégées dans un réseau de capteurs sans fil

Réalisé par : Rebhi Yacine Youcef et Mokdad Mohamed

Promoteur :
M. Bagaa Miloud

Encadreur :
Dr. Djennouri Djamel

Jury:
1. Mlle. Benblidia Nadjia (Presidente)
2. Mlle. Miloud Amal (Examinatrice)
3. Mme. Rezoug Nachida (Examinatrice)

-2012/2013-

Remerciements

Tout d'abord, je tiens à rendre grâce à Dieu tout puissant pour nous avoir donné le courage et la détermination nécessaires pour finaliser ce travail et le mener à terme.

Mon premier remerciement s'adresse au directeur de ce mémoire, Monsieur Bagaa Mouloud, attaché de recherche au CERIST, pour la qualité de son encadrement. Sa disponibilité et son attention du début à la fin de ce mémoire, ont été pour moi une aide précieuse et une source d'équilibre pour la réalisation de ce travail. Pour cela je renouvelle mes remerciements.

Je voudrai également remercier monsieur Djennouri Djamel qui nous as fait l'honneur d'être notre promoteur ainsi que pour son aide, ses conseils et son soutient durant ce projet

Je remercié, également le tous le personnel du CERIST pour leurs aide, leurs conseils ainsi que leurs soutiens tout au long de l'année.

Mes sincères remerciements et Madame Bessegui Hamida ainsi que Madame Naceur Djamila, qui n'ont cessé de m'encouragé et m'ont offert l'opportunité de faire ce que j'aimais

Un grand Merci à mes amis, à tous ceux qui m'ont soutenu, a toute la clique (vous vous reconnaitrez), cette année auras été inoubliable à vos côtés, et à tous ceux qui ont participé de près ou de loin à l'aboutissement de ce travail.

Je remercie les membres de jury pour nous avoir fait l'honneur de juger notre travail

Je termine par dédicacer ce mémoire à mon père et ma mère qui sans eux tout cela n'aurait pas été possible ...

Rebhi Yacine Youcef

Remerciements

Tout d'abord, nous tenons rendre grâce à Dieu tout puissant pour nous avoir donné le courage et la détermination nécessaire pour finaliser ce travail et le mener à terme.

Je remercie notre promoteur M. Bagaa pour son aide précieuse et ses conseils avisés.

Je tiens aussi à remercier tous le personnel du CERIST pour leur sympathie, leurs aides et leurs encouragements.

Nous remercions les membres de jury pour nous avoir fait l'honneur de juger notre travail

Enfin, un grand merci à ma famille et mes amis qui m'ont encouragé tout le temps dans ma vie."

ملخص

ان التطورات الحديثة في تكنولوجيا الاتصالات اللاسلكية والإلكترونيات الدقيقة ساعدت في تطوير أجهزة استشعار لاسلكية منخفضة التكلفة. كما ان هذه الأخيرة مزودة بوحدة حساب، وذاكرة وواجهة اتصالات لاسلكية.

ولأجل ذلك، تم إنشاء نوع جديد من شبكات مخصصة لتوفير حلول مثيرة للاهتمام اقتصاديا لبيئات معقدة وموزعة. وهي شبكات الاستشعار اللاسلكية. التي يتمثل دورها في جمع البيانات من بيئة ونشرها داخل الشبكة. وان هذا النوع من الشبكات لديه تشكيلة واسعة من التطبيقات في المجال المدني أو العسكري أو الطبي.

ورغم ذلك، ما زالت شبكات الاستشعار اللاسلكية تعاني من مجموعة من العقبات المادية التي تؤدي إلى صعوبة إدارتها. وفي الواقع، أجهزة الاستشعار محدودة من حيث موارد الطاقة، وتخزين الذاكرة، والقدرة الحسابية وعرض النطاق الترددي، الخ.

يعد العثور على طريقة لتوجيه المعلومات الموزعة التي تجمعها أجهزة الاستشعار إلى المحطة الأساسية باستخدام تقنيات تجميع البيانات مع تقليل وقت الانتظار أحد مجالات البحث في شبكات الاستشعار اللاسلكية. وفي إطار مشروع نهاية دراستنا (تنفيذ بروتوكول برمجة تجميع البيانات في شبكات الاستشعار اللاسلكية) سنقوم بدراسة ومقارنة مختلف الخوارزميات الموجودة التي تعالج مشكلة تخطيط التجميع.

لحل هذه المشكلة وتقليل الوقت قمنا ببرمجة البروتوكول (DSCA) الذي هو واحد من بروتوكولات تخطيط تجميع البيانات الموجودة التي تقلل من الوقت واستهلاك الطاقة. والهدف من هذا المشروع هو برمجة البروتوكول واختبار أدائه على شبكة من أجهزة الاستشعار التجريبية.

الكلمات الرئيسية

أجهزة الاستشعار اللاسلكية. شبكات الاستشعار اللاسلكية. بروتوكول التخطيط لجمع البيانات المجمعة.

Résumé

Les récentes avancées dans les domaines des technologies de communication sans-fil et microélectroniques ont permis le développement à faible coût des capteurs sans fil. Ces derniers sont munis d'une unité de calcul, d'une mémoire et d'une interface de communication sans fil. De ce fait, une nouvelle variante de réseaux ad-hoc s'est créée afin d'offrir des solutions économiquement intéressantes pour des environnements complexes et distribués. Ce sont les réseaux de capteurs sans fil « *RCSF* », ayant pour rôle de collecter des données d'un environnement et de les diffuser au sein du réseau. Ce type de réseaux a une très grande variété d'applications dans le domaine civil, médical ou encore militaire. Cependant, ces réseaux de capteurs sans fil se caractérisent par un ensemble de contraintes matérielles qui compliquent leur gestion. En effet, les micro-capteurs sont limités en termes de ressources énergétiques, mémoire de stockage, capacité de calcul, bande passante, etc.

Un des axes de recherche dans les *RCSFs* consiste à trouver une méthode qui permet d'acheminer les informations distribuées récoltées par les nœuds de capteurs vers la station de base en utilisant les techniques d'agrégations de données tout en minimisant le temps d'attente. Dans le cadre de notre projet de fin d'étude « Implémentation d'un protocole de planification pour la collection de données agrégées dans un réseau de capteurs sans fil », nous allons étudier et comparer les différents algorithmes existant traitant le problème de planification de l'agrégation. Afin de palier à ce problème et afin de minimiser les délais nous allons implémenter le protocole *DSCA*, qui est l'un des protocoles de planification de l'agrégation de données existants qui minimise le délai et la consommation d'énergie. L'objectif de ce projet est d'implémenter le protocole *DSCA* et tester ses performances sur un réseau de capteurs expérimental constitué de capteurs MicaZ.

Mots Clés

Capteurs sans fils, Réseaux de capteurs dans fil, planification de l'agrégation, *DSCA*, MicaZ.

Abstract

The recent advances in wireless communication technologies and micro-electronics allow the development of low-cost wireless sensor nodes. The latter equipped with a calculation unit, a memory and a wireless communication antenna. Therefore, a new variant of ad-hoc networks has been created to provide economically interesting solutions for remote monitoring and data processing in complex and distributed environments. The main objective of wireless sensor network (WSN) is the collect of data from the environment. Wireless sensor network has a wide range of applications, such as health-care, environment and military. Wireless sensor network is characterized by a set of physical constraints that complicate their management. In fact, the micro-sensors are limited in terms of energy consumption, memory storage, computing capacity, bandwidth, etc...

One of the research areas in WSN is how to gather data from nodes to the base station with a distributed manner. Data aggregation is an important mechanism to reduce energy consumption when gathering data. In the context of this final project study "Implementation of data aggregation scheduling protocol in WSN", we will review and compare the different existing solutions that aim to schedule data aggregation. To resolve data aggregation problem and minimize time latency, we implement *DSCA* protocol, which is one of the data aggregation scheduling protocols. The objective of this project is to produce a real implementation of *DSCA* protocol and thus test it in a test-bed which composed with MicaZ sensors.

Keywords

Micro-Sensors, Wireless Sensor Network, data aggregation scheduling, *DSCA*, MicaZ.

Sommaire

Chapitre -V- : Implémentation, simulation et résultats

Liste des Figures

Table des acronymes

ACS: Aggregation Converge-cast Scheduling.

APC: Arbre du plus court-chemain.

BSPT: Balanced Shortest Path tree.

CCA : Clear channel assesment.

CERIST: Centre de recherché sur l'information scientifique et technique.

CSMA Carrier Sense Multiple Access.

CSMA/CA CSMA with Collision Avoidance.

DADIDA: Delay-Aware Data aggregation in wireless sensor networks.

DAS: Destributed Date Aggregation.

DSCA: Destributed simultanious tree construction and data aggregation scheduling

DST: Dominating set tree.

GPS: Global Positionning System.

ISDA: Improved- SDA.

LLC: Logic link control.

MAC Media Access Control

NCA: Nearly constant approximation.

PDA: Poney-tree based aggregation.

QoS: Quality of Service.

RCFSs: Reseaux de capteurs sans fil.

SAS: Senquential Aggregation Scheduling.

SDA: Schortest Data Aggregation.

TDMA Time Division Multiple Access.

INTRODUCTION GÉNÉRALE

Afin de parfaire les connaissances de notre environnement et récolter un maximum de données scientifiques au plus près du terrain, il est possible de s'appuyer sur de grands nombres de capteurs très discrets, organisés et reliés en réseaux, composés de plusieurs dizaines a plusieurs centaines d'entités capables de prélever des données sur leur environnement. Ces réseaux de capteurs peuvent prélever des données de température, de taux d'humidité, de luminosité, de vitesse, d'accélération, d'onde sismique, etc. Cependant ces capteurs ou nœuds sont très limités en ressources énergétique, en mémoire et portée de leur carte radio. Pour palier a ce problèmes et atteindre le nœud central chargé de récolter et de traiter toutes les données recueillies, les capteurs doivent jouer entre eux le rôle de relais pour acheminer les données de bout en bout, c'est ce que l'on appelé une communication multi-saut. Ainsi constitué ces réseaux peuvent être déployé pour des applications très diverses et cela dans toutes les disciplines. L'élargissement de la gamme des types de capteurs disponibles (thermique, optique, vibrations,...) et l'évolution des supports de communication sans fil, ont élargi le champ d'application des réseaux de capteurs [1]. Ils s'insèrent notamment dans d'autres systèmes tels que le contrôle et l'automatisation des chaînes de montage. Ils permettent de collecter et de traiter des informations complexes provenant de l'environnement (météorologie, étude des courants, de l'acidification des océans, de la dispersion de polluants, etc. [2]

Un des axes de recherche dans les *RCSFs* consiste à trouver une méthode qui permet d'acheminer les informations distribuées récoltées par les nœuds de capteurs vers la *station de base* en utilisant les techniques d'agrégations de données. L'agrégation de données consiste à combiner les données récoltées par différentes sources dans le réseau avant qu'elles n'arrivent à la station de base, en éliminant les redondances, minimisant le nombre de paquets transmis afin d'optimiser la consommation d'énergie. Lorsque l'agrégation de ses données est utilisée, chaque nœud dans le réseau doit attendre tous ses prédécesseurs afin d'agréger leurs données [3]. Par conséquence, l'agrégation de données peut devenir une source de problèmes pour les applications avec contrainte de délai. Afin de résoudre ce problème, récemment plusieurs solutions ont été proposées pour la planification de l'agrégation de données. [4] [5]

Dans ce projet nous nous intéressons aux protocoles de planification de l'agrégation de données. Pour pallier à ce problème et afin de minimiser les délais de transmission des données, nous avons implémenté le protocole *DSCA*, qui est l'un des protocoles de planification de l'agrégation de données

existants qui minimise le délai et la consommation d'énergie. L'objectif de ce projet est d'implémenter le protocole *DSCA* et tester ses performances sur un réseau de capteurs expérimental constitué de capteur MicaZ.

Dans ce contexte, notre Projet de Fin d'étude vise à:

1. Etablir une étude bibliographique sur les différents protocoles de planification pour la collection de données agrégées, et ainsi établir les points faibles de ses protocoles en fiabilité et en optimisation temporelle.
2. Apprendre comment développer des applications pour les réseaux de capteurs sans fil en utilisant TinyOS [6].
3. Implémenter le protocole *DSCA* en utilisant TinyOS et les capteurs MICAz.
4. Tester le protocole *DSCA* en utilisant un Test-Bed qui contient 11 capteurs MICAz

Ce document est organisé comme suit :

Le premier lieu nous allons montrer l'importance du domaine de capteur sans fils dans la recherche et la prévention, ensuite nous donner des exemples des différents champs d'application de réseaux de capteurs.

Le second chapitre s'inscrit dans le contexte d'une étude globale sur les réseaux de capteurs sans fil. Pour cela, nous présenterons leurs éléments de base, l'architecture de communication, les principales caractéristiques et contraintes qui influencent la conception de ce genre de réseaux.

Le troisième chapitre sera une étude bibliographique sur les différent protocoles de planification de l'agrégation, nous montreront aussi pour chacun des protocoles ses points faibles.

Le quatrième chapitre expliquera notre choix pour du protocole *DSCA* ainsi que son fonctionnement.

Le cinquième chapitre l'implémentation du protocole *DSCA* ainsi que les tests réel de notre protocole *DSCA*.

En fin, la conclusion de ce mémoire synthétisera nos principales contributions et donnera quelques perspectives á notre travail.

CHAPITRE -I-

Généralité sur les réseaux de capteurs sans fil

1. Introduction

Le domaine des communications et en particulier les réseaux de capteurs sans fil RCSF a connu une évolution spectaculaire ces dernières années. Un réseau de capteurs sans fil est un cas particulier des réseaux sans fil sans infrastructure (réseaux ad hoc). En effet, ceux-ci sont constitués d'un ensemble de petits appareils, ou capteurs, possédant des ressources particulièrement limitées mais qui leur permettent d'acquérir des données sur leur environnement immédiat, de les traiter et de les communiquer. En effet, ces capteurs sont moins coûteux, ce qui permet de les utiliser dans de plus en plus de domaines [1]. Un réseau de capteurs sans fil présente des intérêts considérables pour le secteur industriel, mais aussi pour les organisations civiles où la surveillance et la reconnaissance de phénomènes physiques sont une priorité. En effet, un réseau de capteurs peut être mis en place dans le but de surveiller une zone géographique plus ou moins étendue pour détecter l'apparition de phénomènes ou mesurer une grandeur physique (température, pression, vitesse...).

Afin de montrer l'importance de ce genre de réseaux, la présentation des différents champs d'applications dans un premier lieu est nécessaire, Ensuite les caractéristiques d'un capteur sans fil seront détaillées. Enfin, nous aborderons plus en détails l'architecture d'un réseau de capteurs sans fil.

2. Les domaines d'application

2.1. Domaine militaire

Le domaine militaire a été un moteur initial pour le développement des réseaux de capteurs. Le déploiement rapide, le coût réduit, l'auto-organisation et la tolérance aux pannes des réseaux de capteurs sont des caractéristiques qui rendent ce type de réseaux un outil appréciable dans un tel domaine. Un réseau de capteurs déployé sur un secteur stratégique ou difficile d'accès permet :

2.1.1 Le contrôle et la gestion des forces : surveiller le statut et l'emplacement des troupes et des armements afin d'améliorer le contrôle, la communication et le

commandement.

2.1.2 La surveillance et le contrôle des champs de bataille [7]: un réseau de capteurs est déployé à partir d'un avion dans un champ d'intérêts, ensuite les capteurs vont se réorganiser afin d'exécuter les tâches prévues (analyser le terrain, détecter et poursuivre des objets ennemis, …etc.).

2.1.3 La protection : les objets sensibles (les stations nucléaires, les ponts, les canaux de pétrole et du gaz, les stations de communications, les réserves d'armes, et les quartiers militaires) peuvent être protégés par un réseau de capteurs intelligent capable de distinguer les différentes classes d'intrus.

2.2. Domaine de la sécurité

Les structures d'avions, navires, automobiles, métros, etc. pourraient être suivies en temps réel par des réseaux de capteurs. De même que les réseaux de circulation [8] ou de distribution de l'énergie. Les altérations de structure d'un bâtiment, d'une route, d'un quai, d'une voie ferrée, d'un pont ou d'un barrage hydroélectrique (suite à un séisme ou au vieillissement) pourraient être détectées par des capteurs préalablement intégrés dans les murs ou dans le béton, sans alimentation électrique ni connexions filaires. Certains capteurs ne s'activant que périodiquement peuvent fonctionner durant des années, voire des décennies. Un réseau de capteurs de mouvements peut constituer un système d'alarme distribué qui servira à détecter les intrusions sur un large secteur. Déconnecter le système ne serait plus aussi simple, puisqu'il n'existe pas de point critique. La surveillance de routes ou voies ferrées pour prévenir des accidents avec des animaux ou des êtres humains ou entre plusieurs véhicules est une des applications envisagées des réseaux de capteurs.

Pour cela une optimisation en temps est plus que vitale pour assurer cette tâche et donc les délais de transmission de ses informations à la station de traitement doivent être optimaux.

2.3. Domaine environnemental

Des thermo-capteurs peuvent être dispersés à partir d'avions, ballons, navires et signaler d'éventuels problèmes environnementaux dans le champ de captage (incendie [9], pollution, épidémies, aléa météorologique...) permettant d'améliorer la connaissance de

l'environnement et l'efficacité des moyens de lutte. Des capteurs pourraient être semés avec les graines par les agriculteurs afin de détecter le stress hydrique des plantes ou le taux de nutriment de l'eau du sol, pour optimiser les apports d'eau et de nutriments ainsi que le drainage ou l'irrigation [10]. Sur les sites industriels, les centrales nucléaires ou sur les pétroliers, des capteurs peuvent être déployés en réseau pour détecter des fuites de produits toxiques [11] (gaz, produits chimiques, éléments radioactifs, pétrole, etc.) et alerter les utilisateurs et secours plus rapidement, et permettre une intervention efficace. Une grande quantité de capteurs pourrait être déployée en forêt ou dans certaines aires protégées pour recueillir des informations sur l'état des habitats naturels et sur les comportements de la faune, de la flore et de la fange (déplacements, activité, état de santé… etc.). Il devient ainsi possible "d'observer la biodiversité", sans déranger, des espèces vulnérables ou difficiles à étudier dans leur environnement naturel, et proposer des solutions plus efficaces pour la conservation de la faune.

Dans un contexte mondial où le réchauffement de la planète devient une préoccupation grandissante, l'utilisation des réseaux de capteurs sans fil pour optimiser la consommation des ressources énergétiques peut avoir une conséquence environnementale positive. Un exemple de ce type d'applications est l'intégration de plusieurs micro-capteurs dans le système de climatisation et de chauffage des immeubles. Ainsi, la climatisation ou le chauffage ne sont déclenchés qu'aux endroits occupés par des personnes présentes et seulement si c'est nécessaire. Le système distribué peut aussi maintenir une température homogène dans les pièces. Utilisée à grande échelle, une telle application permettrait de réduire la demande mondiale en énergie. Dans ce cas d'utilisation les capteurs sont disséminés sur de grands espaces et par conséquent la progression des informations doit être planifié et dynamique pour pallier aux problèmes de communication et ainsi assurer l'information en temps réel.

L'environnement est aussi le champ qui apporte le grand intérêt au réseau de capteurs. Plusieurs applications peuvent montrer l'utilité de ce genre de réseaux, parmi lesquelles on cite:

- **La précision d'agriculture :** les informations collectées par un réseau de capteurs, dans un terrain[12], sur le climat et l'état de la terre, permettent un contrôle précis de

l'utilisation de l'eau et des engrais.

- **L'évolution des plantes:** étudier l'évolution des différentes espèces de plantes dans les grandes forêts, ou dans les profondeurs sous-marines, où la présence de l'homme est quasi impossible.

- **Exploration planétaire:** exploration et surveillance dans les environnements hostiles, par exemple les volcans, les régions toxiques, … etc.

- **La surveillance géophysique:** les activités sismiques peuvent être détectées et étudiées á l'aide d'un réseau de capteurs implanté dans les régions sensibles.

2.4. Domaine médical et vétérinaire

La surveillance des fonctions vitales d'un organisme vivant pourrait à l'avenir être facilitée par des micro-capteurs avalés ou implantés sous la peau. Des gélules multi-capteurs ou des micro-caméras pouvant être avalées existent déjà, pouvant sans recours à la chirurgie, transmettre des images de l'intérieur d'un corps humain (avec une autonomie de 24 heures). Une récente étude présente des capteurs fonctionnant dans le corps humain, qui pourraient traiter certaines maladies. Un projet est de créer une rétine artificielle composée de 100 micro-capteurs pour corriger la vue. On évoquera d'autres applications biomédicales ambitieuses, tel que : la surveillance de la glycémie, la surveillance des organes vitaux ou la détection précoce de cancers. Des réseaux de capteurs permettraient théoriquement une surveillance permanente des patients et une possibilité de collecter des informations physiologiques de meilleure qualité, facilitant ainsi le diagnostic de certaines maladies [13].

Dans ce domaine également le gain de temps est crucial pour donner un sens aux informations et pouvoir ainsi agir avant qu'il ne soit trop tard.

2.5. Domaine du traçage et de la localisation

Suite à une avalanche il est nécessaire de localiser les victimes enterrées sous la neige en équipant les personnes susceptibles de se trouver dans des zones à risque par des

capteurs. Ainsi, les équipes de sauvetage peuvent localiser plus facilement les victimes. Contrairement aux solutions de traçabilité et de localisation basées sur le système de *GPS* (Global Positionning System), les réseaux de capteurs peuvent être très utiles dans des endroits confinés comme les mines par exemple.

L'impact du temps dans un tel domaine est plus qu'évident, par conséquent le gain de temps dans ce cas de figure et critique.

2.6. Domaine Industriel

Si nous considérons maintenant les applications qualifiées "d'industrielles" pour désigner des activités touchant à la production de produits manufacturés ou des fournisseurs d'énergie, dans ce domaine, les réseaux de capteurs sans fil offrent une grande flexibilité d'emploi puisqu'ils permettent de s'affranchir des contraintes liées aux câblages. Il est alors possible de satisfaire des contraintes de poids (d'un équipement dans un navire ou dans un avion), de mobilité (d'un robot dans une usine), de facilité de déploiement, d'isolation galvanique... Parmi ces applications nous pouvons citer comme exemples : la surveillance des ouvrages de génie civil pour établir un diagnostic à distance d'une évolution de leur état sans avoir à se rendre sur place pour faire une observation in situ , la surveillance de l'état de santé d'un ouvrier ou du risque de le voir exposé à des conditions de travail dangereuses (exposition à la radioactivité), l'observation d'un site susceptible de subir les effets d'une pollution et la construction en temps réel d'une cartographie de sa contamination grâce à des capteurs, disséminés sur le site, capables de relever les niveaux de pollution et de relayer l'information vers les services spécialisés.

2.7. Maison, bureau, et vie civile

Les maisons et les bureaux de travail peuvent être équipés de dizaines de capteurs qui surveillent la température et l'humidité pour contrôler la climatisation et le chauffage. Les PCs peuvent être équipés de claviers et souris sans fil et peuvent être relié entre eux en réseau ad hoc pour faire des travaux collaboratifs et collecter des informations vers une station de calculs pour accomplir un traitement centralisé.

Des réseaux de capteurs peuvent être déployés dans les différentes structures de la ville: ponts, bâtiments, réservoirs d'eau et d'énergie (électricité, carburant, gaz), afin de les

surveiller et détecter le moindre danger le plus tôt possible.

Un réseau de capteurs de mouvement peut constituer un système d'alarme distribué qui servira á détecté les intrusions sur un large secteur. Déconnecter le système ne serait plus aussi simple, puisque il n'existe pas de point critique.

Utiliser des réseaux de capteurs dans les routes et les rails permet une meilleure planification urbaine.

3. Le capteur sans fil

Le capteur sans fil est l'élément de base qui nous permet la réalisation des applications cité ci-dessus. Par conséquent une présentation d'un nœud de capteurs est indispensable. Un capteur, habituellement appelé *Mote*, est un dispositif autonome de taille extrêmement réduite capable d'effectuer des mesures simples sur son environnement immédiat L'utilisation de ces capteurs n'a rien d'une nouveauté. Ceux-ci sont présents depuis longtemps dans plusieurs domaines comme l'industrie, l'aéronautique ou l'automobile. Ces capteurs étaient en général reliés à une base de traitement grâce à des liaisons filaires. Ce qui est novateur c'est la possibilité pour ces dispositifs de communiquer, grâce à des liens sans fil (onde radio Wifi ou Zig Bee [14] par exemple), avec d'autre distants de quelques mètres. Il est donc possible de constituer un réseau de capteurs qui collaborent sur une étendue assez vaste ce qui permettra un déploiement facile et rapide dans un environnement pouvant même être inaccessible pour l'être humain. Pour d'autres dispositifs, l'innovation consiste, en plus des liens sans fil, en une capacité de traitement (processeur) et une mémoire embarquée. Il existe différents capteurs avec différent caractéristique (TelosB, MicaZ, Mica2, etc…) Ces appareils sans fil possèdent trois (03) fonctions :

1- Capter des données (Vibration, lumière, humidité, pression..) en détectant des phénomènes dans un environnement proche.

2- Traiter les données à l'aide des informations collectées.

3- Communiquer les données obtenues, via les ondes radio, à une autre entité (capteurs, nœud, station de base, unité de traitement..) à courte portée.

Les capteurs respectent globalement la même architecture basée sur deux (02) Structures : matérielle et logicielle (voir *Figure 1.1*).

3.1. La structure matérielle (hardware)

Elle est basée un noyau central autour duquel s'articulent les différentes interfaces d'entrée/sortie, de communication et d'alimentation. De ce fait, un nœud capteur sans fil contient quatre (04) unité de base : l'unité de captage, l'unité de traitement, l'unité de communication et enfin l'unité de puissance ou batterie. Un capteur peut également disposer de diverses unités optionnelles [15].

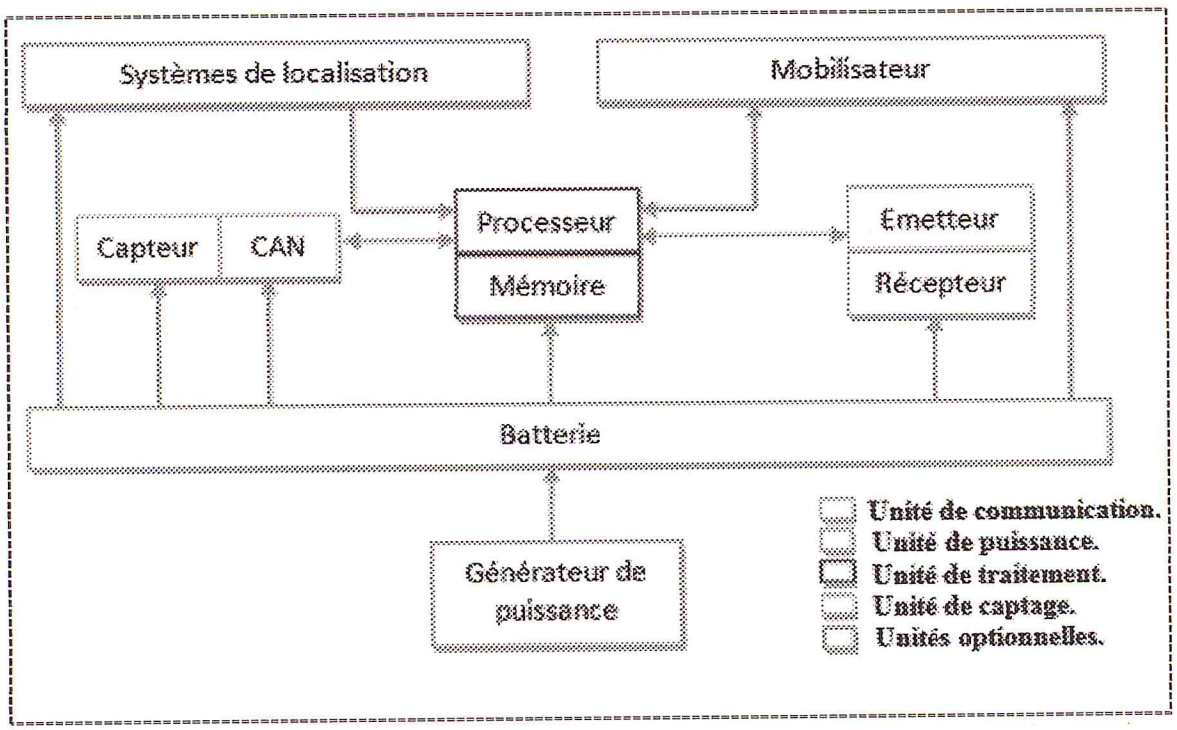

Figure 1.1 : « Architecture générale d'un nœud capteur »

3.1.1. L'unité de captage

Elle est généralement composées de deux (02) sous-unités : le capteur lui-même qui permet de capturer le phénomène observé et un convertisseur analogique/numérique (CAN) qui convertit le signal analogique en un signal numérique. Il sera ensuite, fourni à l'unité de traitement [15].

3.1.2. L'unité de traitement

L'unité de traitement comprend un processeur associé généralement à une petite unité de stockage et fonctionne à l'aide d'un système d'exploitation spécialement conçu pour les capteurs (tel que TinyOS) [6]. Cette unité est chargée d'exécuter les protocoles de communication qui permettent de faire collaborer un nœud avec d'autres nœuds du réseau. Elle peut aussi analyser les données captées pour alléger la tâche de la station de base.

3.1.3. L'unité de communication

Elle est chargée d'effectuer toutes les émissions et les réceptions de données. La radio est le moyen utilisé par la communication. Celle-ci a plusieurs modes de fonctionnement (émission, réception, veille). La consommation d'énergie diffère d'un mode à un autre et la gestion de ces modes joue un rôle important dans la conservation d'énergie.

3.1.4. L'unité d'énergie

Un capteur doit disposer de sa propre source d'énergie qui alimente l'ensemble des unités en question. Cette unité se présente généralement sous la forme d'une batterie standard à basse tension. A cette dernière est incorporé un dispositif qui contrôle et distribue l'énergie sur les différents composants et réduit les dépenses en énergie, par exemple, en mettant en veille les composants inactifs. Ce dispositif peut également gérer des systèmes de rechargement d'énergie à partir de l'environnement tels que les cellules solaires afin d'étendre la durée de vie du réseau.

3.1.5. Les unités optionnelles

En fonction des applications pour lesquelles ils sont conçus, les capteurs sans fil pourraient également avoir d'autre modules, comme une unité de localisation (par exemple un système de localisation global : GPS) [16]. Certaines applications pourraient aussi avoir besoin de capteurs équipés d'un mobilisateur afin qu'ils puissent se déplacer. Enfin, s'il est nécessaire qu'un nœud soit maintenu en activité pendant une très longue période de temps, un générateur de puissance serait utile, dans ce cas-là, afin de tenir le nœud alimenté électriquement sans avoir à changer ses batteries.

3.2. La structure logicielle (software)

La structure logicielle d'un capteur se divise en deux (02) sous-systèmes de base :

3.2.1. Système d'exploitation (Operating System) [17]

Il existe plusieurs systèmes d'exploitation pour les capteurs tels que : TinyOS [6], MagnetOS, OSPM, EYE OS, SenOS, RetOS [18]. Le système d'exploitation (SE) est utilisé par toutes les unités du capteur afin d'accomplir leurs différentes fonctions. Nous nous intéressons plus particulièrement au système TinyOS.

TinyOS

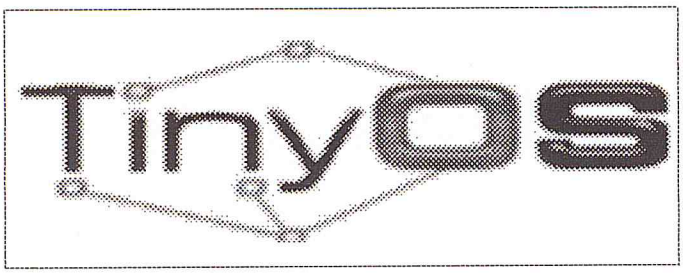

Figure 1.2 : « TinyOS »

TinyOS est un système d'exploitation intégré, modulaire, destiné aux réseaux de capteurs. Cette plate-forme logicielle open-source est composée d'une série d'outils développés par l'Université de Berkeley. TinyOS est le plus répandu des systèmes d'exploitation pour les réseaux de capteurs sans fil. Il est utilisé dans les plus grands projets de recherche. Il respecte une architecture basée sur une association de composants,

réduisant la taille du code nécessaire à sa mise en place afin de respecter les contraintes de mémoires qu'observent les réseaux de capteurs [19].

La bibliothèque de composants de TinyOS est particulièrement complète puisqu'on y trouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données.

En s'appuyant sur un fonctionnement événementiel, TinyOS propose à l'utilisateur une gestion très précise du fonctionnement du capteur et permet de mieux s'adapte à la nature aléatoire de la communication.

Le système TinyOS, ses librairies et ses applications sont écrits en NesC [20]. Ce dernier est un nouveau langage pour le développement d'applications orientées composants. Le langage NesC est principalement dédié aux systèmes embarqués comme les réseaux de capteurs. NesC a une syntaxe proche du langage C mais supporte le modèle concurrent de TinyOS ainsi que des mécanismes pour la structuration, le nommage et l'assemblage de composants logiciels en des systèmes réseaux embarqués fiables. L'objectif principal est de permettre aux concepteurs d'applications de construire des composants qui peuvent être composés rapidement en des systèmes complets, concurrents, tout en permettant une vérification profonde à la compilation.

3.2.2. Les pilotes (Sensor Drivers)

Il y a des modules logiciels qui gèrent les fonctions de transmission de base dans les capteurs ainsi que des pilotes qui gèrent les applications logicielles de bas niveau pour accomplir les fonctionnalités du nœud capteur.

4. le réseau de capteurs sans fil

4.1. Les caractéristiques d'un réseau de capteurs

Les réseaux de capteurs sans fil ont plusieurs caractéristiques communes avec les réseaux ad-hoc comme la nature du réseau. Cependant, plusieurs différences existent entre les deux types de réseaux comme le déploiement, les contraintes matérielles....etc. Parmi les caractéristiques [15] des réseaux de capteurs on trouve:

Densité de déploiement

La nature de l'application joue un rôle important dans le déploiement des réseaux de capteurs. Une des motivations de cette caractéristique est le faible coût des capteurs (relativement aux autres unités).

Energie limitée

Dans un *RCSF* l'alimentation de chaque nœud est assurée par une source d'énergie limitée et généralement irremplaçable à cause de l'environnement hostile où il est déployé. De ce fait, la durée de vie d'un *RCSF* dépend fortement de la conservation d'énergie au niveau de chaque nœud.

Communication

Les nœuds communiquent entre eux via des liaisons radiofréquences au sein du réseau, et fonctionnent avec le mode de communication par diffusion.

Absence d'adresse

Les nœuds dans les réseaux sans fil classiques sont identifiés par des adresses IP. Cependant, cette notion n'existe pas dans les *RCSFs* car les nœuds n'ont pas d'identifiants ou d'adresse IP.

Topologie

La topologie des réseaux de capteurs change d'une manière dynamique. Le nombre de nœuds capteurs déployés est très considérable (des centaines ou des milliers) selon le type d'application. Ces nœuds collaborent ensemble afin d'atteindre un objectif commun.

Interférences

Les liens radio ne sont pas isolés, deux transmissions simultanées sur une même fréquence, ou utilisant des fréquences proches, peuvent interférer et causer aussi la perte des messages.

4.2. La pile protocolaire des communications dans les réseaux de capteurs

La pile protocolaire utilisée par la station de base ainsi que tous les autres capteurs du réseau est illustrée par la *figure 1.3* cette pile prend en charge le problème de consommation d'énergie, intègre le traitement des données transmises dans les protocoles de routage, et facilite le travail coopératif entre les capteurs [3].

Elle est composée de la couche application, transport, réseau, liaison de données, physique, ainsi que de trois niveaux qui sont : le niveau de gestion d'énergie, de gestion de tache et le niveau de gestion de mobilité.

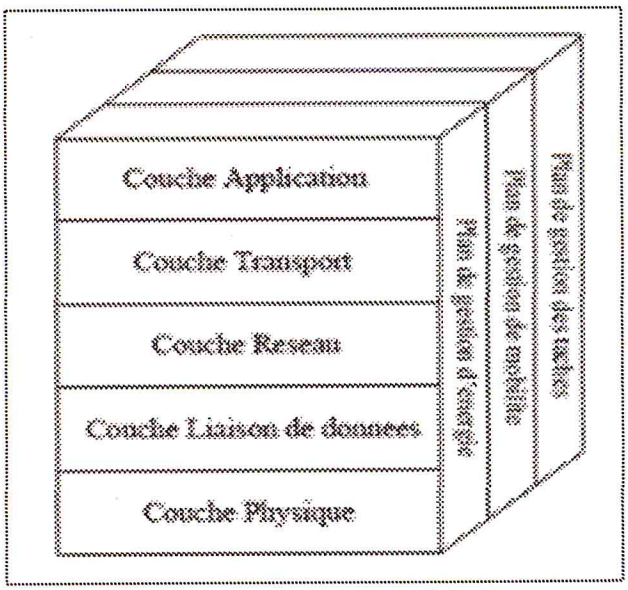

Figure 1.3 : « la pile protocolaire dans les réseaux de capteurs »

Suivant la fonctionnalité des capteurs, différentes applications peuvent être utilisées et bâties sur la couche application. La couche transport, quant à elle, sert à maintenir le flux de données en cas de nécessité dans les applications utilisées, particulièrement lors d'une connexion avec Internet, tandis que la couche réseau s'occupe du routage des données fournies par la couche transport.

Comme l'environnement des réseaux de capteurs est bruyant et les nœuds peuvent être mobiles, la couche MAC doit garantir une faible consommation d'énergie et un taux de collision minimum entre les données diffusées par les nœuds voisins. Enfin, la couche physique doit assurer des techniques d'émission, réception et modulation de données simples mais robustes.

Les niveaux de gestion d'énergie, de mobilité et de tâches sont responsables du contrôle de l'énergie consommée, des mouvements des nœuds et de la distribution des tâches à travers toute la pile protocolaire, ces niveaux permettent aux capteurs de coordonner leurs tâches et minimiser la consommation d'énergie.

4.2.1. Les couches de la pile protocolaire

Les cinq (05) couches de la pile protocolaire sont les suivantes :

1. **La couche Physique :** S'occupe de la spécification du câblage, des fréquences porteuses …etc. Elle décrit les techniques nécessaires pour l'émission, la réception et de la modulation de données entre deux capteurs.

2. **La couche liaison de données :** Spécifie comment les données sont expédiées entre deux nœuds dans une distance d'un saut. Elle est responsable du multiplexage des données, du contrôle d'erreurs, de l'accès au media…etc. Elle assure la liaison point à point et multi point dans un réseau de communication. La couche liaison de données est composée de deux sous couches :

- **La sous-couche LLC** : assure le Contrôle des Liens Logiques.

- **La sous-couche MAC :** sa fonctionnalité est d'établir les liens de communication entre les nœuds, et de fournir une fiabilité et un partage équitable des canaux de communication entre les nœuds du réseau.

3. **La couche réseau** : Le but de cette couche est de trouver une route et une transmission pour les données captées vers la station de base, en optimisant la consommation d'énergie.

4. **La couche transport :** Cette couche est chargée du transport des données, de leur découpage en paquets, du contrôle de flux, de la conservation de l'ordre des paquets et de la gestion des éventuelles erreurs de transmission.

5. **La couche application :** Cette couche est conçue selon l'information à capter par le capteur, elle assure l'interface avec les applications. Il s'agit donc du niveau le plus proche des utilisateurs géré directement par les logiciels.

Ce routage diffère de celui des réseaux de transmission ad hoc sans fil par les caractéristiques suivantes :

- Il n'est pas possible d'établir un système d'adressage global pour le grand nombre de capteur.
- Les applications des réseaux de capteurs exigent l'écoulement des données mesurées de sources multiples à une station de base particulière.
- Les différents capteurs peuvent produire les mêmes données à proximité d'un phénomène (redondance).

4.2.2. Les niveaux de gestion dans les réseaux de capteurs sans fil

4.2.2.1. Le niveau de gestion d'énergie

Les fonctions intégrées à ce niveau consistent à gérer l'énergie consommée par les capteurs, dès lors, un capteur peut par exemple éteindre son interface de réception dès qu'il reçoit un message d'un nœud voisin afin d'éviter la réception de messages dupliqués. De plus, quand un nœud possède un niveau d'énergie faible, il peut diffuser un message aux autres capteurs pour ne pas participer aux tâches de routage, et conserver l'énergie restante aux fonctionnalités de captage.

CHAPITRE -II-

Stratégies d'optimisation dans les réseaux de capteurs sans fil

1. Introduction :

Un réseau de capteurs sans fil consiste en un ensemble de nœud capteurs déployées à grande échelle dans un champ de captage pour détecter, recueillir et transmettre les données concernant un phénomène observé, vers la station de base via les liens sans fil.

Cependant suivant le nombre de nœuds dans le réseau et l'étendu du champ de captage, certains nœuds ne pourront pas transmettre directement leurs messages au nœud collecteur (ou station de base). Ainsi la collaboration entre les nœuds pour garantir cette transmission est une exigence. De cette manière les messages sont propagés par les nœuds intermédiaires en établissant les chemins multi-saut entre la source lointaine et la station de base.

Dans ce chapitre nous parleront des différents modèles de communication dans les *RCSF*s avant d'arriver aux modes d'acheminement des données dans ces derniers. Nous aborderont par la suite l'optimisation de la consommation d'énergie ainsi que les stratégies employées afin résoudre ses problèmes.

2. Modèle de communications dans les réseaux de capteurs sans fil:

Selon le mode de communication des données, plusieurs modèles de transmission peuvent être identifiés [3]. Nous en distinguons essentiellement trois:

1. Modèle basé *événement* (***Event-Driven***).
2. Modèle *périodique.*
3. Modèle *hybride.*

2.1. Modèle basé événement (Event-Driven) :

Dans ce type d'applications, le transfert des données vers la station de base est déclenché lorsqu'un événement particulier est détecté. La plupart des applications ***Event-Driven*** sont des applications intolérantes aux retards, interactives et critiques. L'inconvénient majeur de ce modèle est la redondance des données. En effet, les nœuds

détectent le même événement et envoient une information identique à la station de base. On peut citer l'exemple de surveillance des feux dans les forêts où un capteur envoi des alarmes à la station de base dès que la température dépasse un certain seuil.

2.2. Modèle *périodique* :

Dans le modèle périodique, les capteurs collectent des informations dans des intervalles de temps réguliers. Ensuite, ils envoient les données à la station de base de manière périodique.

2.3. Modèle *hybride* :

Dans ce modèle une application utilise l'un ou plusieurs modèles de transmission, décrit ci-dessus en même temps.
De plus en plus d'applications utilisent ce modèle de nos jours, i.e. L'application de télésurveillance médicale peut utiliser tous les modèles de transmission à la fois. Les capteurs envoient d'une façon périodique l'état des fonctions vitales du patient. Si entre temps un problème est détecté, un événement est transmis au professionnel de santé. Et ce dernier peut envoyer une requête pour vérifier l'état du patient à un instant donné.

3. L'acheminement de données dans les *RCSFs*:

Comme nous l'avons évoqué dans les chapitres précédents l'une des contraintes majeures des *RCSF* est l'énergie. Les capteurs sont dotés de piles généralement déployés dans les zones où il est quasiment impossible le remplacer ou de les recharger. La durée de vie d'un *RCSF* est donc étroitement liée à la façon de consommer cette énergie.

Le module le plus consommateur en énergie est le module radio (la transmission d'un bit consomme l'énergie nécessaire à l'exécution de 1000 instructions dans un capteur) Il est donc nécessaire de minimiser le nombre de bits à transmettre dans un RCSF [21]. Pour ce faire, les protocoles d'agrégation comprennent des algorithmes pour éliminer les données redondantes réduisant la quantité de données à transmettre sur le réseau et consommation moins d'énergie.

3.1. L'acheminement de données dans un Cluster:

Les nœuds capteurs détectent puis transmettent les données à la station de base. Ceci implique que le routage ou la coordination entre les nœuds n'est pas exigé. Donc, ce paradigme utilise un modèle de communication centralisé constitué d'un seul saut point à point. Ainsi, les capteurs récupèrent des données et les transmettent directement et immédiatement après leurs collectes ou leurs traitements dans un certain intervalle de temps. Le problème de ce paradigme est que les nœuds capteurs sont hors communication quand ils sont loin de la zone de captage de la station de base parce que le champ de captage ne peut pas être très grand.

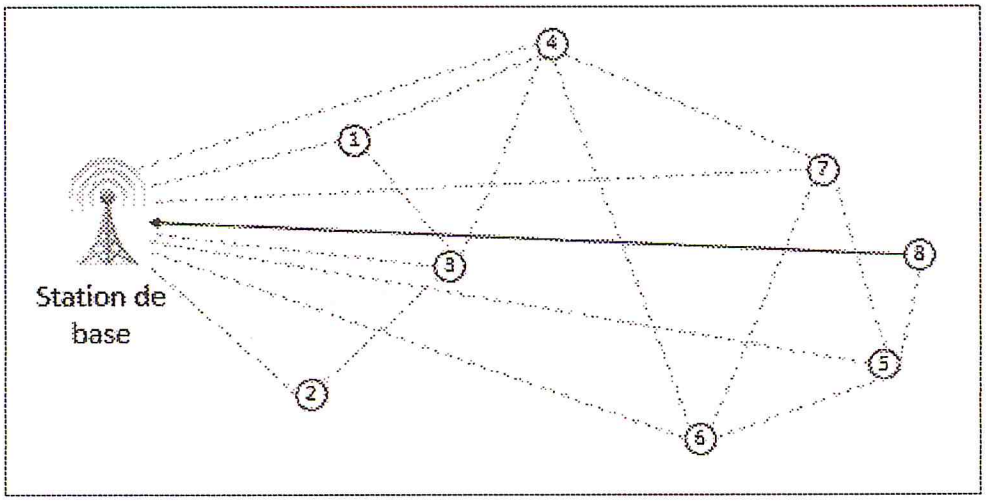

Figure 2.1 : « Paradigme à un seul saut à la station de base »

3.2. L'acheminement de données dans un arbre:

Ce paradigme permet aux nœuds capteurs éloignés de la station de base de transmettre leurs données aux nœuds voisins qui les expédient à leurs tours vers la station de base. Le processus d'expédition peut impliquer plusieurs nœuds de capteurs sur le chemin entre le nœud source et le point de collecte. Donc, ce paradigme utilise le modèle de communication centralisé, à multi sauts, indépendamment de la longueur du chemin jusqu'à ce que les données atteignent la station de base. La coordination et la

niveau MAC où se déroule le contrôle d'accès au support sans fil. Certains de ces phénomènes sont les causes majeures de la perte d'énergie et ont été recensés dans [24, 25, 26] :

1. **Les collisions :** elles sont la première source de perte d'énergie. Quand deux trames sont émises en même temps et se heurtent, elles deviennent inexploitables et doivent être abandonnées. Les retransmettre par la suite, consomme de l'énergie. Tous les protocoles MAC essayent à leur manière d'éviter les collisions. Les collisions concernent plutôt les protocoles MAC avec contention.

2. **L'écoute à vide (*idle* listening) :** un nœud dans l'état « *idle* » est prêt à recevoir un paquet, qui n'arrive pas. Ceci est coûteux et inutile dans le cas des réseaux à faible charge de trafic. Plusieurs types de radios présentent un coût en énergie significatif pour le mode « *idle* ». Eteindre la radio est une solution, mais le coût de la transition entre les modes consomme également de l'énergie, la fréquence de cette transition doit alors rester « raisonnable ».

3. **L'écoute abusive (Overhearing) :** cette situation se présente quand un nœud reçoit des paquets qui ne lui sont pas destinés. Le coût de l'écoute abusive peut être un facteur dominant de la perte d'énergie quand la charge de trafic est élevée et la densité des nœuds est grande, particulièrement dans les réseaux « *Mostly-on* ».

4. **L'overmitting :** un nœud envoie des données et le nœud destinataire n'est pas prêt à les recevoir.

5. **L'overhead des paquets de contrôle :** l'envoi, la réception, et l'écoute des paquets de contrôle consomment de l'énergie. Comme les paquets de contrôle ne transportent pas directement des données, ils réduisent également le débit utile effectif.

Comme nous l'avons souligné ci-dessus, c'est la transmission de données qui se révèle extrêmement consommatrice par rapport aux tâches du nœud-capteur. Cette caractéristique conjuguée à l'objectif de maximisation de la durée de vie du réseau a suscité de nombreux travaux de recherche. Avant de citer ces travaux dans le chapitre suivant, nous introduisons dans ce paragraphe certains mécanismes de base que l'on utilisera pour la réalisation de notre projet :

4.2. Mode d'économie d'énergie :

Ce mode est possible quelle que soit la couche MAC adoptée. Cela consiste à éteindre le module de communication dès que possible. Par exemple, des protocoles MAC fondés sur la méthode TDMA (Time Division Multiple Access) offrent une solution implicite puisqu'un nœud n'échange des messages que dans les intervalles de temps qui lui sont attribués. Il peut alors garder sa radio éteinte durant les autres slots. Comme nous l'avons souligné précédemment, il faut toutefois veiller à ce que le gain d'énergie obtenu en mettant en veille le module radio ne soit pas inférieur au surcoût engendré par le redémarrage de ce module.

4.3. Duty-cycling :

Cette technique est principalement utilisée dans l'activité réseau. Le moyen le plus efficace pour conserver l'énergie est de mettre la radio de l'émetteur en mode veille (**low-power**) à chaque fois que la communication n'est pas nécessaire. Idéalement, la radio doit être éteinte dès qu'il n'y a plus de données à envoyer et ou à recevoir, et devrait être prête dès qu'un nouveau paquet de données doit être envoyé ou reçu. Ainsi, les nœuds alternent entre périodes actives et sommeil en fonction de l'activité du réseau. Ce comportement est généralement dénommé **Duty-cycling** [27]. Un **Duty-cycle** est défini comme étant la fraction de temps où les nœuds sont actifs.

Comme les nœuds-capteurs effectuent des tâches en coopération, ils doivent coordonner leurs dates de sommeil et de réveil. Un algorithme d'ordonnancement Sommeil/Réveil accompagne donc tout plan de **Duty-cycling**. Il s'agit généralement d'un algorithme distribué reposant sur les dates auxquelles des nœuds décident de passer entre l'état actif et l'état sommeil. Il permet aux nœuds voisins d'être actifs en même temps, ce qui rend possible l'échange de paquets, même si les nœuds ont un faible duty-cycle (i.e., ils dorment la plupart du temps).

4.4. L'agrégation de données dans les réseaux de capteurs sans fil:

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 éventuellement capter les mêmes données (température, pression, humidité équivalentes par exemple) il apparaît donc nécessaire d'introduire le mécanisme d'agrégation de données afin d'éviter la redondance d'information au sein du réseau de capteurs et ainsi préserver leur énergie afin d'augmenter la durée de vie du réseau.

4.4.1 Définition :

L'agrégation de données dans les réseaux de capteurs consiste à remplacer les lectures individuelles de chaque capteur par une vue globale, collaborative sur une zone donnée (Clustering) [3].

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 la station de base qu'un seul message résumant l'information contenue dans ces n messages.
Ceci réduit le nombre de messages envoyés de façon considérable et donc économise l'énergie.

Exemple d'acheminement des données avec et sans agrégation :

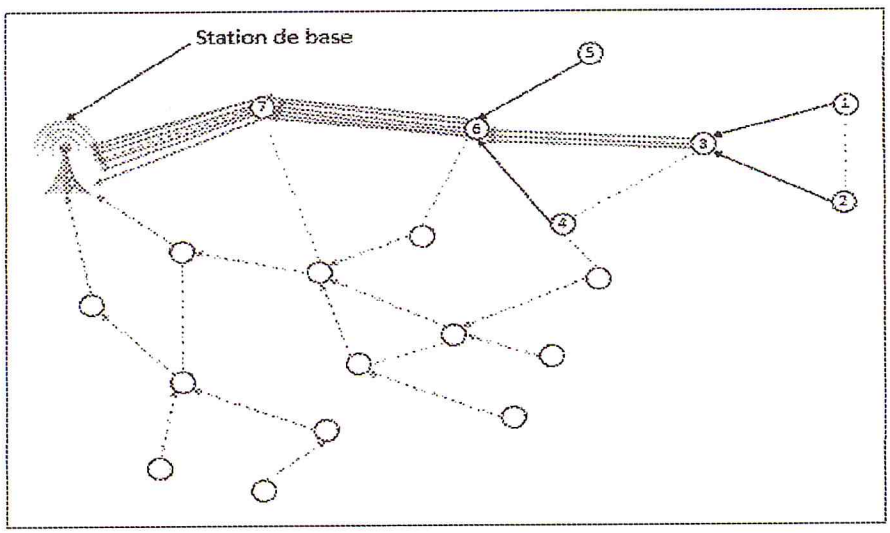

Figure 2.3 : « Acheminement des messages sans agrégation »

Au total, 20 messages ont été envoyés sur le réseau de capteurs, en fonction de la profondeur (nombre de saut jusqu'à la station de base) le nombre de messages augmente, et la consommation d'énergie (nombre de messages reçu et envoyée) des nœuds proche de la station de base (6, 7 …) est nettement supérieur que celle des nœuds les plus éloignée (1, 2 …).

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

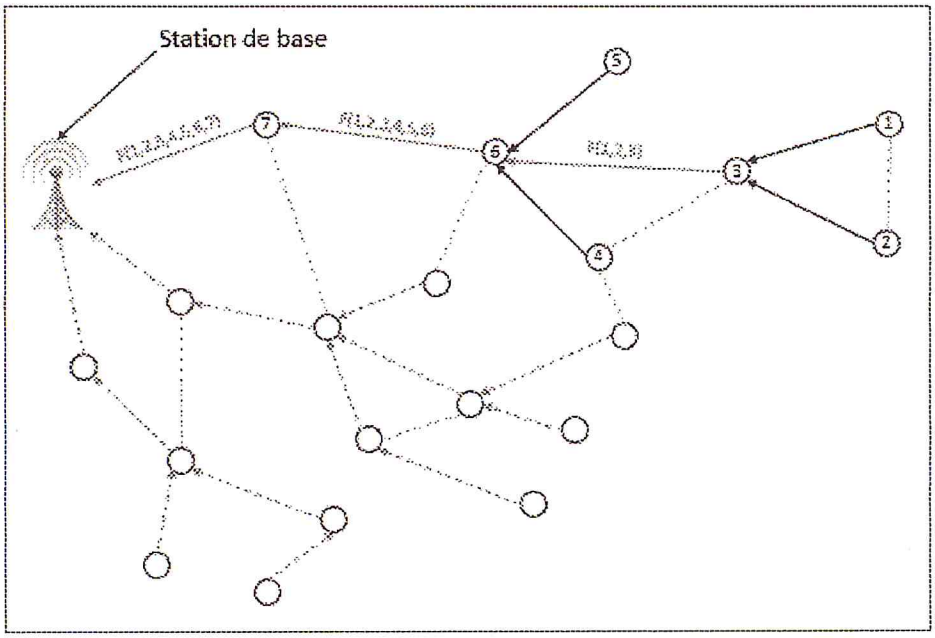

Figure 2.4 : « Acheminement des messages avec agrégation »

4.4.2. Types d'agrégation de données

Les techniques d'agrégation de données peuvent êtres découpées en deux :

4.4.2.1 Agrégation centralisée :

Agrégation dans des clusters, formés via un protocole de Clustering (classification). On définit d'abord des zones (clusters) via ce protocole puis ensuite on agrège les données dans ces zones grâce à un chef de zone. Ce chef peut éventuellement changer au cours du temps afin de répartir au mieux la consommation d'énergie entre tous les nœuds du réseau. Ce genre de fonction d'agrégation est utilisé pour avec des fonctions tel que Min, Max ou Moyenne …etc.

4.4.2.2. Agrégation distribuée :

L'agrégation dans un arbre, c'est-à-dire que le réseau est vu de manière globale, cette technique est utilisée dans des fonctions d'agrégat plus global tel que le calcul de la médiane ou encore l'écart-type.

4.4.3. Différence entre l'agrégation de données et le routage (l'acheminement de données)

Le routage dans les réseaux de capteurs est un routage multi-sauts. L'acheminement des paquets d'une source donnée à une destination se fait à travers plusieurs nœuds intermédiaires. Ainsi, un nœud consomme de l'énergie soit pour transmettre ces données ou pour relayer les données des autres nœuds. Ainsi les nœuds proches de la station de base devront effectuer plus de taches que les autres. Etant limités en ressources, il n'est pas rare que ces nœuds (proches de la station de base) se trouvent dans l'état où ils ne peuvent plus effectuer toutes ces taches. Cette situation est dite *congestion* [28].

Il existe plusieurs sources de congestion comme le débordement des mémoires tampon, les transmissions concurrentes ou la collision de paquets. Parmi les problèmes causés par la congestion, on peut citer: le retardement de l'information, la perte de paquets qui contiennent parfois des informations critiques, gaspillage de la bande passante. Il est donc facile de constater que la congestion dégrade la performance du réseau en l'empêchant de garantir certaines exigences QoS (quality of service) comme le temps réel, le routage de bout en bout, la maximisation de la durée de vie du réseau [29].

Il est évident que ces deux modes d'acheminement (routage et l'agrégation) sont différents, chacun opère selon un principe de fonctionnement distinct, par exemple lors de la détection d'un feu de forêt l'information de la température excessive devient redondante et sera communiquée par tous les nœuds se trouvant à proximité, Dans ce cas un algorithme d'agrégation sera nécessaire pour éliminer les informations superflues. Par contre si nous cherchons à surveiller la progression des flammes, un routage est nécessaire afin de transmettre en temps réel les températures récoltées par chaque nœud.

4.4.4. L'impact de l'agrégation de données sur le délai

L'objectif de l'agrégation est de réduire la quantité de données (le nombre de paquets et leur taille) à transmettre sur le réseau et ainsi réduire l'énergie consommée [27]. La diminution du nombre de paquet circulant sur le réseau va entrainer la baisse du nombre des collisions lors de la transmission des messages et affecter directement les délais de transmission des paquets.

5. 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 leurs domaines d'application tels que la santé, l'environnement, l'industrie, etc. Dans ce chapitre, nous avons présenté les *RCSFs*, leurs architectures de communication. Les *RCSFs* sont confrontés à une multitude de défis, parmi lesquels la planification de l'agrégation de données que nous allons traiter d'une manière approfondie dans le chapitre suivant en énumérant les différents protocoles traitant ce type d'acheminement de données

CHAPITRE -III-

Étude des différents protocoles de planification de l'agrégation

1. Introduction

Dans ce chapitre nous allons nous focaliser sur la planification de l'agrégation dans les réseaux de capteurs sans fil (*RCSF*), Nous allons voir les différents algorithmes et protocoles existant qui traitent ce problème. Afin de minimiser la latence certains travaux sur ce problème ont adoptés une approche structurée, Une structure en arbre a été utilisée en tant qu'entrée pour l'algorithme de planification. Comme la performance de planification dépend principalement de l'arbre d'agrégation fourni. Ces approches doivent garantir des performances optimales. Ensuite nous allons voire une classification de ces algorithmes avant de conclure ce chapitre.

2. Études et comparaison des protocoles existant

2.1. NCA (Nearly Constant Approximation)

Huang et al [5], ont proposé une solution centralisée appelé *NCA* (*Nearly Constant Approximation for data agrégation scheduling*), afin de minimiser la latence de l'agrégation de données. Basée sur les informations Hop-Count (nombre de saut) le réseau est divisé en niveau, où chaque niveau contient un ensemble de nœuds avec le même Hop-Count vers la station de base. En supposant que la station de base soit au centre de la topologie (c.-à-d. ce trouvant au centre du réseau), la borne maximum de latence (temps d'attente) atteint par NCA est $23R+\Delta-18$ (où R est le rayon du réseau et Δ est le degré du réseau). Un nœud i au niveau L (L est le plus court chemin entre le nœud i et la Station de Base) a uniquement des liens avec les nœuds du niveau $(L-1)$, (L) et $(L+1)$. Afin de construire l'arbre d'agrégation, les nœuds sont regroupés en ensembles maximaux indépendants. Cet ensemble est construit en coloriant le maximum de nœuds indépendants pour chaque niveau en noir, en commençant par le premier niveau, qui contient uniquement la station de base. Un ensemble maximal de nœuds indépendants ne doit ne pas avoir de liens avec l'ensemble maximal indépendant des niveaux $(L+1)$ et $(L-1)$. Ensuite, NCA prend quelque nœuds du niveau L et les colorie en bleu pour interconnecter les nœuds Noir du niveau L+1 avec les nœuds noir du niveau L et L-1, et ainsi de suite jusqu'à ce que l'arbre d'agrégation soit construit. Apres cela, *NCA* exécute

la planification de l'agrégation de données basé sur l'arbre construit en utilisant la procédure de planification préliminaire. Premièrement tous les nœuds blancs sont planifiés afin que les collisions soient évitées. Commençant par le niveau le plus bas, les nœuds noirs et les nœuds bleus sont planifiés niveau par niveau utilisant la procédure de planification préliminaire. Cette dernière est exécutée après l'itération. A chaque itération, un ensemble de nœuds est planifié ensemble au même slot. Un nœud u est planifié au slot t s'il n'a aucun lien avec un parent d'un nœud déjà planifié pendant ce slot. Néanmoins, seul l'effet de u sur le nœud planifié est testé (il n'y as pas de test sur l'effet du nœud planifié sur le parent de u), comme il a été prouvée dans [30], *NCA* ne garantit pas une planification sans collision dans plusieurs cas, de même qu'il existe plusieurs erreurs dans la procédure de planification.

2.2. DAS (Distributed data Aggregation Scheduling)

Yu et al.[30] Ont proposé une solution distribuée, appelée *DAS* (*Destributed data agrégation scheduling in Wireless Sensor network*), pour la planification de l'agrégation de données dans un *RCSF*. Ce protocole génère une planification sans collision avec une latence qui atteint $24D+6\Delta+16$ (où D est le diamètre du réseau et Δ le degré maximum des nœuds). Dans la phase de construction de l'arbre d'agrégation, l'arbre généré par *DAS* ressemble à celui généré avec *NCA* [5]. L'arbre utilisé par *DAS* est construit de manière distribuée en utilisant l'algorithme proposé dans [31]. Dans la phase de planification de l'agrégation, afin de planifier tous les nœuds de manière distribuée, l'auteur définit l'ensemble concurrent du nœud u en tant que nœuds qui ne peuvent pas être planifié durant le même slot, à cause des collisions. Les auteurs ont prouvé que cet ensemble inclus tous les voisins du parent de u et tous les enfants des voisins de u. Un nœud est prêt à être planifié, si et seulement s'il est un nœud feuille ou si tous ses voisins ont été planifiés. Les nœuds qui sont planifiés en premier lieu sont ceux qui ont l'identifiant le plus petit. Après qu'un nœud donné soit planifié à un slot donné, tous les nœuds dans son ensemble de concurrents classifient ce slot comme slot interdit, et effacent ce nœud de leur ensemble de concurrents.

2.3. ACS (Aggregation Converge-cast Scheduling)

Malhotra et Al. [32] Ont proposé un protocole centralisé pour la planification de l'agrégation appelé *ACS* (agrégation converge-cast scheduling in Wireless Sensor networks). Basé sur l'observation que le chemin dans un ensemble d'arbre dominant (*DST*: *dominating set tree*) en partant des nœuds vers la station de base n'est pas nécessairement optimal, les auteurs suggèrent la création d'un arbre équilibré du plus court chemin (*BSPT*: *Balanced shortest path tree*) plutôt que *DST* comme arbre d'agrégation de données. Ainsi, le plus court chemin pour connecter un nœud à la station de base est utilisé pour minimiser le nombre de slot. En outre, l'affectation parent-fils de la même profondeur est balancée et équilibrée dans toute la mesure afin de minimiser le temps d'attente. Après la construction du *BSPT*, un algorithme de planification est lancé pour planifier chaque nœud de l'arbre. L'algorithme commence par planifier les nœuds prêts à être planifiés, lesquels sont seulement des nœuds feuilles qui ont le plus de liens avec le niveau suivant. Après cela, l'algorithme efface les nœuds planifiés du graphe du réseau et va au slot suivant. S'il existe des nœuds admissibles (se trouvant dans le nouveau graphe), l'algorithme les planifie et ainsi de suite jusqu'à la fin de l'algorithme de planification.

2.4. PDA (Peony-tree-based aggregation)

Une nouvelle solution centralisée est appelée *PDA* (*Peony-tree-based data Aggregation*) avec une latence qui atteint $15R+\Delta-15$, démontrée ici [33]. En utilisant les informations du hop-count (nombre de saut), les nœuds du réseau sont divisés en niveaux. Le premier niveau (*L0*) contient uniquement la station de base, et le denier niveau (*Lr*) contient uniquement les nœuds feuilles. Afin de créer l'arbre d'agrégation, un ensemble maximum indépendant est sélectionné en premier lieu à partir du graphe *G*. Cet ensemble est construit de haut en bas. Initialement, *D* contient uniquement la station de base. Débutant par le premier niveau jusqu'au dernier, pour chaque nœud *u* du niveau *Li*, le test suivant est effectué:

Si *u* est indépendant de tous les autres nœuds de *D*, il est ajouté à *D* et retiré du graphe *G* ;

Tous les voisins de *u* sont également retiré de *G*, autrement *u* sera retiré de *Li*.

Enfin, au niveau *Lr*, tous les nœuds dominant de *D* seront sélectionnés. Afin d'interconnecter les nœuds dominant du niveau *Li* (*Di*) avec les nœuds dominant du niveau *L-1* (*D-1*), un ensemble de nœuds non redondants appelés connecteur (*C* : connector) sont sélectionnés. Ici, un nœud connecteur x est dit non redondant pour un dominateur u, si le retrait de c déconnectera au moins l'un des voisins dominant de deuxième saut de u. Les nœuds restants *(G- (D ∪C)*) sont noté nœud dominant (*W*). Pour chaque nœud dominant u au niveau *Li*, ses parents sont sélectionnés à partir de ses voisins $D \cap Li\text{-}1$. Apres cela la planification de l'agrégation est exécutée utilisant l'algorithme préliminaire (*first-fit*) en deux étapes:

1. Tous les nœuds dominant sont subdivisé en ensembles maximum de nœuds concurrents disjoint W1, W2, W3, …, Wh. Un ensemble Wi est constitué du maximum de nœuds concurrents disjoint, si $\exists$ u $\in$ Wi , alors $\nexists$ v $\in$ Wi tel que le parent de u (resp. v) est voisin de v (resp. u) pour cela, les nœuds dans Wi peuvent transmettre des données à leur parents sans interférence au iéme slot.
2. Les nœuds dominant et connecteurs sont planifié niveau par niveau commençant par le niveau Lr. Les dominants ainsi que les connecteurs à chaque niveau sont aussi subdivisé en ensembles maximum de nœuds disjoint et en suite planifié.

2.5. SDA (Shortest Data Aggregation)

Chen et Al [4] ont proposé une heuristique centralisé appelé *SDA*, qui réduit la latence á $((\Delta\text{-}1) \times R)$ (où Δ est le degré de nœud maximale et *R* est le rayon de réseau). Dans *SDA*, l'Arbre du Plus Court chemin *APC* est d'abord crée. Ensuite l'arbre crée est utilisé comme une entrée pour l'algorithme de planification. Remarquer que *APC* ne sera pas utilisé comme un arbre d'agrégation de données, il ne sera exploité que pour trier les nœuds de réseau dans le processus de planification. Ce dernier est exécuté en plusieurs itérations, qui visent à déterminer l'ensemble de nœuds qui peuvent être planifiés au même slot. Les nœuds feuilles de l'APC sont classés selon le nombre de leurs voisins non-feuilles, où le nœud qui a le plus grand nombre de voisins non-feuilles est d'abord considéré. Soit *S* et ($\overline{S}$) désignent respectivement l'ensemble des nœuds feuilles triés et nœuds non-feuilles dans l'APC. Pour tout nœuds u dans *S*, le test suivant sera effectué :

si $\exists\ v \in \bar{S}$ ou u est le voisin exclusif de v (i.e., $\nexists$ w $\in S - \{u\}$, dans lequel w est un voisin de v), alors u est planifié et v est sélectionnée comme parent dans ce slot. Ici, u est retiré de l'APC. Lorsque tous les nœuds dans S sont testés, les nœuds feuilles qui ne sont pas planifié ainsi que les nouveaux nœuds feuilles (dans l'APC) sont considérés pour la prochaine itération, et ils sont triés et testés en utilisant la même approche.

2.6. ISDA (Improved SDA)

Basé sur [4] *Zhu et Al.* Ont proposé une nouvelle solution centralisée, appelé ISDA (une amélioration de SDA) dans [36], qui vise à minimiser la latence des données. Ce protocole génère une planification sans collision avec un temps de latence de données de $(7\Delta / \log2\ (|N|))^*$ (R-1) pour un arbre d'agrégation, ou R est le rayon du réseau et N est le nombre de nœuds, Δ est le degré du réseau. ISDA commence par diviser le réseau en cellules. Chacune d'entre elle peut contenir un ensemble de nœuds. Dans la première étape pour maintenir la communication inter-cellule, ISDA créé un arbre du plus court chemin APC dans le graphe G dont les sommets sont les cellules. Le lien entre le parent et son fils est réalisé par la sélection d'une paire arbitraire de nœuds voisins auprès du parent et fils cellule. A la fin, un APC est créé sur les cellules, lorsque chaque cellule contient un nœud fils et chaque cellule sans feuille contient au moins un nœud parent. La deuxième étape consiste à établir la connexion entre chaque parent et le nœud fils dans chaque cellule, en utilisant un ensemble de nœuds à partir de l'arbre d'agrégation.

Cependant, il y-a des nœuds faisant partie du réseau qui ne peuvent pas se joindre à l'arbre d'agrégation. Afin de résoudre ce problème, ces nœuds se joignent à l'arbre d'agrégation en tant que des nœuds feuilles, ils seront alors les premiers à être planifiés. Pour planifier le reste des nœuds de l'arbre, les auteurs utilisent le protocole SDA proposé dans [4].en combinant TDMA assure que le slot du parent est supérieur à ceux des enfants. Dans ISDA tous les nœuds feuilles doivent être planifiés avant les autres.

2.7. CIAS (Centralized Improved data Aggregation Scheduling)

Xu et Al [34], ont proposé une nouvelle solution centralisée, appelée *CIAS* (*Centralized Improved data agrégation scheduling*), qui vise à minimiser la latence de l'agrégation en utilisant une topologie *CDS*. Les auteurs ont choisi le centre de la topologie du réseau comme racine de l'arbre d'agrégation au lieu de la station de base. Ce choix est fait afin de réduire la borne du temps de latence du *CIAS* en fonction du rayon du réseau R au lieu du diamètre Δ. Le temps de latence avec *CIAS* est a $16R+\Delta-14$. Ici il est important de noter que *Bagaa et al.* ont prouvé dans [35] que la borne supérieure de latence avec *CIAS* est incorrecte.

CIAS est réparti en deux phases: la construction de l'arbre dominant et la planification de l'agrégation. L'ensemble dominant est construit en premier lieu en la même approche que [31]. Ensuite, la planification de l'agrégation est effectuée en deux étapes successives. Lors de la première, les données sont toutes agrégées et envoyées des nœuds nominés vers les nœuds dominants planifiant le nombre maximum de couples (dominant, dominé) simultanément et cela pour chaque slot. Lors de la seconde, les nœuds dominants agrègent leurs données vers les nœuds racines niveau par niveau en utilisant un ensemble non redondant de nœuds dominés que nous appelons connecteurs. Ce dernier est utilisé afin de connecter chaque nœud dominant aux autres nœuds dominants se trouvant à deux sauts plus loin. Ici un connecteur x (nœud dominé d'un dominant u) est dit non redondant pour le nœud dominant u, si le retrait de x conduira à déconnecter u d'au moins un nœud dominant se trouvant à 2 sauts plus loin.

2.8. SAS (Sequential Aggregation scheduling)

Wan et Al [37]. Ont proposé solution centralisée appelée SAS (Séquentiel agrégation scheduling). Cette solution génère une planification sans collision avec un temps de latence de $15R + \Delta - 4$, (où R est le rayon du réseau et Δ est le degré maximum des nœuds. Comme pour CIAS, Bagaa et al [35] ont prouvé que la borne supérieure du temps de latence de SAS est incorrecte. SAS, comme solution précédente est exécutée en deux étapes: la construction de l'arbre d'agrégation et la planification de l'agrégation. La première est construite utilisant l'approche proposée en [31]. La seconde planifie tous les nœuds dominants utilisant la même approche que [34]. Afin de faire suivre les données agrégés à partir des nœuds dominants vers la station de base, le réseau est divisé en

niveaux tel que les nœuds du même niveau sont les dominants (resp connecteur non-redondant) .Les dominants et les connecteurs sont planifiés niveau par niveau, de manière ascendante en partant du plus bas niveau jusqu'à la station de base comme suit:

1. Pour chaque nœud dominant (resp connecteur non redondant) *u* du niveau *i*, son parents p(*u*) est sélectionné à partir du niveau *i*-1 comme connecteur non redondant (resp. dominant) dont l'identifiant est le plus petit. A la fin, deux ensembles seront généré; le premier (*Ni*) est constitué de dominateurs (resp connecteurs non-redondant) dans le niveau i, tandis que le second (*Pi*) est constitué de parents de *Ni*.

2. L'ensemble des liens à partir des nœuds de Ni vers ceux de Pi est noté Ai.

3. Le graph de conflit (*conflict graph*) de *Ai*, noté CG(Ai) = (V, E), est généré. V est l'ensemble des liens (u, p(u)) de Ai et E est l'ensemble d'arrêt conflictuel entre eux. Ici, il existe une arrêt entre (u, p(u)) $\in$ V et (v, p(v)) $\in$V, si et seulement si il y a un conflit entre eux (u, p(u)) $\in$Ai ou (v, p(u)) $\in$Ai.

4. La coloration préliminaire de CG(Ai) est calculée. Alors, chaque lien de Ai avec la couleur j est planifié au J^{éme} slot du i^{éme} niveau.

3. Classification des algorithmes

Chacun des algorithmes précèdent utilise une approche spécifique, tel que certains sont basés sur les ensembles dominant, d'autres utilisent des topologies non structurées ou même semi-structurées. La classification des algorithmes cités ci-dessus se fait en premier lieu selon le type de sélection de parent qu'ils adoptent, tel que les algorithmes DAS, NCA, PDA, CIAS et SAS ont des critères de sélection basés sur les ensembles dominants de telle sorte qu'un nœud faisant partie d'un ensemble dominant ne peut choisir que l'un des nœuds faisant partie de l'ensemble dominé, les algorithmes ACS, SDA et ISDA ont un critere de sélection libre (Spaning tree based).

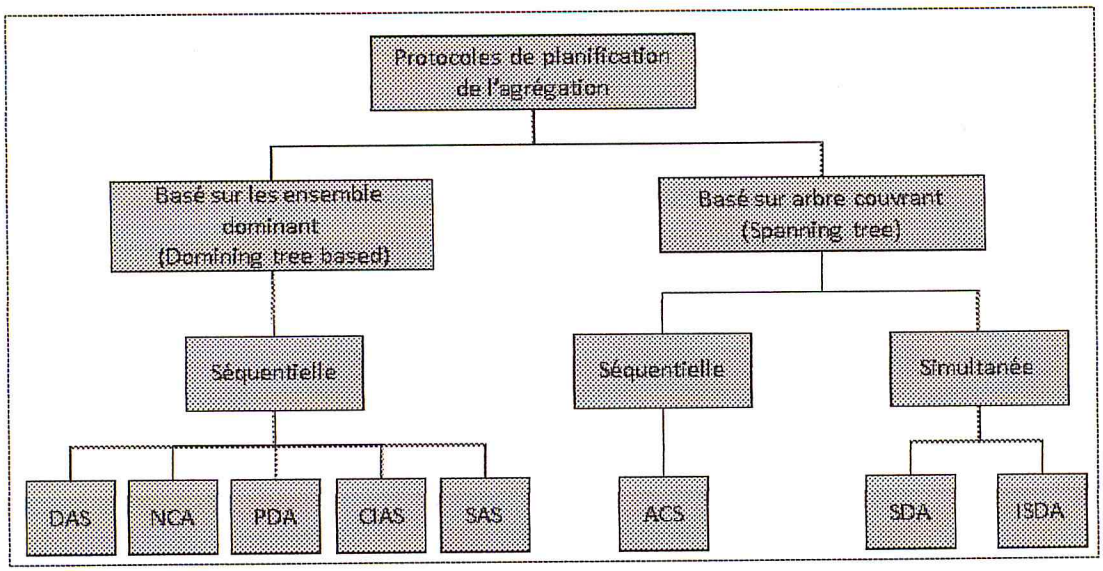

Figure 3.1 « Classification des algorithmes »

En second lieu les algorithmes sont classifiés selon l'ordre dans lequel est fait la construction de l'arbre et la planification des nœuds, tel que SDA et ISDA voient la construction de l'arbre et la planification des nœuds simultanément, les autres algorithmes voient les deux étapes se dérouler séquentiellement.

4. Conclusion

La majorité des algorithmes précédents de planification de l'agrégation ont adopté une approche séquentielle, dans laquelle des structures en arbre ont été utilisé comme entrée pour les algorithmes de planification. Ces approches forcent l'algorithme de planification à suivre un ordre d'agrégation imposé par l'arbre d'agrégation, nous verrons dans le chapitre suivant que l'algorithme que nous avons implémenté peut surpasser tous les protocoles précédemment cités en termes de latence ainsi qu'en durée de vie du réseau, Cela prouve que les topologies non-structurées ont un grand potentiel pour la planification de l'agrégation dans les réseaux de capteurs sans fil.

CHAPITRE -IV-

Implémentation du protocole *DSCA*

1. Introduction

Un réseau de capteurs sans fil *RCSF* est composé d'une multitude de nœuds capteurs qui peuvent récolter des informations sur l'environnement avoisinant, mesurer les conditions ambiantes et rapporter certains phénomènes de leurs environnements. Les informations récoltées sont transmises vers la station de base qui agit comme un nœud central afin de collecter toutes les données. Néanmoins les nœuds capteurs souffrent de limitations en ressources comme nous l'avons cité dans le chapitre -II-. Le design du réseau doit donc être optimisé pour pouvoir répondre à ses contraintes et fournir des solutions pertinentes. Le défi principal de l'optimisation de la planification de l'agrégation dans les *RCSFs* est de définir une planification sans-collision avec le nombre minimum possible de *Slots*. Chen et Cie [4] ont prouvé que le temps minimum d'agrégation de données est un problème NP- Difficile. Les travaux évoqués (chapitre précédent) tel que, NCA[4], SAS[37], DAS[30], CIAS[34] et ACS[32] ont opté pour une approche structuré basé arbre, dans lesquels le réseau est organisé en niveaux ainsi que des ensembles de pairs (parent , fils) composant l'arbre qui sont utilisé en tant qu'entrée par l'algorithme de planification. Chaque niveau contient l'ensemble de nœuds avec le même nombre de sauts jusqu'à la station de base, de telle sorte que les performances liées à un arbre d'agrégation fourni, ne peuvent pas garantir des performances optimales.

Sur la quête d'une planification sans collision de tous les nœuds capteurs et l'acheminement des informations recueillies vers la station de base, dans un minimum de temps, et contrairement à toutes les solutions antérieures qui séparent les phases de construction de l'arbre et le processus de planification des nœuds, celle proposé ici effectue les deux processus en parallèle. Par conséquent, la réutilisation des slots est considéré à la fois pour la planification et pour la construction de l'arbre, cela garantit de meilleures performances par rapport à la planification sur un arbre fixe. En d'autres termes, l'arbre d'agrégation est dynamiquement lié à la planification, tout en maximisant la réutilisation des *Slots* et en réduisant les conflits de planification. Afin de réaliser l'implémentation pour de véritables Motes, *DSCA* est encapsulé dans un Framework appelé *DADIDA*. En plus de la simulation que nous verrons dans le Chapitre -V-, *DSCA* est la première solution

parmi toutes les solutions existantes qui ont été implémentée sur des capteurs réels en utilisant la plateforme *TinyOS 2.x* [6].

Dans la première partie de ce chapitre nous verrons une description de l'algorithme *DSCA* ainsi que les détails du fonctionnement et de l'implémentation de l'algorithme, nous verrons également un exemple de déroulement de DSCA et enfin nous expliquerons les détails du Framework que nous avons conçu et réalisé afin de permettre à *DSCA* de fonctionner sur des nœuds capteurs réels.

2. DSCA (Distributed Simultaneous tree Construction and data Aggregation scheduling)

Alors que l'agrégation des données peut introduire une latence supplémentaire à chaque nœud pour attendre les transmissions de ses enfants, certaines applications critiques telles que la détection précoce des incendies de forêt [9] et la surveillance de sécurité [23], alarmes sur les événements en temps réel, doivent être déclenchées et acheminées vers la station de base avec un minimum de latence, Afin qu'une action appropriée puisse être prise.

Des travaux précédents dans l'agrégation des données dans un réseau, ont adoptés un arbre de routage équilibré sans tenir compte du problème de niveau de liaison [38], [39]. Récemment, des solutions Cross-Layer (ou inter-couches) qui combinent la formation de l'arbre d'agrégation au media d'accès de planification sans collision ont été proposées [4-5-30 -32], Cette combinaison a un impact positif sur la latence des données et de la consommation d'énergie. Toutes les solutions précédemment proposées ont adopté une approche séquentielle, où une structure d'arbre est réalisée en premier puis les nœuds du réseau sont planifiés sur l'arbre produit. Comme l'arbre agrégation a un impact sur la planification des nœuds, cette approche ne peut garantir des performances optimales. Dans cette partie nous aborderons ce problème et nous étudierons une solution appelée DSCA (Distributed Simultaneous tree Construction and data Aggregation scheduling).

La solution proposée repose sur trois éléments clés de la conception :

(1) fonctionnalités réparties

(2) l'exécution simultanée de la construction de l'arbre d'agrégation et la planification

(3) Critères de sélection des parents qui maximalise les choix des parents pour chaque nœud et maximise la réutilisation des slots.

En plus de DSCA, nous avons réalisé un Framework général qui encapsule DSCA, ce Framework, que nous avons appelé DADIDA pour (Delay-Aware Distributed Data Aggregation in Wireless Sensor Network) est le premier qui considère tous les aspects de l'agrégation de données, de telle sorte que toutes les solutions précédentes sont limitées aux phases de construction de l'arbre et de la planification des nœuds. En outre, l'implémentation et l'expérimentation sur un Test-Bed contenant des nœuds capteurs réels est une caractéristique unique du protocole proposé par rapport dans l'état de l'art.

2.1. Modèle réseau

Les communications entre les nœuds sont modélisées sous forme de graphes : $G = (V, E)$, où, V est l'ensemble de tous les nœuds dans le réseau (Station de base comprise). Un nœud dans V, notée, B, est la station de base ayant responsabilité de transmettre les données collectées à partir des nœuds du réseau à l'utilisateur final.

Une arête, $(u, v) \in E$, est définie si et seulement si les deux nœuds se trouvent dans la portée de transmission, r, de l'autre. Chaque nœud est supposé avoir un émetteur-récepteur unique qui lui permet de transmettre et de recevoir des paquets. Cet émetteur-récepteur ne permet pas de transmettre et de recevoir simultanément, ni la transmission ou la réception de plus d'un paquet à la fois. Tous les nœuds sont supposés avoir la même portée d'émission, r.

2.2. Description de l'algorithme

La Construction de l'arbre et la planification des nœuds sont séparées et sérialisés dans toutes les solutions proposées jusqu'à présent. C'est la structure de l'arbre qui est d'abord construite, puis les nœuds du réseau sont planifiés, obligeant la planification d'agrégation de suivre un ordre particulier imposé par l'arbre d'agrégation et ne

garantissant pas la minimisation du nombre de slots. Contrairement aux solutions précédentes, la construction de l'arbre d'agrégation des données et le processus de planification de nœuds s'entrelacent dans DSCA. Avant de commencer l'exécution DSCA, chaque nœud doit avoir les informations suivantes:

(1) l'ensemble des voisins de u, que nous noterons γ_u.

(2) son niveau dans le réseau.

Un nœud u est dans au niveau N, si et seulement si il est N saut à partir de la station de base en fonction du premier ordre de classement. Le détail de la découverte de voisinage et l'organisation des nœuds en niveaux seront donnés dans la seconde partie de ce chapitre. Les voisins définis γ_u, peuvent être subdivisés en trois sous-ensembles disjoints d'après le niveau de u :

(1) l'ensemble des voisins de u appartenant au même niveau, dénoté par $\gamma_u\ [L]$.

(2) $\gamma_u\ [L-1]$ l'ensemble des voisins de u appartenant au niveau suivant.

(3) $\gamma_u\ [L+1]$ l'ensemble des voisins de u appartenant au niveau précédent.

Quand un nœud enfant envoie des données à son nœud parent au cours de l'intervalle de temps t, t est appelé le *Slot* de transmission (respectivement *Slot* de réception) pour le nœud enfant (resp. le nœud parent). Soit TS_u qui désigne le *Slot* pour l'envoi des données à partir du nœud u vers son parent, Grâce au chevauchement de la construction de l'arbre et la planification des nœuds, les voisins d'un nœud u, c.-à-d. les nœuds dans l'ensemble γ_u, peuvent être subdivisés en trois sous-ensembles disjoints selon leur *Slot* t, (c.-à-d. $TS_u = t$). Tout d'abord, l'ensemble des nœuds $NeighborSC_u[t]$ qui (a) ne sont pas sélectionnées comme parent pendant le Slot t et (b) ne peuvent pas recevoir à pendant ce Slot en raison de collisions. Chaque nœud dans $NeighborSC_u[t]$ a au moins un voisin qui a déjà désigné son parent et définie son Slot t (c.-à-d. un nœud planifiée). Deuxièmement, l'ensemble des nœuds $ParentSC_u[t]$ qui sont sélectionnés en tant que parents par d'autres nœuds déjà planifiés pendant le Slot t, Les voisins de cet ensemble ne peuvent pas utiliser t en raison de collisions. Enfin, l'ensemble des nœuds candidats $CandidateParent_u[t]$ ceux qui peuvent être des parents du nœud u au cours du Slot t, Formellement, ce dernier ensemble est défini par :

$$CandidateParents_u[t] = \gamma_u - (NeighborSC_u[t] \cup ParentSC_u[t] \cup £_u[t])$$

De telle sorte que $£_u[t]$ est l'ensemble des voisins de u qui sont (1) planifiée et, (2) leur

Slot est inférieur à t, Nous retirons, $£_u[t]$ de $CandidateParents_u[t]$ pour assurer la fraîcheur des données et éviter la création de cycles. Si $CandidateParents_u[t] \neq \emptyset \vee ParentSC_u[t] = \emptyset$ le nœud u devrait sélectionner un Slot supérieur à *t* afin d'éviter d'entrer en collision avec les nœuds appartenant à l'ensemble $ParentSC_u[t]$. Sinon, ie, $CandidateParents_u[t] \neq \emptyset \vee ParentSC_u[t] \neq \emptyset$ nœud u sélectionne son parent P_u à partir de $CandidateParents_u[t]$.

En plus de TS_u ,P_u ,γ_u ,$NeighborSC_u[t]$, $ParentSC_u[t]$, et $CandidateParents_u[t]$ défini précédemment, les notation suivante seront utilisée dans la description:

- n_i : est l'identifiant du nœud *i*.
- (u, v) : Lien entre les nœuds u et v.
- SC_u : Ensemble de nœuds dans γ_u qui sont plannifieé, c.-à-d., les nœuds qui ont déjà sélectionnent leurs parents et fixent leur Slot.
- $\overline{SC_u}$: Ensemble de nœuds dans γ_u qui ne sont pas planifiée, c.-à-d., les nœuds dans γ_u qui n'ont pas encore choisi leurs parents.Formellement : $\overline{SC_u} = \gamma_u - SC_u$
- TC_u : Le *Slot* maximale des enfants de u, initialisé à zéro.
- $State_u$: l'état du nœud u. Chaque nœud dans le réseau peut être dans l'un des états suivants: (1) *Not_Ready_State* (notée NRY), (2) *Wait0_State* (notée W0), (3) *Ready_State* (notée RY), (4) *Wait1_State* (notée W1), et (5) *Scheduled_State* (notée SC).

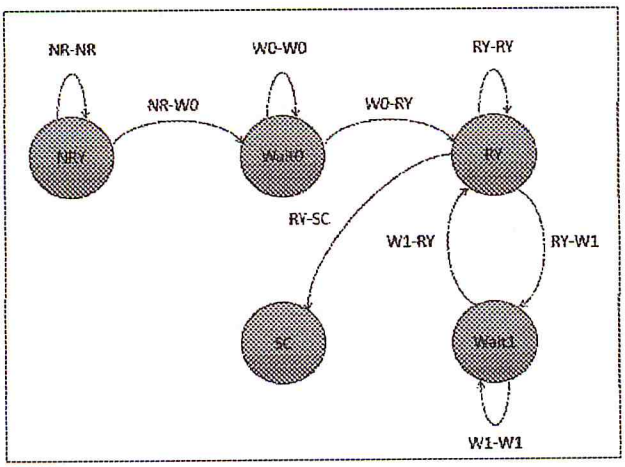

Figure 4.1 « Automate d'état du comportement d'un nœud »

Pendant l'exécution de DSCA, le fonctionnent du nœud est suivant l'automate représenté sur *figure 4.1*. Afin de faciliter la présentation des *DSCA*, la *figure 4.2* servira d'exemple détaillé de l'exécution de *DSCA*. Dans la *figure 4.2*, une flèche en pointillé entre deux nœuds a et b, indique que le nœud a a choisi le nœud b en tant que parent, alors qu'une flèche continue indique que le lien (a, b) est planifiée.

Les lignes en pointillés représentent la connectivité des graphes, c.-à-d. la présence d'une liaison de communication entre une paire de nœuds. Le nombre à côté des flèches (pointillé ou solide) (a, b) représente le Slot de a.

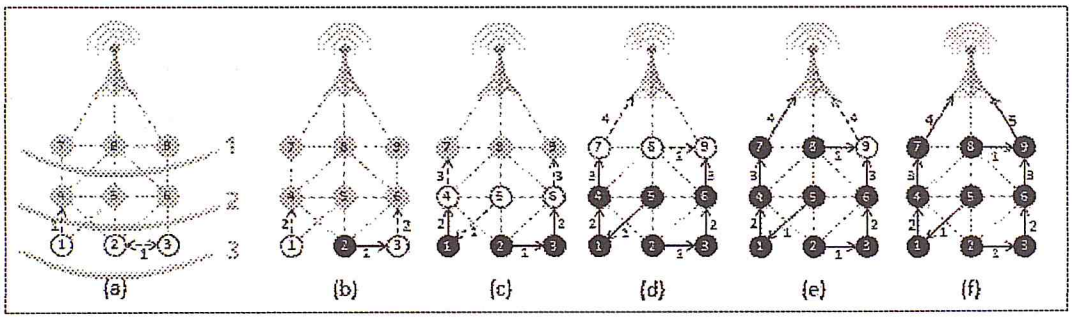

Figure 4.2 « Exemple d'exécution de DSCA »

2.2.1. L'état Not Ready (NRY):

Quand *DSCA* est lancé dans un nœud u, il initialiser:

(1) l'état de u à NRY.

(2) TS_u à zéro.

(3) $CandidateParents_u, \gamma_u, NeighborSC_u$ et $ParentSC_u$) à Ø.

Initialement, tous les nœuds sont dans l'état *NRY*, ce qui signifie qu'ils ne sont pas aptes à être planifié. Le passage à l'état planifié est exécuté niveau par niveau, en commençant à partir du niveau le plus bas vers le haut jusqu'à la station de base. Par exemple dans la *figure 4.2(a)* les nœuds n_1, n_2 et n_3 qui sont au niveau 3, doivent donc être programmée avant le nœud n_5 qui est au niveau 2. La planification des nœuds dans un ordre *ascendant* ne signifie pas que les données agrégées soient également acheminées sur un arbre de façon ascendante. Dans *DCSA*, quand un nœud de niveau L devient prêt (*Ready*), il choisit son parent à partir de $\gamma_u\,[L-1]$, $\gamma_u\,[L]$ et $\gamma_u\,[L+1]$. Par exemple, dans la figure 4.2 (d)

le nœud n_5 est au niveau 2 choisit son parent n_1 à partir du niveau 3. Chaque nœud u du niveau L, change d'état à W0, si tous ses voisins dans le niveau précédent sont planifiée c.-à-d. il doit avoir reçu les messages «*Scheduled* » à partir de tous les nœuds appartenant à $\gamma_u\,[L+1]$ (Automate figure 4.1: Transition NR-NR). Une fois que tous les messages planifiée (*Scheduled*) ont été reçus par u, celui-ci passe à l'état W0, puis diffuse le message *Wait* (figure 4.1: Transition NR-W0). Notez que, comme les nœuds dans le niveau inférieur du réseau n'ont pas $\gamma_u\,[L+1]$ alors ils sont automatiquement transformés à l'état W0 puis diffusent le message *Wait*.

2.2.2. L'état Wait0 (W0)

Chaque nœud u, appartenant au niveau L, reste dans l'état W0 et reçoit le message d'attente (*Wait*) de tous ses voisins v (tel que $v \in \gamma_u\,[L]$), se référant à la figure 4.1, transition W0-W0 , cette étape as été proposé afin de permettre au nœud u, de passer à l'état Prêt (*Ready*) si et seulement s'il n'a aucun des voisins au même niveau avec un ou plusieurs voisins non planifiés dans le niveau précédent. Une fois que tous les messages d'attente sont reçues par u, ce dernier se transforme en état *Ready* (*figure 4.1*: Transition W0-RY).

2.2.3. L'état Ready (RY)

2.2.3.1 Sélection du parent et du Slot

La construction de l'arbre d'agrégation ainsi que la planification de transmissions du nœud sont étroitement liées dans *DSCA*. Ainsi, quand un nœud u est prêt (*Ready*), il doit sélectionner simultanément son parent P_u et définir son Slot TS_u . Pour réduire la latence des données, TS_u doit être le plus petit slot t possible:

(1) plus grand que tous les Slots de ses enfants (cela assure la fraîcheur des données).

(2) satisfasse la condition suivante $ParentSC_u[t] = \emptyset \wedge CandidateParents_u[t] \neq \emptyset$ Par exemple, dans la *figure 4.2* (a), $ParentSC_1[1] = \emptyset$ et $CandidateParents_1[1] = \{n_4, n_5\}$ alors le Slot minimal qui peut être affectée au nœud n_1 est *1*. Considérant que, dans la *figure 4.2 (b)*, $CandidateParents_u[t] = \emptyset$, ce qui signifie que $NeighborSC_1 = \{n_4, n_5\}$, et si le noeud n_1 utilise ce Slot alors il ne peut pas sélectionner un parent plus tard afin d'éviter d'éventuelles collisions. Le nœud parent P_u est sélectionné parmi $CandidateParents_u[t]$ d'une manière a maximiser la réutilisation des Slots. Notez que $CandidateParents_u[t]$ peut comprendre des nœuds dans $\gamma_u\,[L-1]$, $\gamma_u\,[L]$ et $\gamma_u\,[L+$

1] . Dans DSCA, P_u est sélectionné parmi trois niveaux de manière à augmenter le nombre de candidats.

Si le nœud *u* est planifié au Slot *t* avec *w* comme parent, donc $ParentSC_v[t] \neq \emptyset$, $\forall v \in \overline{SC_w}$, Ainsi, ce dernier sera avisé quant à l'utilisation du Slot *t* lors de sa planification.

Si le nœud *u* choisit son parent *w* qui a le plus grand nombre de voisins non planifiés beaucoup de nœuds seront empêchés d'utiliser TS_u , ce qui limite la réutilisation des Slots. Pour cette raison, dans DSCA le parent P_u d'un noeud u est sélectionné parmi $CandidateParents_u[TS_u]$, tel que le nombre de nœuds qui seront empêchés d'utiliser TS_u ($|\overline{SC_{P_u}} - \{u\}|$) est minimisé. S'il y-a plus d'un nœud dans $CandidateParents_u[TS_u]$ qui respectent les conditions ci-dessus, le parent P_u est choisi en tant que celui qui a le plus petit Identifiant. Contrairement aux solutions précédentes, *DSCA* permet aux nœuds de sélectionner leurs parents à partir des nœuds déjà planifiées, c'est à dire, $P_u \cap SC_u \neq \emptyset$, un nœud u choisit son parent w à partir de $CandidateParents_u[TS_u] \cap SC_u$, Si $TS_u < TS_w$, Par conséquent, la fraîcheur des données est assurée et la création de cycles est empêchée.

Se référant à la *figure 4.2 (a)* le nœud n_2 doit choisir son parent dans le slot 1 à partir de $CandidateParents_2[1] = \{n_3, n_4, n_5, n_6\}$. Le nœud n_3 a deux (02) voisins non planifiés, alors que les nœuds n_4, n_5 *et* n_6 ont cinq voisins non planifiés alors quatre nœuds seraient empêchés d'utiliser le slot 1, alors qu'un seul nœud ($|\overline{SC_3} - \{n_2\}| = 1$) serait empêché si le nœud n_3 est sélectionné comme parent. Pour cette raison le nœud n_2 sélectionne le nœud n_3 en tant que parent. La *figure 4.2 (c)* montre un exemple pour la sélection de parent du nœud n_5 à un nœuds planifiées au Slot *1*, Ici, $\gamma_5 = \{n_1, n_2, n_4, n_6, n_8\}$, $ParentSC_5[1] = \emptyset$, $NeighborSC_5[1] = \{n_4, n_6\}$, $£_5[1] = \{n_2\}$ et $CandidateParents_5[1] = \gamma_5 - \{ParentSC_5[1] \cup NeighborSC_5[1] \cup £_5[1]\} = \{n_1, n_8\}$. Le nœud n_5 doit utiliser le Slot 1 (c'est à dire, $TS_5 = 1$), et sélectionner son parent P_5 a partir de $CandidateParents_5[1]$. Comme le nœud n_1 a le plus petit nombre de voisins non planifiés, (c'est à dire, $|\overline{SC_1} - \{n_5\}| = 0$) le nœud n_5 sélectionne le nœud n_1 en tant que parent.

2.2.3.2. Planification des nœuds en réseau :

L'étape de sélection du parent mentionné ci-dessus est un processus localisé au niveau du nœud dans le sens où un nœud sélectionne son parent et définit son Slot de telle sorte qu'aucune collision ne se produise avec des nœuds déjà planifiées. Cependant, des conflits peuvent se produire entre les nœuds prêts quand ils sont planifiés simultanément. Pour illustrer, considérons la *figure 4.2 (a)*. Sur la base des critères de sélection des parents indiqués ci-dessus, les nœuds n_1, n_2 *et* n_3 sélectionneront les nœuds n_4, n_3 *et* n_2 comme parents au slot 1, respectivement. Si les noeuds n_1, n_2 *et* n_3 sont planifiés simultanément, des collisions se produiront. En général, un conflit entre les nœuds *u* et *v* se produisent si l'une des conditions suivantes est vraie:

1) Une collision : Si $TS_u = TS_v$ et $(P_u \in \gamma_v \, ou \, P_v \in \gamma_u)$.
2) Violation de fraîcheur de données ou la création d'un cycle:

   Si un nœud prêt (*Ready*) sélectionne un autre comme parent, c'est à dire, $P_u = v \, ou \, P_v = u$.

Afin de résoudre la collision entre les deux nœuds *u* et *v*, *DSCA* planifie d'abord le nœud qui permet plus de réutilisation de Slot. Un nœud *u* est considéré comme ayant une meilleure configuration que le nœud *v*, Si celui-ci maximise la réutilisation de *Slots*.

Dans ce qui suit le meilleur concept de meilleures configurations entre deux nœuds conflictuels *u* et *v* est défini. Nœud *u* a une meilleure configuration qu'un nœud *v* si l'une des conditions suivantes est satisfaite:

1- le parent de *u* as un nombre de voisins non planifiés inferieur au parent de *v*, ce qui permet de réutiliser plus de *Slot*.
2- Si les deux parents ont le même nombre de voisins non planifiés mais que le nœud *u* a moins de nombre de voisins non planifiés.
3- Si les nœuds *u* et *v* (respectivement leurs parents) ont le même nombre de voisins non planifiés, mais *u* possède un *ID* inférieur.

La condition -1- permet une plus grande réutilisation des *Slots*, Alors que Condition -2- permettre d'augmenter le nombre de nœuds pouvant être sélectionnés en tant que parents dans ce *Slot*, c'est à dire, réduire la taille de $NeighborSC_w[TS_u]$ pour chaque nœud $w \in \gamma_u$.

La violation de la condition de fraîcheur de données ou la création de cycles sont réglées comme suit:

Si $P_u = v$ *ou* $P_v = u$ alors le nœud qui possède le plus petit *Slot*, disons u devrait avoir la meilleure configuration. Si un nœud v définit u en tant que parent et v est planifié en premier, le nœud u devrait changer de *Slot* afin qu'il soit plus grand que TS_v, de telles sortes que la fraîcheur de données soit assurée. Si $TS_u = TS_v$, le conflit sera résolu en utilisant la même approche.

Revenons à l'exemple illustré dans la *figure 4.2 (a)*, et considérons les trois paires possibles de nœuds n_1, n_2 *et* n_3.

Le nœud n_1 sélectionne le noeud n_4 en tant que parent, on suppose que les nœuds n_1 *et* n_2 sélectionnent des nœuds distincts. Par conséquent, si les nœuds n_1 *et* n_2 sont planifiés dans le même *Slot*, une collision affectera le nœud n_4. Si le noeud n_1 est planifié en premier, il empêchera les autres voisins non planifiée d'utiliser le Slot *1*, nous aurons $ParentSC[1]$ des quatre nœuds ($|\overline{SC_4} - \{n_1\}| = \{ n_2, n_5 n_7, n_8\}$=4) non vide.

L'utilisation du Slot *1* par le nœud n_2 rend le noeud n_3 bloqué ainsi il ne lui restera qu'un seul voisin non planifié à choisir. Le nœud n_2 a une meilleure configuration que le nœud n_1, il se verra donc alloué le *Slot 1* avant le nœud n_1.

La même analyse conclut que le nœud n_1 devrait être programmé avant le nœud n_3.

Donc, au nœud n_2 est alloué le slot 1 et aux nœuds n_1 et n_3 sera affectés le *Slot 2*, comme le montre la *figure 4.2 (b)*.

Dans ce qui suit, un échange de messages permettant de résoudre ce conflit d'une manière distribuée est décrit :

Chaque nœud u prêt (*Ready*) diffuse à tous les nœuds $CandidateParents_u[TS_u]$ un message *Parent_Selection_Request* dès qu'il sélectionne son parent P_u, et définit son *Slot* TS_u. Ce message contient les informations suivantes : *Node_Id* = u, *Parent_Id* = P_u , TS = TS_u , $\overline{SC_u}$ et $\overline{SC_{P_u}}$, Chaque nœud w dans $CandidateParents_u[TS_u]$ recevant ce message le sauvegardera pour une utilisation ultérieure. Quand un nœud w reçoit le message *Parent_Selection_Request* de tous ses nœuds voisins prêt (*Ready*), il vérifie s'il n'est pas sélectionné en tant que parent pour chaque nœud prêt. Si oui, il envoie le message *Parent_Selection_Response* contenant la réponse positive à tous les nœuds prêts.

Dans le cas contraire, la réponse du nœud w dépend du *Slot* où il est sélectionné comme parent. Le nœud w répond positivement aux nœuds prêts qui reprennent des Slots différents de ceux dans lesquels w est choisi en tant que parent.

Pour les nœuds conflictuels qui ont sélectionné le nœud w Comme parent ou ont pris un

Slot où le nœud *w* est sélectionné comme parent, ce dernier envoie une réponse positive seulement au nœud qui a la meilleure configuration, et une réponse négative aux autres.
Chaque nœud *u* prêt attend jusqu'à ce qu'il reçoive le message *Parent_Selection_Response* de tous les nœuds dans $CandidateParents_u[TS_u]$.
Si le nœud *u* reçoit une réponse négative d'un nœud dans $CandidateParents_u[TS_u]$ il se tourne vers l'état $Wait_1$.
Sinon, il change son état a *Scheduled* (planifiée) puis diffuse un message Scheduled contenant les informations suivantes: Node_Id = u, Parent_Id = P_u, TS = TS_u.
Chaque nœud $w \in \gamma_u$ recevant un message *Scheduled* à partir du noeud *u* effectue les tâches suivantes:

1- Si *w* est le parent de *u* :
   a. *w* met à jour son Slot maximum TC_w, $TC_w = Max(TC_w, TS_u)$.
   b. *w* diffuse un message *Forbidden* contenant:
      $Node\_Id = w$, TS = TS_u, $isParent$ = True.
2- Si *w* n'est pas le parent de *u*, il diffuse un message *Forbidden*, comprenant:
   $Node\_Id = w$, TS = TS_u, $isParent = False$.

Chaque nœud *v* qui reçoit un message *Forbidden* « $Node\_Id = w$, $TS = t$, $isParent$ » à partir de *w*, vérifie $isParent$ joint au le message.
Si $isParent = True$, le nœud *v* met-à-jour $ParentSC_v[t]$ comme suit: $ParentSC_v[t] = ParentSC_v[t] \cup \{w\}$.
Si non, le nœud *v* met-à-jour $NeighborSC_v[t]$ comme suit: $NeighborSC_v[t] = NeighborSC_v[t] \cup \{w\}$.

2.2.4. L'état Wait$_1$ (W1)

Quand un nœud *u* prêt (*Ready*) reçoit une réponse négative de l'un des nœuds dans $CandidateParents_u[TS_u]$ il change son état à $Wait_1$. Un nœud dans l'état $Wait_1$ reste dans cet état tant que la condition n'est pas satisfaite (c'est à dire, le noeud concurrent qui a une meilleure configuration soit planifié). Dans ce cas, le nœud u transforme son état a prêt (*Ready*) (Automate 4.1: Transition W1-RY). Puis il définit son Slot et sélectionne son parent. Comme le montre la *figure 4.2(b)*, les nœuds n_1 et n_3 changer leur état à prêt (*Ready*) lorsque nœud n_2 est planifiée.

3. Framework DADIDA

3.1. Présentation du Framework

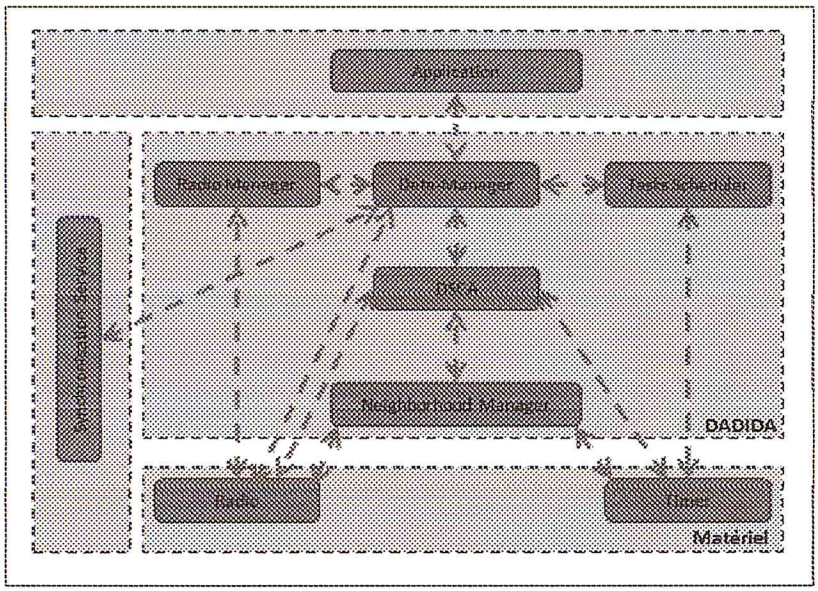

Figure 4.3 « Framework DADIDA »

L'exécution de *DSCA* s'effectue aussitôt que les solutions précédentes sont terminées. La structure en arbre sera construite et les nœuds dans le réseau planifié, tel que à chaque nœud est assigné un *Slot* afin d'envoyer ses données sans collision, la phase de collecte de données n'as pas encore été mentionné. Plusieurs questions peuvent être posées ici, comment les nœuds peuvent exploiter l'arbre crée ainsi que les *Slots* pour rassembler les informations? Comment les *Slots* sont distribuées dans la *Frame* de données? Quand et comment les nœuds en réseau seront synchronisés? Afin de répondre à ces questions nous avons implémenté un Framework appelé *DADIDA* qui encapsule le module *DSCA*, la *figure 4.3* représente l'architecture de *DADIDA*, ainsi que son interconnexion avec différent modules de différentes couches, comme le décrit la figure, le module principal de *DADIDA* est *Data-Manager*. *DSCA*, *Neighborhood-Manager*, et *Task-Scheduler* seront décrits par la suite.

Dans le but d'optimiser la consommation d'énergie ainsi que la latence, *DADIDA* as été conçu en utilisant une approche *Cross-Layer* (ou inter-couche), *DADIDA* intègre la couche *MAC*. La construction de topologies réseau ainsi que le routage. *DADIDA* est interconnecté avec trois composants, le niveau application, le niveau matériel (hardware) et le service de synchronisation. *DADIDA* offre à la couche application une interface afin d'envoyer et de recevoir des données, lorsqu'un paquet est reçu d'un nœud fils, un événement est déclenché dans la couche application qui sauvegarde l'ensemble des données pour une utilisation ultérieure, ou bien l'agrégation est immédiatement effectuée et le résultat partiel de l'agrégation est enregistrée. Lorsque les paquets de données sont reçu de la part de tous les nœuds fils un événement est déclenché au niveau de l'application, cette dernière effectue le processus d'agrégation et dès lors envoi les données agrégées via l'interface d'envoi vers le nœud parent, le module *MAC* de *DADIDA* est basée *TDMA*, ce qui rend l'énergie ainsi que la latence efficace. Un protocole qui synchronise les nœuds en réseau est utilisé à cet effet.

3.1.1. Radio-Manager : Ce module fournit une interface afin d'allumer ou d'éteindre l'unité de transmission, ce qui permet de minimiser la consommation d'énergie. Dans la *figure 4.3* cela est présenté en tant que connexion entre l'unité de communication et *Radio-Manager*. Nous avons également modifié l'implémentation de l'interface d'envoi des messages de la carte radio *CC2420* enlevant ainsi la partie *clear chanel Assessment* (CCA) un module qui randomisait l'envoi des messages. Ce module a était initialement créé afin d'éviter les collisions lors de l'envoi en choisissant aléatoirement un temps pour l'envoyer d'un paquet dans un intervalle de 14 milliseconde, cette modification as été faite afin de pouvoir baisser les délais de transmission d'un paquet jusqu'à 2 milliseconde. Le module MAC de DADIDA qui est basée TDMA ce qui lui offre l'avantage de ne pas se soucier d'éventuelles collisions et donc plus besoin de CCA.

3.1.2. Task-Scheduler: Ce module donne la possibilité à chaque nœud de planifier l'exécution d'une tache en temps voulu. Chaque tâche est exécutée à son heure dans un modèle non préemptible, tel qu'il n'y est aucune priorité entre les taches, une horloge hardware est utilisé afin de déclencher l'exécution des taches.

3.1.3. Neighborhood-Manager : Ce composant donne la possibilité à chaque nœud de découvrir son voisinage, il organise également les nœuds en niveau, de telle sorte que les nœuds ayant la même distance en termes de saut (plus court chemin respectifs jusqu'à la

station de base) soient au même niveau. Ce résultat est obtenu par la diffusion d'un message de découverte à partir de la station de base en utilisant une variable *hop-count* (compteur de saut) qui est initié à 0. Lors de la première réception de ce message, le nœud définit son niveau à partir du hop-count reçu plus un (+1). Le nœud continue à mettre à jour son niveau lors de la réception d'un message de découverte contenant un *hop-count* plus petit, Ce dernier est retransmis après l'incrémentation du *hop-count* par le récepteur, tandis que les messages de découverte avec un hop-count plus grands sont tout simplement ignorés. Ce module est également chargé de tester la qualité des liens afin d'éliminer les liens trop faibles qui risquent de perdre beaucoup de paquets, ainsi que les liens unidirectionnels.

3.1.4. DSCA: comme nous l'avons vu dans les sections précédentes, ce module est chargé de la construction de l'arbre de topologie et de la planification des nœuds du réseau en affectant les *Slots* respectifs aux nœuds. L'arbre construit et les slots sont utilisés pour transmettre les données agrégées à la station de base tout en empêchant les collisions de massages en réduisant la latence. Contrairement aux solutions précédentes, la construction de l'arbre et l'affectation des Slots dans DSCA sont exécutés simultanément afin de maximiser la réutilisation des Slots et réduire le temps de latence.

3.3.5. Data-Manager: Ce module assure la collecte des données à partir des nœuds de réseau vers la station de base. Il est également le module principal assurent la communication entre les différents modules, il est ainsi responsable de la communication avec la couche d'application. Ce module fournit à la couche application une interface pour envoyer et recevoir les données. *DADIDA* comprend essentiellement deux étapes, qui sont périodiquement exécutés l'une après l'autre en utilisant le planificateur de taches *Task-Scheduler*. La première est la construction de l'arbre et la planification des nœuds, qui est exécutée par *DSCA*. La deuxième étape est la collecte de données. La deuxième étape doit être beaucoup plus longue que la première pour la conservation de l'énergie et la réduction de la latence. Au cours de l'étape de collecte de données, le service de synchronisation de temps est appelé périodiquement afin de synchroniser les nœuds en réseau. En utilisant Radio-Manager, *Data-Manager* éteint l'émetteur-récepteur radio quand il n'y-a pas de paquets à transmettre ou à recevoir, afin de minimiser au maximum la consommation d'énergie.

3.2. DADIDA : Etapes et descriptions :

Il convient de souligner deux phases importantes:

(1) La collecte de données.

(2) La construction de l'arbre et la planification des nœuds.

Les deux phases sont exécutées périodiquement l'une après l'autre. Généralement, la première phase est beaucoup plus longue que la seconde, cela assure l'optimisation de la consommation d'énergie. Comme mentionné ci-dessus, *DSCA* est le responsable de la construction de l'arbre et de la planification des nœuds. L'objectif de la construction de l'arbre de *DSCA* et de la programmation des nœuds est la minimalisation du nombre (*Slots*), ce qui réduit inévitablement la latence. Dans la phase de collecte, les données sont périodiquement recueillies et transmises à partir des nœuds à la station de base, par encapsulation dans de nombreuse Frames. Le module *Data-Manager* est le responsable de cette phase. Dans chaque nœud, *DSCA* informe *Data-Manager* sur son parent, ses fils, ainsi que les Slots pour transmission ou réception de données.

Contrairement aux protocoles *MAC* précédentes, le but d'un protocole d'agrégation de données est d'assurer la répartition des Slots et permet ainsi la collecte de données à partir de tous les nœuds du réseau en une seule *Frame*. Pour cette raison les Slots attribué à un nœud parent doit être ultérieur à tous les Slots attribués à ses nœuds fils. Habituellement, la durée des Slots est prédéfinie, donc l'optimisation du *Duty_Cycle*[1], la consommation d'énergie et la latence sont proportionnelles à la réduction du nombre de Slots. Un écart est utilisé entre deux Frames successives, dans lequel les nœuds éteignent leur émetteur-récepteur. Ceci est connu comme la période de sommeil (ou *Sleep_Period*), dont la durée varie en fonction de l'application. Cette période peut être ignorée lorsque le taux de données de télédétection est très élevé, notamment dans les applications en temps réel, par exemple, applications de surveillance et de sécurité. Néanmoins, dans de nombreuses applications de surveillance, telles que l'irrigation automatique, où la différence entre deux captage successifs est mineure, et le taux d'échantillonnage des données est généralement de faible importance, l'extension des périodes de sommeil pour des raisons de conservation de l'énergie est justifiée.

Deux types de Frames sont utilisés au cours de la phase de collecte de données:

(1) La Frame ordinaire.

(2) Frame avec synchronisation.

Dans une Frame ordinaire, les données sont reçues en provenance des nœuds enfants, puis regroupées et transmises au nœud parent pendant le slot spécifié. Chaque nœud allume sa radio uniquement pendant les Slots de transmission ou la réception de données. La Frame avec synchronisation comprend deux étapes successives. Tout d'abord, les nœuds du réseau sont synchronisés via le protocole *FTSP* à partir de la station de base. Ensuite, les données sont agrégées et acheminées à partir des nœuds du réseau vers la station de base en utilisant une Frame ordinaire.

Soit Th_A, le nombre de Frame avec synchronisation qui sont nécessaire pendant les processus de collecte de données avant de créer l'arbre d'agrégation et la planification des nœuds du réseau avec *DSCA*. Dans le schéma représenté dans la *figure 4.4*, (*A*) compte le nombre de Frame afin de contrôler le nombre de Processus de collecte de données avant la mise à jour de la structure de l'arbre. Comme le montre le schéma, un test sur la valeur de *A* est périodiquement effectué. Si la valeur de *A* est égal à Th_A alors le module *DSCA* est appelé pour reconstruire l'arbre d'agrégation et à réorganiser les nœuds du réseau. Dans le cas contraire, la phase de collecte de données est démarré en ré-exécutant le module *Data-Manager*.

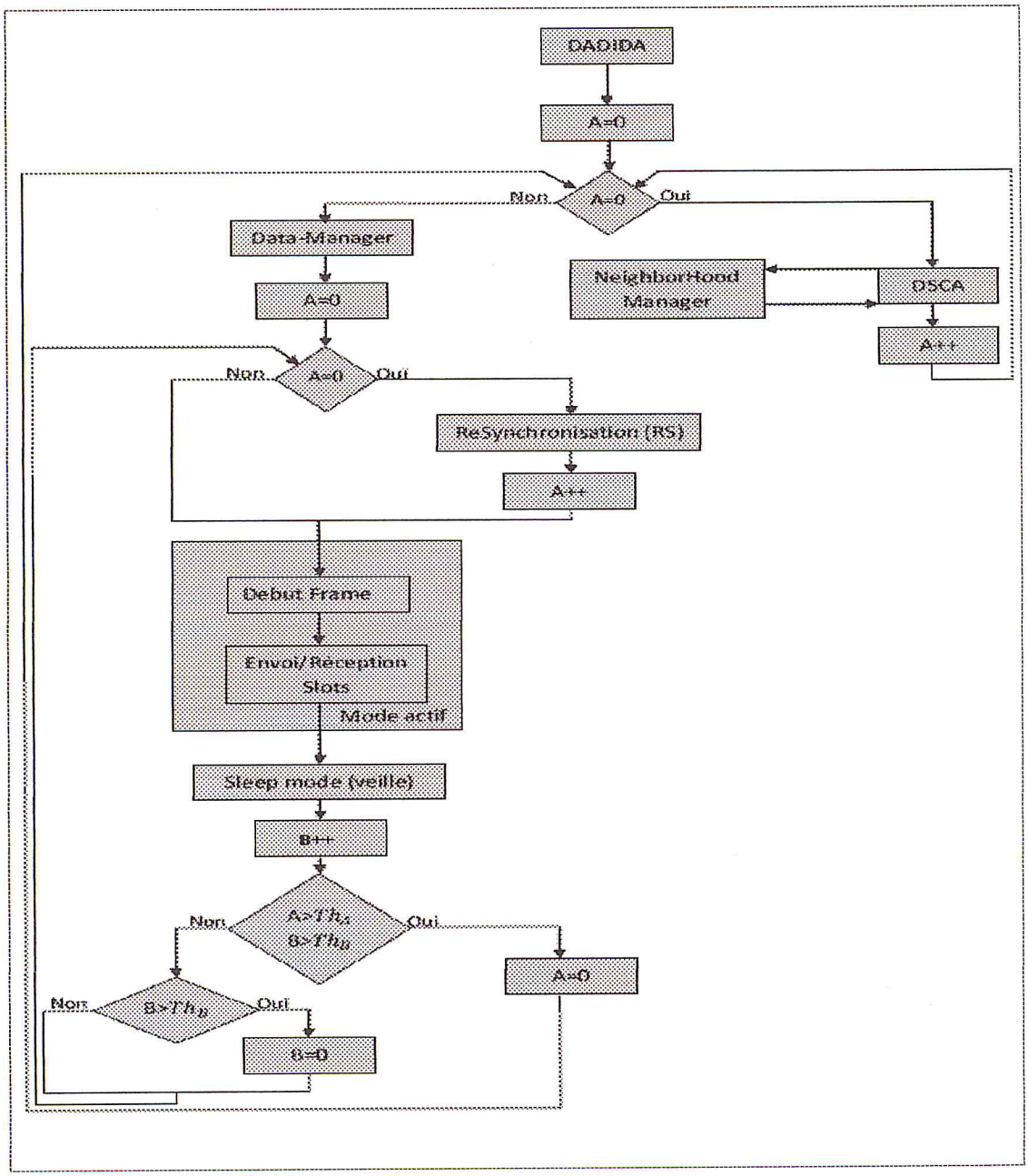

Figure 4.4 « Etapes de l'exécution de DADIDA »

DSCA commence une phase de découverte en appelant *Neighborhood-Manager* Comme il est expliqué à la section précédente, le processus de découverte des voisins commence à la station de base et se propage à tous les nœuds du réseau. A la fin de cette phase, chaque nœud du réseau découvre ses voisins, et les nœuds du réseau seront organisés en niveaux. A la fin de l'exécution de *Neighborhood-Manager*, *DSCA* est appelé pour construire l'arbre d'agrégation et de planifier les nœuds du réseau en fonction de

l'automate représenté sur la *figure 4.1*. Comme vu dans la section précédente, *DSCA* est terminé lorsque l'état de tous les nœuds du réseau est "*Scheduled*" (ou planifié). Quand l'état d'un nœud devient planifié, il attend le signal de la station de base pour démarrer le module *Data-Manager*.

Lorsque les voisins les plus proches de la station de base sont planifiés, la station de base diffuse à travers le réseau le message de Démarrage de *Data-Manager* qui contient le dernier (ou plus grand) *Slot* noté *l*. Comme représenté dans la *figure 4.2*, lorsque la station de base reçoit un message "*Scheduled*" à partir du nœud 9, il diffuse sur le réseau le message de démarrage de *Data-Manager* avec le *Slot* égal à 5 (*Slot* maximum dans la *figure 4.2*). En supposant qu'un *Slot* de durée *D* est déjà fixé, la durée de la Frame et pour chaque nœud calculé comme suit:

$$Durée\_Frame = D \times l \quad .$$

Chaque nœud recevant ces messages calcule la durée de la *Frame*, puis lance le module *Data-Manager* afin de récolter les données.

Soit, Th_B le nombre de *Frames* ordinaires avant que les nœuds ne soient synchronisés, c'est à dire, le nombre de *Frames* ordinaires entre deux *Frames* avec synchronisation et *B* compte le nombre de Frame ordinaire avant d'effectuer la synchronisation. Il y-a un compromis entre la consommation d'énergie et le débit du réseau. La synchronisation rapide des nœuds a un impact positif sur le débit du réseau mais augmente la consommation d'énergie. Pour cette raison, Th_B doit être soigneusement choisi afin de créer un équilibre entre la consommation d'énergie et le débit du réseau. Pour cela le module *DSCA* est périodiquement appelé après $Th_A \times Th_B$ Frames de récolte de données.

Dans le module *Data-Manager*, chaque nœud obtient des Slots pour transmettre ou recevoir des données à partir du module *DSCA*. C'est en utilisant cette information que *Task-Scheduler* planifie les tâches de réception ou d'émission pour chaque nœud au début de chaque Frame, ainsi que la planification de l'entrée en mode Sommeil. Notez que chaque nœud met en marche sa radio seulement pendant ses slots actifs (*Frame*). Afin d'assurer la fiabilité du réseau, un nœud retransmet le même paquet pendant tout le *Slot* jusqu'à ce qu'il soit correctement reçu ou que la durée du slot se soit écoulée. Par ailleurs, au début du mode veille, la Frame suivante est également planifié grâce au *Task-Scheduler*.

Figure 4.5 « Périodes d'activité d'un nœud »

La *figure 4.10* représente la phase de collecte des données au niveau du nœud 1 lorsque les nœuds sont prévus et l'arbre est construit selon la *figure 4.2*. Dans la *figure 4.10*, le nœud 1 envoie ses données à son parent 4, au cours du Slot 2, et il reçoit les données à partir de son enfant 5 autour du Slot 1. Dans la *figure 4.10* la couleur blanche représente le moment où le nœud 1 éteint sa radio.

4. Conclusion

La phase de planification est la phase la plus importante. L'optimisation de celle-ci en minimisant le nombre de slots et en optimisant les chemins aura un impact considérable sur la latence et ainsi sur toute la procédure de collecte et d'agrégation des données Dans ce chapitre nous avons vu les détails de l'implémentation de *DSCA* ainsi que le fonctionnement du Framework dans lequel il est encapsulé, tel que chaque module est prévu afin de répondre a un besoin précis. Dans le prochain chapitre nous entameront la simulation de *DSCA* afin de faire une étude comparative avec les autres protocoles MAC traitant la planification des données agrégées. Nous verrons également les simulations de du Framework *DADIDA* et évaluerons ses performances sur un simulateur ainsi que sur une plateforme réelle.

CHAPITRE -V-

Simulations et résultats expérimentaux

1. Introduction

Le déploiement d'un réseau de capteurs nécessite impérativement une phase de test avant sa mise en place, et ce, afin de s'assurer de la cohérence du réseau, tant au niveau fonctionnel que matériel. Pour cela, une solution peu coûteuse consiste en la simulation qui sert d'un premier pas afin de mettre en œuvre une application. La simulation des réseaux de capteurs consiste principalement en la reproduction du comportement et du fonctionnement des nœuds capteurs dans un environnement de simulation. La solution proposée a été préalablement évalué par simulation et par rapport aux algorithmes dans l'état-de-l' art. Les résultats montrent que l'algorithme proposé *DSCA* (Distributed Simultaneous tree Construction and data Aggregation scheduling) surpasse six algorithmes compétitifs en termes de latence et de durée de vie du réseau. En plus de l'analyse de simulation, *DSCA* est l'unique solution ayant été implémenté sur capteurs réels en utilisant plateforme TinyOS 2.x.

Dans la première partie de ce chapitre nous passeront en revue les différents outils matériels et logiciels que nous avons dû utiliser afin de mener à bien la mise en œuvre ainsi que les test de *DSCA* et le Framework *DADIDA* qui as permis de tester réellement cette solution. Dans la seconde partie nous présenterons et analyserons les performances de DSCA comparée aux autres algorithmes de planification de l'agrégation de l'état de l'art grâce aux simulations que nous avons effectuées. Enfin dans la 3eme partie nous décrirons notre implémentation de *DADIDA* sur une plateforme réelle de réseaux de capteurs (Test-Bed) au *Centre de recherche sur l'information scientifique et technique CERIST* [40] et les résultats obtenu lors des tests réels

2. Les outils utilisés

2.1.1. Le Matériel

Durant nos expérimentations, nous avons utilisé la plate-forme *MicaZ (Figure 5.1)* produite par *Crosbow Technology*. Elle comporte un microcontrôleur *ATMEGA* 128L (8-Bit) avec 4 KB de RAM et 128 KB FLASH. Le nœud capteur inclut une radio sans fil de type *CHIPCON CC2420* possédant un débit de 250 Kbits/s et une portée qui varie entre 1 et 30 mètres. Cette radio fonctionne sous la norme IEEE 802.15.4, qui représente la norme la plus utilisée dans le domaine des *RCSFs*.

Notre implémentation de DSCA a été réalisée en utilisant le système d'exploitation **TinyOS** et le langage de programmation **nesC** qui représentent la solution *de factory* pour ce genre de systèmes embarqués.

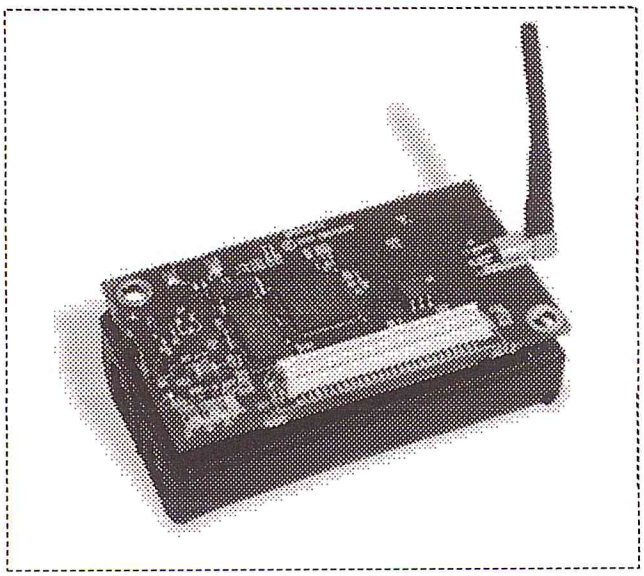

Figure 5.1 « Capteur MicaZ »

2.1.2. TinyOS

TinyOS [6] est un système d'exploitation open source conçu pour le développement des applications embarquées. Le caractère open source permet à ce système d'être régulièrement enrichi par une multitude d'utilisateurs. 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 [44], 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.

2.1.3. NesC

TinyOS permet de faciliter le développement d'applications complexes en utilisant le concept de composants et interfaces. Cela est possible grâce à son langage *NesC* [45 - 47] qui est un langage orienté composant syntaxiquement proche du C. Une application NesC consiste en un ou plusieurs composants assemblés pour former un exécutable. Un composant offre et utilise des interfaces. L'interface est le seul point d'entrée au composant, elle définit un ensemble de commandes implémentées dans le composant qui l'offre, et un autre ensemble d'évènements implémentés dans le composant qui l'utilise.

Un même composant peut fournir ou utiliser plusieurs interfaces différentes ou plusieurs instances d'une même interface. Il existe deux types de composants : le module et la configuration. Les configurations sont utilisées pour faire le câblage des composants en reliant les interfaces utilisées par des composants aux interfaces fournis par d'autres composants. Les modules contenant le code de l'application.

2.1.4. Python

Python [48] est un langage de programmation interprété et libre, créé par Guido van Rossum en 1990. Il s'agit d'un langage multi-paradigme, ce qui signifie qu'au lieu d'imposer aux développeurs un type de programmation précis, il leur laisse adopter celui de leur choix. Il permet ainsi la programmation orientée objet, impérative structurée ou encore fonctionnelle.

Python gagne partout du terrain car il facilite le développement d'applications, de scripts et de prototypes.

On utilise ce langage pour dessiner les graphes de simulation et ainsi que pour la génération des topologies de réseau. Nous utilisons python également dans la vérification automatique des *log*s retournés pas les nœuds capteurs.

Afin de faciliter l'opération de génération de topologies de simulations et varier les résultats, nous avons utilisé le package NetworkX [49] pour la création, la manipulation de structure de réseaux complexes.

2.1.5. Gnuplot

Gnuplot [50] est un utilitaire graphique portable utilisable en lignes de commande pour Linux, OS / 2, MS Windows, OSX, VMS, et de nombreuses autres plates-formes. Le code source est protégé par un copyright, mais distribué gracieusement. A l'origine il a été créé pour permettre aux scientifiques et aux étudiants de visualiser les fonctions et les données mathématiques de manière interactive, mais a grandi et peut fonctionner avec de nombreuses applications non-interactives tels que des scripts web. Il est également utilisé comme un moteur de traçage par des applications tierces comme Octave. Gnuplot a été créé et est toujours en cours de d'évolution depuis 1986.

2.1.6. Le simulateur TOSSIM

TOSSIM [51] est un simulateur d'évènement discret permettant de tester une application TinyOS sans utiliser des capteurs réels. Il exécute un code développé avec le langage NesC et permet de simuler un réseau de capteurs en exécutant le code.

Nous avons utilisé cet outil durant les premières phases de notre projet afin de déboguer, tester et analyser nos algorithmes dans un environnement contrôlable. Il nous a été très utile car le débogage d'un algorithme distribué sur une plateforme matérielle réelle est difficile. Cette difficulté est due à la simplicité du matériel qui ne permet pas une interactivité facile lors de l'exécution du code, contrairement aux PC conventionnels qui possède des IDE avancés permettant une exécution pas à pas et un affichage aisé des messages d'erreurs sur écran.

2.1.7. L'émulateur Avrora

Avrora [52] est un émulateur des RCSFs, écrit en java. Il réalise une simulation complexe du microcontrôleur, des périphériques et de la communication radio. Ce simulateur offre plusieurs outils, par exemple l'analyse énergétique qui permet d'étudier la consommation d'énergie au niveau des nœuds-capteurs, ainsi que la durée de vie possible de la batterie utilisée. Cependant, Avrora n'implémente pas certains composants principaux comme un gestionnaire d'horloge. Ceci le rend incapable de modéliser une dérive d'horloge entre les nœuds du réseau. A noter que ce simulateur est spécifique aux plateformes MICA2 et MicaZ. Dans notre cas il était la dernière étape par laquelle on a dû passer avant de terminer les simulations et passer aux capteurs réels puisque ses résultats sont quasi identiques au résultat obtenu dans une machine réelle.

3. Résultats de simulation de DSCA

Dans cette partie, DSCA est comparée à NCA, SDA, ACS, DAS, CIAS, et SAS que nous avons vue dans le Chapitre -III-. Les algorithmes sont évalués en fonction des indicateurs suivants:

- **Temps de latence:** est défini comme le temps nécessaire à la station de base pour recevoir les données agrégées de tous les nœuds de capteurs.

- **Répartition des nœuds parents entre niveaux:** Cette métrique compte le pourcentage des parents à partir $n_{L+1}(u)$,$n_L(u)$ et $n_{L-1}(u)$ sélectionnés par un nœud u appartenant au niveau L.

- **Pourcentage de Slot explorées :** Cette métrique compte le pourcentage de nœuds qui ont sélectionnés leur parent à partir des nœuds planifiées (voir la *figure 4.2 (c)*).

- **Le nombre maximum d'enfants par nœud:** Cette mesure reflète la durée de vie du réseau, qui est défini comme la durée à partir du lancement du capteur jusqu'au moment où le premier nœud capteur n'as plus d'énergie [53]. Dans l'application d'agrégation de données, un nœud allume sa radio seulement quand il doit transmettre des données à son parent ou quand il doit recevoir les données de ses enfants. Par conséquent, la durée de vie du réseau tel que défini ci-dessus est une fonction

décroissante du nombre maximum d'enfants que n'importe quel nœud peut avoir dans le réseau.

- **Messages générée:** est défini comme étant le nombre de message produit lors de l'exécution de l'algorithme réparti.

La plupart des solutions de planification agrégation de données proposées dans la littérature sont centralisées, ce qui rend l'utilisation de simulateurs de réseaux tels que TOSSIM [51] et Avrora [52] impraticable. Par conséquent, pour évaluer la performance de ces algorithmes, nous avons développé un simulateur en utilisant le langage Python et un package additionnel sur la théorie des graphes appelé Networkx [49]. Dans les résultats de la simulation, chaque point tracé représente la moyenne de 35 exécutions. Les plots sont présentés avec un intervalle de confiance de 95%. Dans notre simulation, N nœuds sont déployés de manière aléatoire sur une surface carrée de longueur D selon une distribution uniforme. Nous générons une topologie de réseau pour chaque N et densité de réseau. $\Psi = \frac{\pi * r^2 * N}{D^2}$, où r est le rayon de transmission des nœuds. Nous considérons que r est fixe pour tous les nœuds, et donc l'évaluation d'algorithmes est réalisé en faisant varier le nombre de nœuds et de la densité du réseau. Nous menons deux types d'exécutions:

(1) Nous faisons varier N en fixant Ψ à 50

(2) On fait varier Ψ et fixant N à 300.

3.1. Temps de latence

La *figure 5.2* montre la latence de temps en fonction de N et de Ψ. La première observation qu'on peut tirer de la figure est que *DSCA* surpasse les autres algorithmes d'agrégation de données. Le temps de latence dépend de trois paramètres:

- La détente (relaxation) dans le choix de couches pour la sélection de parent.
- La taille de la liste des parents candidats.
- Le mécanisme de réutilisation de slots.

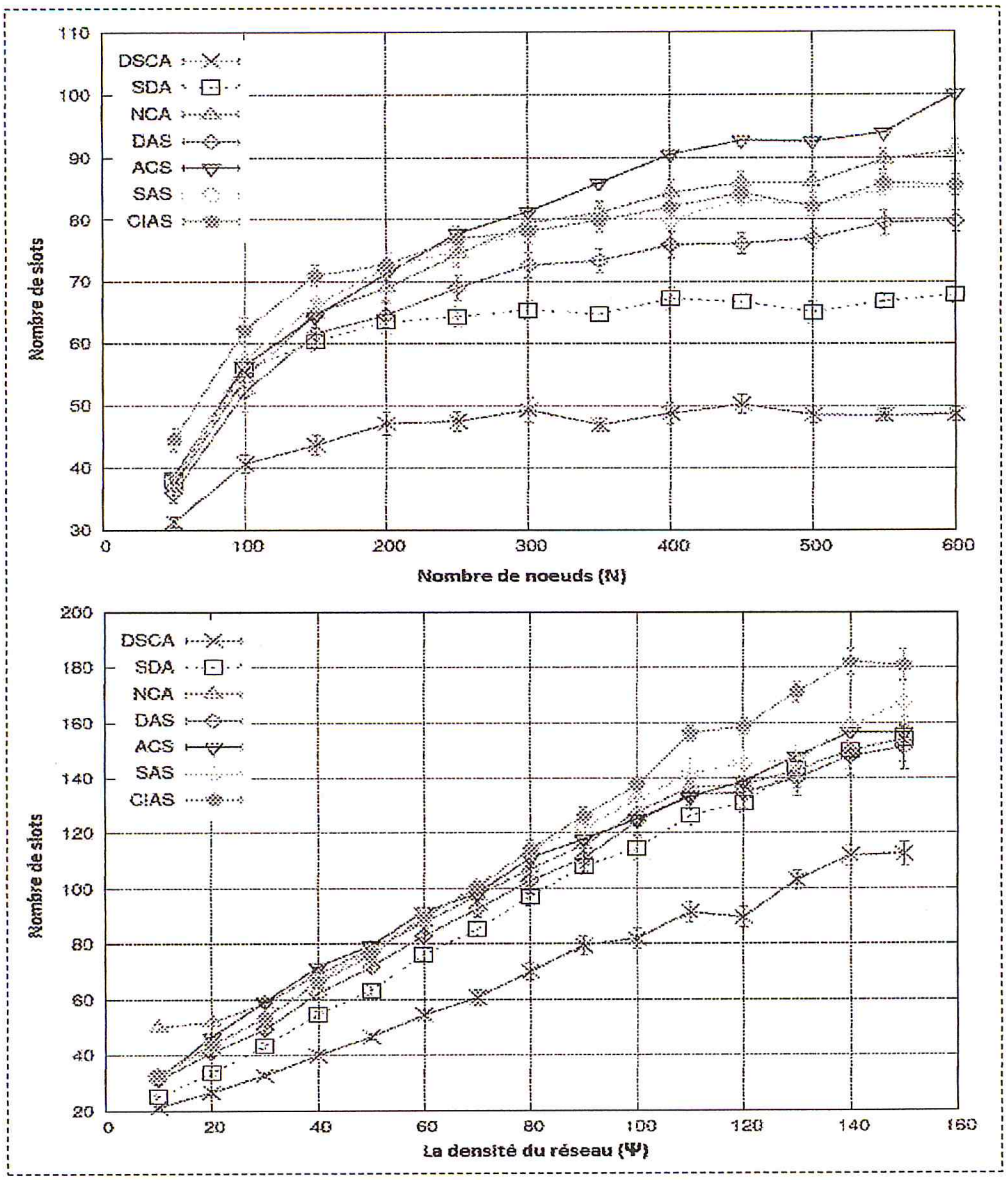

Figure 5.2 « Temps de latence »

Contrairement à la plupart des solutions existantes, un nœud dans *ACS* choisit son parent seulement à partir du niveau supérieur. Pour cette raison, ACS a la plus forte latence. Le mécanisme de réutilisation de slots dans *NCA* et *CIAS* ne conduit pas à des valeurs optimales. En *NCA*, les feuilles sont les premier à être planifiée, puis le reste des nœuds est planifié niveau par niveau commençant par le nœud avec le plus grand Slot, ce qui conduit à une latence élevée. Avec *CIAS*, après l'agrégation des données par le centre topologie (*topology center*), ce dernier doit renvoyer la valeur agrégée à la station de base en utilisant le plus court chemin. *SDA* a de meilleures performances en comparaison aux

autres solutions. Un nœud dans *SDA* peut choisir son parent à partir du niveau précédent. *DSCA* emploie la méthode d'exécution simultanée et critères de sélection de niveau le plus relaxée afin de choisir les nœuds parents parmi tous les algorithmes d'agrégation présentée. En outre, les nœuds dans *DSCA* sélectionnent les parents d'une manière qui maximise la réutilisation des Slot. Comme représenté dans les *figures 5.3*, entre 65% et 80% des parents dans *DSCA* sont sélectionnés à partir leur précédent niveau ou du même niveau. Entre 25% et 55% des parents dans *DSCA* sont choisis parmi les nœuds déjà planifiés, ce qui signifie que les enfants ont réussi la réutilisation des Slot existants, et de ce fait le temps de latence sera très faible.

3.2. Répartition des nœuds parents entre niveaux

La *figure 5.3* montre le pourcentage de parents sélectionnés dans le suivant, le même niveau ainsi que le niveau précédent, respectivement en fonction de N et Ψ. Tous les nœuds parents dans ACS sont choisis dans le niveau suivant. Dans *NCA*, *DAS*, *SAS*, et *CIAS*, entre 45% à 55% des nœuds sélectionnent leurs parents à partir du même niveau, tandis que les autres choisissent leur parent seulement à partir du niveau supérieur. Nous avons constaté que dans *SDA* seulement 2,5% des parents ont été sélectionné parmi le niveau précédent, bien que ce critère de sélection n'ait pas été mentionné dans [4] et les auteurs n'avaient pas l'intention de l'utiliser. Cela est dû au fait que l'arbre d'agrégation dans *SDA* est construite après la planification, et donc certains parents peut être sélectionné à partir leur niveau précédent. Avec *DSCA*, entre 65% et 80% des parents en moyenne sont choisis parmi les niveaux précédents et le même niveau, ce qui signifie que *DSCA* peut offrir une réutilisation des Slots, et qu'un temps de latence faible peut être atteint. Dans certains cas, jusqu'à 25% des nœuds réussissent à sélectionner leurs parents à partir niveau précédent.

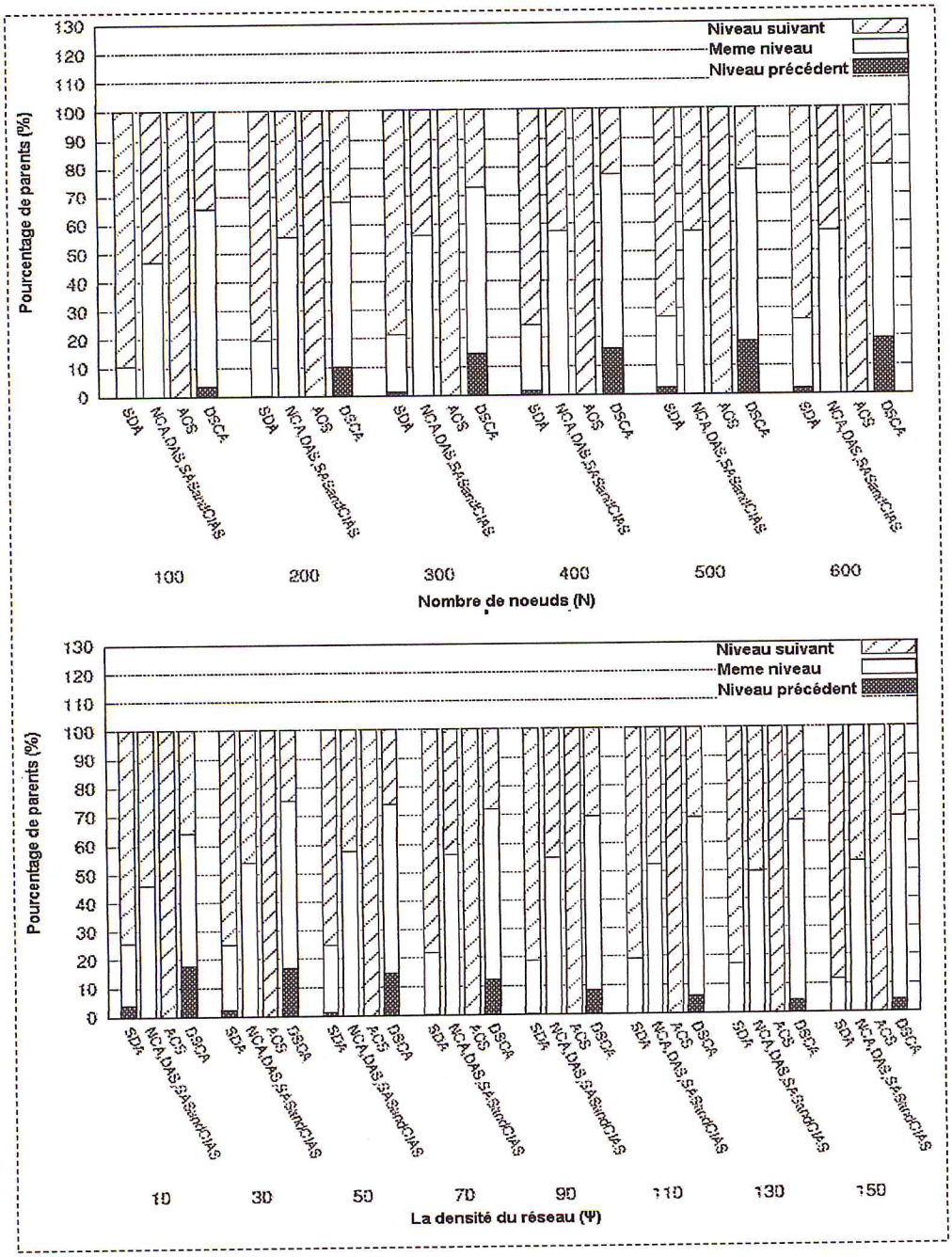

Figure 5.3 « Répartition des nœuds parents entre niveaux »

2.3. Pourcentage de Slot explorées

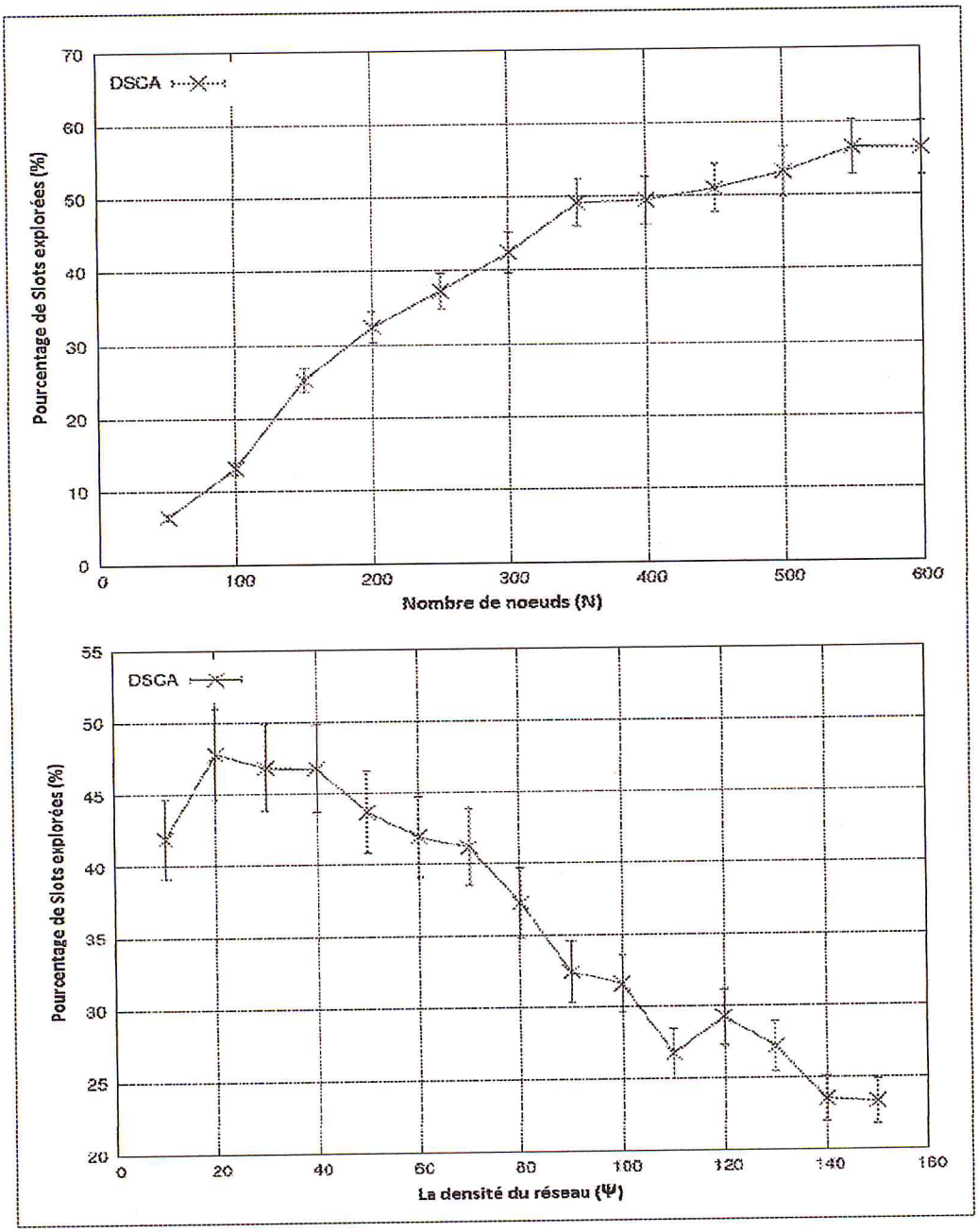

Figure 5.4 « Pourcentage de Slot explorées »

La *figure 5.4* montre le pourcentage de nœuds qui sélectionnent leurs parents à partir des nœuds déjà planifiées en fonction de N et Ψ . Dans *DSCA* plus de 50% des nœuds, leur parents sont choisis à partir des nœuds déjà planifiés. Ainsi, plus de 50% de nœuds arrivent à explorer des Slots existants.

3.4. Le nombre maximum d'enfants par nœud

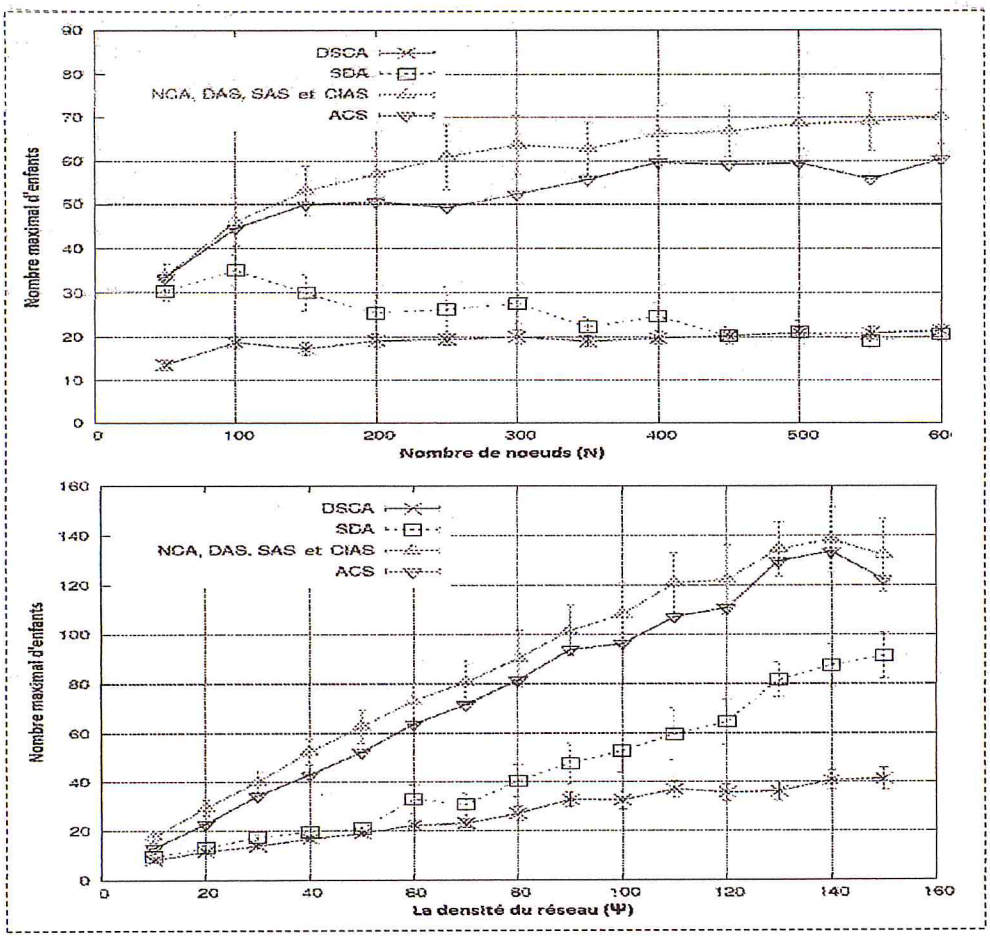

Figure 5.5 « Le nombre maximum d'enfants par nœud »

La *figure 5.5* représente le nombre maximum d'enfants par nœud en fonction de N et Ψ. Notons que *DSCA* vise le plus petit nombre d'enfants par nœud parmi tous les algorithmes de planification d'agrégation, ce qui signifie que la durée de vie d'un nœud dans le réseau sera plus longue. Il peut également être remarqué que *NCA*, *DAS*, *CIAS*, et *SAS* visent le plus grand nombre. Bien que *ACS* adopte un critère de sélection de niveau moins relaxée que *NCA*, *DAS*, *CIAS* et *SAS*, il vise un nombre plus bas d'enfants par nœud. Ceci peut être expliqué comme suit: la structure de l'ensemble dominant construite par *NCA*, *DAS*, *CIAS* et *SAS* conduit à la création d'une architecture à base de cluster, qui signifie que le plus grand nombre de nœuds va choisir le même parent. *SDA* est mieux qu'*ACS* car il sélectionne les parents des trois niveaux et il n'utilise pas de structure de

l'ensemble dominant. *DSCA* surpasse tous les algorithmes d'agrégation en termes de durée de vie du réseau ainsi, les critères de sélection de parent adoptée par DSCA conduisent à un arbre d'agrégation optimale. Chaque nœud choisit le parent avec le minimum de voisins qui ne sont pas encore planifiés, et donc les enfants sont bien répartis entre les nœuds parents.

3.5. Messages générés

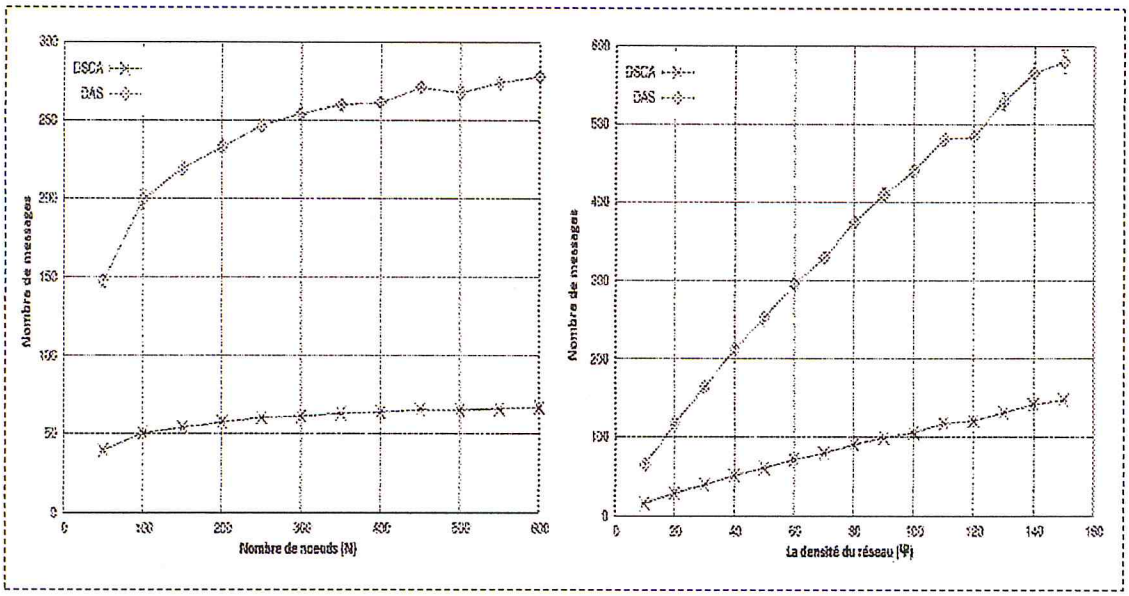

Figure 5.6 « Nombre de messages »

La *figure 5.6* montre l'impact de la variation de N et Ψ sur le nombre de messages générés. Nous remarquons que DSCA génère moins de messages que DAS. Les nœuds de *DSCA* communiquent uniquement avec leurs voisins directs, alors que chaque nœud de DAS doive communiquer avec des nœuds dans son ensemble concurrent qui sont 2 sauts plus loin.

4. Simulations et Tests réels de DADIDA

Le Framework proposé est implémenté en utilisant le système d'exploitation TinyOS [6]. Au cours de l'élaboration du Framework DADIDA nous avons utilisé TOSSIM simulateur [51] pour déboguer et corriger les problèmes. L'avantage principal d'utiliser TinyOS et TOSSIM est que le même code peut être déployé dans de véritables nœuds-capteurs plus tard. Dans notre implémentation, nous avons supprimé le protocole MAC standard fournis avec TinyOS. DADIDA est une approche cross-layer qui adopte un protocole MAC sans contention. DADIDA n'utilise pas CCA (*Clear Channel Assessment*), ni de backoff en cas d'envoi de données. Avrora [52] est un émulateur qui prend en compte les différents protocoles de la pile et les limitations hardwares des vrais nœuds-capteurs. Dans cette partie, les performances du Framework proposé sont évaluées par le biais d'Avrora ainsi qu'un véritable test-Bed composée à 11 capteurs MicaZ. L'utilisation d'Avrora nous donne plus de flexibilité pour faire varier le nombre de nœuds et le degré de réseau lors de l'évaluation de DADIDA.

Dans ce qui suit, nous présenterons d'abord les différents outils que nous avons créé afin de nous aider pour l'évaluation des simulations, ensuite nous expliqueront les paramètres considérés afin d'évaluer DADIDA. Enfin, nous verrons les simulations et les résultats expérimentaux.

4.1 Simulation de DADIDA

Avant de tester le Framework que nous avons implémenté sur de vrais nœuds capteurs nous devons le tester en simulateur. Afin de nous aider pour l'utilisation du simulateur et l'interprétation de ces résultats nous avons eu recourt à deux modules que nous avons créé :

1. le premier module est un Framework adapté à notre protocole et aux simulations que nous voulons réaliser. Ce Framework encapsule le simulateur Avrora et lui fournit pour chaque simulation les paramètres d'entrée dont il a besoin afin de mener à bien les simulations :

- Un générateur de topologies qui est basée sur Networkx lui fournit une topologie (fixe ou aléatoirement générée) avec en entrée le nombre de nœuds et leur degrés de connexion.
- Le temps de la simulation.
- Le code source compilée, à exécuter.
- réinitialise le répertoire */log* (contient tous les fichiers de retours de débogage de tous les nœuds) afin d'éviter des confusions de *log* lors de l'interprétation.
- A la fin de la simulation, vérifie grâce à une procédure écrite en python qu'il n'y a pas de collisions entre les nœuds dues à un éventuel problème de planification de slots.
- Enfin, génère une image topologie avec l'arbre d'agrégations obtenues et les slots dessinés.

2. Le second module est un générateur de graphes basé sur Gnuplot [50], ce dernier qui est chargée d'interpréter (dessiner) les données retournées pas les capteurs (*Log*) pour les trois (3) axes de simulations en variant le nombre de nœuds et le degré du réseau.

Le choix pertinent des axes de comparaison est un détail de première importance puisqu'il s'agit de retourner des informations précises sur les délais (en milliseconde) d'acheminement des données entre des intervalles de slots très restreint. Comme nous l'avons mentionnée dans le Chapitre -II- les techniques de Duty-cycling ne doivent être appliqués que si le coup de la réactivation du nœud est plus couteux qu'une radio en mode « idle » pendant ce temps puisque la durée moyenne d'un slot est de 2 à 3 milliseconde. Les résultats de simulations vont être évalués en fonction des indicateurs suivants :

1. **Goodput :** Chaque nœud non-feuille dans le réseau attend de recevoir les données de ses enfants, puis calcule et transmet le résultat global à son parent. Chacun d'eux ajoute ses données lors de l'exécution du processus d'agrégation. Par conséquent, théoriquement le nombre de données participant au processus

d'agrégation à la station de base est égal au nombre de nœuds dans le réseau. Cependant, il se peut que quelques paquets soient perdus pendant le processus de collecte en raison de facteurs externes, tel que d'autres réseaux sans fil (802.11, Wi-Fi, etc...) ou encore des objets mobiles tel que les personnes dans les bureaux, qui perturbent les communications entre les nœuds. Lorsque les données agrégées d'un nœud avec un nombre important de prédécesseurs est perdu, l'exactitude des données peut être affectée de façon significative. Afin de montrer la performance de l'exactitude des données de DADIDA, l'indicateur de Goodput est considéré. avec cet indicateur, nous comptons le pourcentage des données nœuds ayant participé au processus d'agrégation. En d'autres termes, le Goodput est le pourcentage de données ayant participé au processus d'agrégation.

2. **Latency :** cet indicateur de temps de latence est a été utilisé pour évaluer DSCA dans la partie précédente afin de compter le nombre de slots générés par DSCA. Contrairement à la latence de temps utilisée dans la partie précédente, la latence de données se réfère ici à la fois au temps en millisecondes nécessaire à la station de base pour recevoir les données à partir du nœud le plus éloigné dans chaque Frame. En d'autres termes, la latence des données est le temps en millisecondes pour recevoir les premières données générées dans le réseau dans chaque frame. Généralement, le temps de latence des données augmente en fonction de la durée et du nombre de slots.

3. **Duty-Cycle :** L'émetteur-récepteur radio est le composant qui consomme le plus d'énergie dans un nœud capteur [27]. La plupart de l'énergie est consommée dans l'émetteur-récepteur radio pas uniquement lors de la transmission ou la réception de paquets, mais également lorsqu'il est allumé (c.-à-d. nœud en mode actif) [24]. Pour cette raison, la plupart des solutions proposées dans la littérature visent à garder le nœud en mode veille (c.-à-d. la radio éteinte) autant que possible. dans notre cas cet indicateur correspond au pourcentage de temps où un nœud est en mode actif par rapport au total de la Frame.

Afin d'avoir des résultats exacts lors de la simulation, chaque point tracé représente une moyenne de 10 exécutions. Les plots sont présentés avec un intervalle de confiance de 95%. Dans notre simulation, N nœuds sont déployés de manière aléatoire sur une surface carrée de longueur D selon une distribution uniforme. Nous considérons que r est le

niveau de puissance de l'émetteur de chaque nœud dans le réseau, p est fixé pour tous les nœuds, et donc l'évaluation d'algorithmes est réalisée en faisant varier le nombre de nœuds du réseau.

4.1.1. Goodput

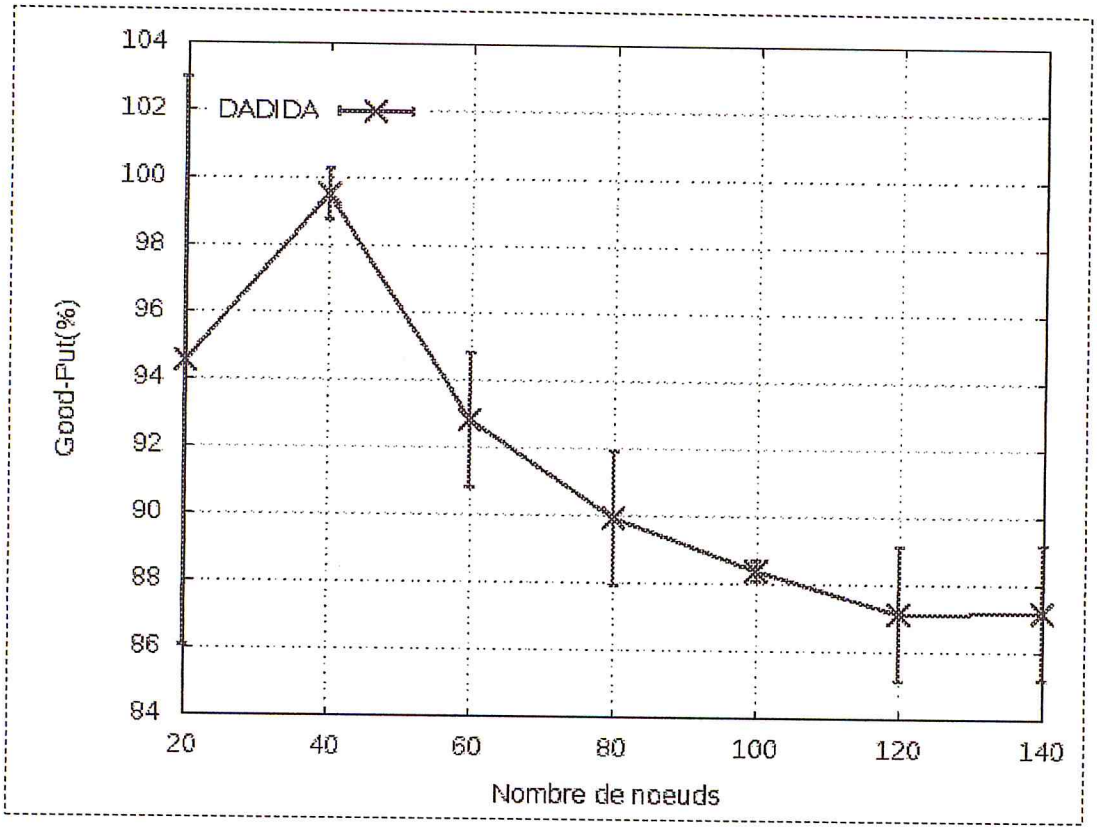

Figure 5.7- « Goodput de DADIDA»

La *figure 5.7* montre les résultats de la simulation du Goodput en fonction de N. Le premier constat qu'on peut tirer de cette figure, c'est que quel que soit le nombre de nœuds le Goodput de DADIDA dépasse les 87%. La réalisation de haute bon posée par Dadida peut être expliqué comme suit: L'utilisation de la contention MAC supprime les collusions que l'utilisation de la fonction d'agrégation réduit le nombre de paquets et résout le problème de la congestion. L'augmentation du nombre de nœuds N augmente la probabilité de perdre le paquet d'un nœud avec un nombre de prédécesseurs important. Pour cette raison, comme indiqué dans la *figure 5.7* le Goodput diminue en fonction de N.

4.1.2. Latency

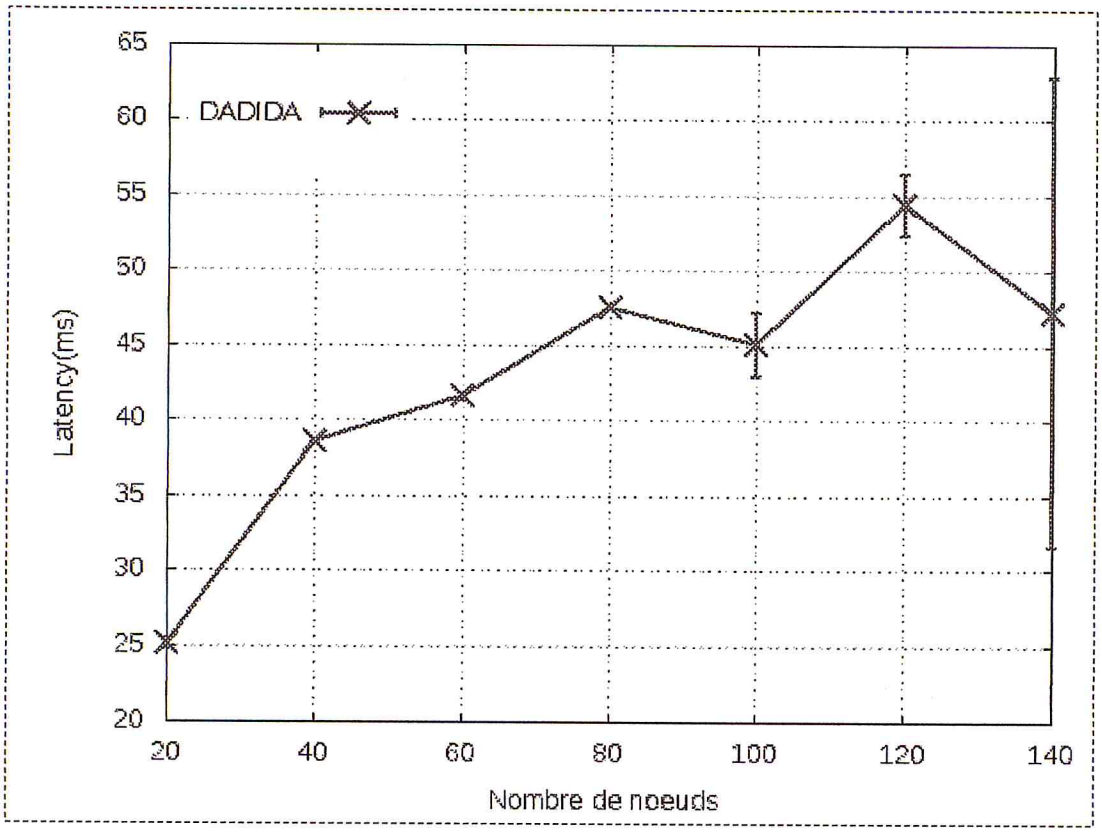

Figure 5.8 « Latence de temps de DADIDA »

La *figure 5.8* montre les résultats de la simulation de latence de données en fonction de N. Dans cette simulation, nous avons fixé la durée de la fente à 3 millisecondes. Chaque nœud retransmet les mêmes données à son parent au cours d'un slot jusqu'à ce que les données soient correctement reçues, ou que la durée du slot soit écoulée. Comme représenté à la *figure 5.2*, le nombre de slots augmente proportionnellement avec le nombre de nœuds. Pour cela, comme indiqué dans la *figure 5.8*, le temps de latence des données augmente lorsque le nombre de nœuds augmente. Dans la *figure 5.8*, quel que soit le nombre de nœuds, la latence de données ne dépasse pas 55 millisecondes.

4.1.3. Duty-Cycle

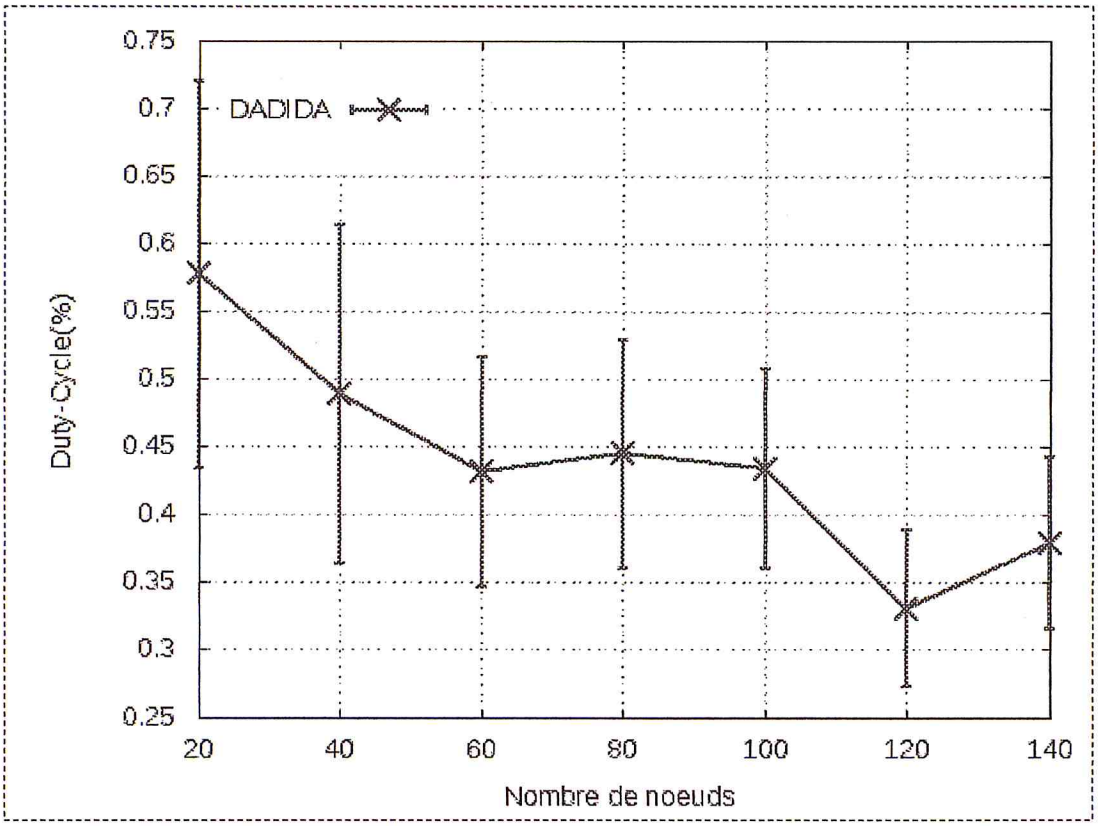

Figure 5.9 « Duty-Cycle de DADIDA »

La figure 5.9 montre les résultats de simulation du duty-cycle en fonction du nombre de nœuds. La durée de vie d'un capteur sans fil est liée à la façon avec laquelle le nœud consomme son énergie, et celle-ci est liée en grande majorité à l'utilisation de sa carte radio. L'algorithme que nous avons implémenté dans le module Data-manager de DADIDA, assure l'allumage minimum de la carte radio du nœud exécutant le Framework DADIDA. *La figure 5.9* montre qu'au début de l'exécution de la simulation (20 nœuds) l'indicateur Duty-Cycle montre un très faible rapport de moins de 0,55% de la longueur de la Frame avec la carte radio allumé. L'augmentation du nombre de nœuds implique l'accroissement du degré de connexion des graphes de topologies et ainsi la moyenne du nombre de fils par nœuds augmentant le nombre de Slot d'écoute ce qui augmente le rapport du duty-cycle mais l'architecture basée TDMA d'un protocole d'agrégation assure le minimum de consommation d'énergie et élimine la congestion.

4.2. Tests réels de DADIDA

La simulation des réseaux de capteurs consiste principalement en la reproduction du comportement et du fonctionnement des nœuds capteurs dans un environnement de simulation. Mais la simulation ne peut se substituer complètement à la réalité, par exemple les effets de perturbation radio ne peuvent pas être simulés avec précision sur un simulateur et le décalage de synchronisation ne peut pas être totalement simulé sur simulateur il y a donc un besoin évident de passer à l'expérimentation réelle. Dans cette section nous allons évaluer les performances de DSCA encapsulée dans notre Framework DADIDA dans des essais réels. Nous avons utilisé pour cela un réseau composé de 11 nœuds capteurs de type MicaZ déployés à l'intérieur d'un immeuble au sein du CERIST. La surface de cette zone de déploiement est approximativement égale à 12 x 12 mètres. Le bâtiment est occupé par le personnel durant les heures de travail, ce qui permet de tester la fiabilité de notre implémentation en présence d'obstacles fixes (les murs) et mobiles (les personnes). En plus, cette zone est couverte partiellement par un réseau Wifi, permettant aussi de tester l'effet des perturbations radio.

4.2.1 Description du déploiement des nœuds

La *figure 5.10* montre le déploiement des nœuds capteurs dans les locaux du CERIST en essayant de garder une répartition uniforme, le choix de l'emplacement n'est pas totalement libre puisque nous devions les placer dans des emplacements spécifique vu que les capteurs ont besoin de connexion filaire (LAN) afin de retourner leurs résultats logs sans perturber le trafic. La figure 5.11 montre les liens créés par les nœuds lors de leur mise en marche avec une puissance de 7dB. Nous avons choisis spécialement cette puissance afin de montrer que l'application de ce protocole sur un réseau ayant besoin d'un acheminement multi-saut, à cause des multitudes d'obstacles physiques présents dans l'enceinte du bâtiment dans lequel nous avons fait les tests d'une puissance moins importante aurait créé des zones de topologies non connectée au reste de la topologie.

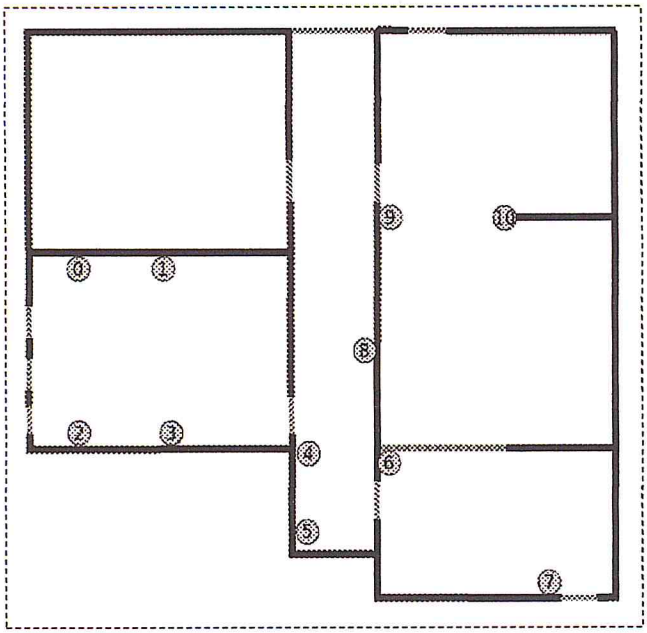

Figure 5.10- « Schéma des locaux et disposition des nœuds »

4.2.2. Description de la topologie de simulation

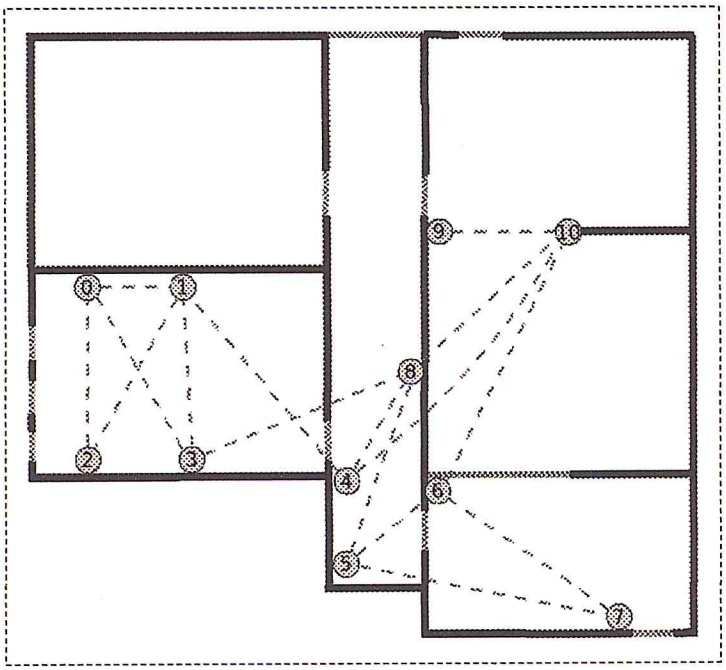

Figure 5.11 – « Schéma des liens obtenu »

Dans la figure 5.11, on voit clairement que quatre 4 niveaux de nœuds se sont formés, et que le nombre maximum de voisins est 4, ce qui rend ce déploiement idéal pour les tests que l'on veut réaliser. Ainsi sa structure multi-saut est un atout important afin de voir les trois types de sélection de parents (à partir du niveau supérieur, même niveau et niveau précèdent). Le nombre important de voisins va pousser le code à être testé dans le cas de forte concentration de nœuds dans cette topologie.

4.2.3. Description de la planification des nœuds

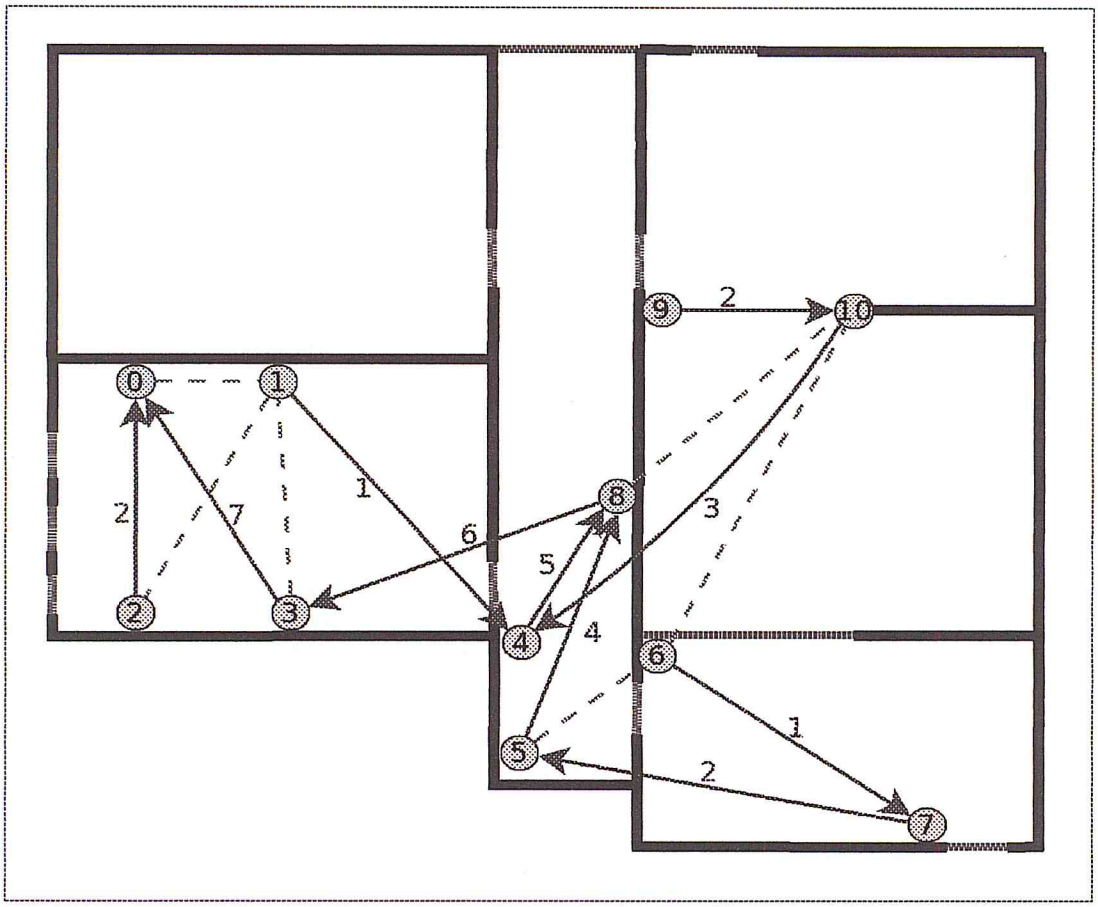

Figure 5.12 « Schéma de planification obtenu »

Dans le reste de la simulation l'identifiant de la station de base est le 0. La figure 5.12 montre le résultat de la planification effectuée par le protocole DSCA. L'exécution de DSCA dans cette topologie va créer 7 slots que sera le maximum du temps de latence. Ce nombre de slots est justifié par le nombre de sauts dans la topologie (4) et le degré de la topologie (4) afin que les nœuds les plus éloignés atteignent la station de base. Le nombre de prédécesseur pas nœud contraint également à accroitre le nombre de slots.

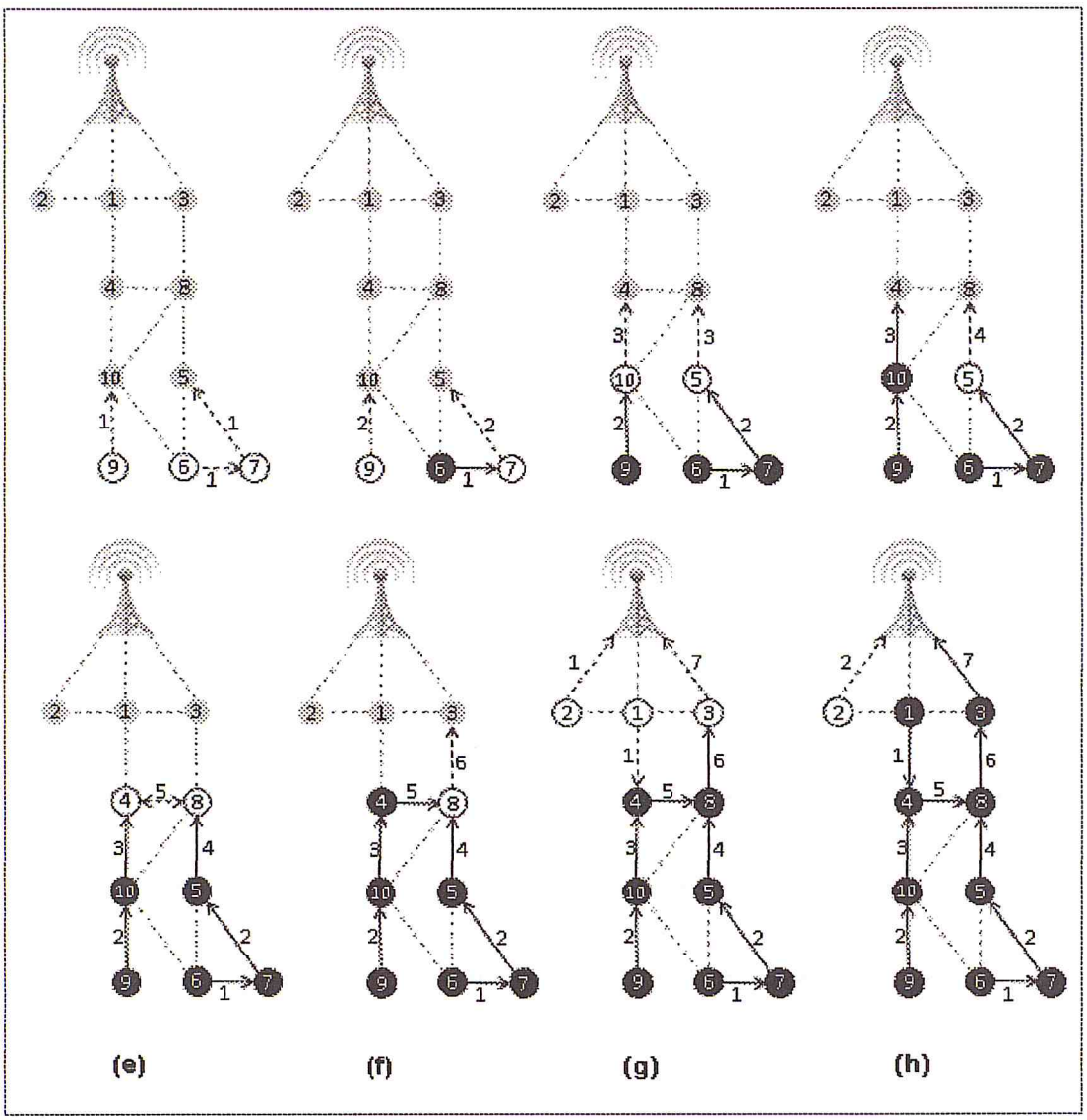

Figure 5.13 « Etapes de la planification »

Dans la *figure 5.13* sont décrites les étapes de la planification de la topologie testée. Sur cette figure les nœuds ont été réorganisés selon leurs profondeurs de telle sorte que les nœuds avec le même nombre de sauts soient aux mêmes niveaux. Et voici la description de chaque état :

(a): seuls les nœuds 7, 6 et 9 sont prêts et peuvent donc choisir le slot 1, en sélectionnant respectivement les nœuds 5, 7 et 10 comme meilleur parent. Les transmissions simultanées des nœuds 9 et 6 dans le cas où ils sont tous deux planifiées créeraient

une collision au niveau du nœud 10. Les nœuds 6 et 7 créeront aussi une collision au niveau du nœud 5. Comme le nœud 6 a une meilleure configuration que les nœuds 7 et 9.

(b): Le nœud 6 est planifié en premier lieu, les nœuds 9 et 7 fixent leurs slot à 2 et sélectionnent respectivement les nœuds 10 et 5.

(c): Les deux nœuds 7 et 9 sont automatiquement planifiés au cours du slot 2 du moment qu'il n'y a pas de conflits. Les nœuds 5 et 10 deviennent prêts à être planifié, ces derniers sélectionnent respectivement les nœuds 4 et 8 à partir du niveau supérieur comme meilleur parent au slot 3. Les transmissions simultanées des nœuds 5 et 10 créeront une collision au niveau du nœud 8.

(d): du moment que le nœud 10 a une meilleur configuration, il est planifié en premier, et le nœud 5 fixe son slot à 4.

(e): les nœuds 4 et 8 deviendront prêts. Les nœuds 4 et 8, chacun d'eux sélectionne l'autre comme parent au slot 5. Comme chacun des nœuds 4 et 8 a sélectionné l'autre en tant que meilleur parent, la procédure de planification choisi le meilleur des deux nœuds à être planifiée. Dans ce cas les deux nœuds ont la même configuration (nombre de voisins non planifié et nombre de voisin du parent non planifie), c'est la valeur de l'identifiant qui va les départager.

(f): Le nœud 4 est planifié en premier et le nœud 8 sélectionne le nœud 3 au slot 6.

(g): Le nœud 8 est planifié et les nœuds 1, 2 et 3 deviennent prêts. Les nœuds 2 et 3 sélectionnent le nœud 0 (La station de base) aux slots 1 et 7 respectivement, si considère que le nœud 1 sélectionne le nœud 4 à partir du niveau inferieure comme meilleur parent au slot 1. Les transmissions simultanées du nœud 1 et 2 créeront une collision au nœud 0.

(h): Le nœud 3 est planifié en premier, Comme le nœud 1 est meilleur que le nœud 2, le nœud 1 est planifié au slot 1 et le nœud 2 choisis le slot 2.

Vu que la réalisation de test réels n'offre pas autant de flexibilité quant au nombre de nœud utilisé et leurs degrés (topologie fixe) les axes d'évaluation devront changer par rapport aux simulations effectuées sur Avrora tel que on va étudier les résultats obtenu par rapport aux autres nœuds afin de voir l'impact des différents critères sur chaque nœud.

4.2.4. Résultats Obtenu

a. Goodput

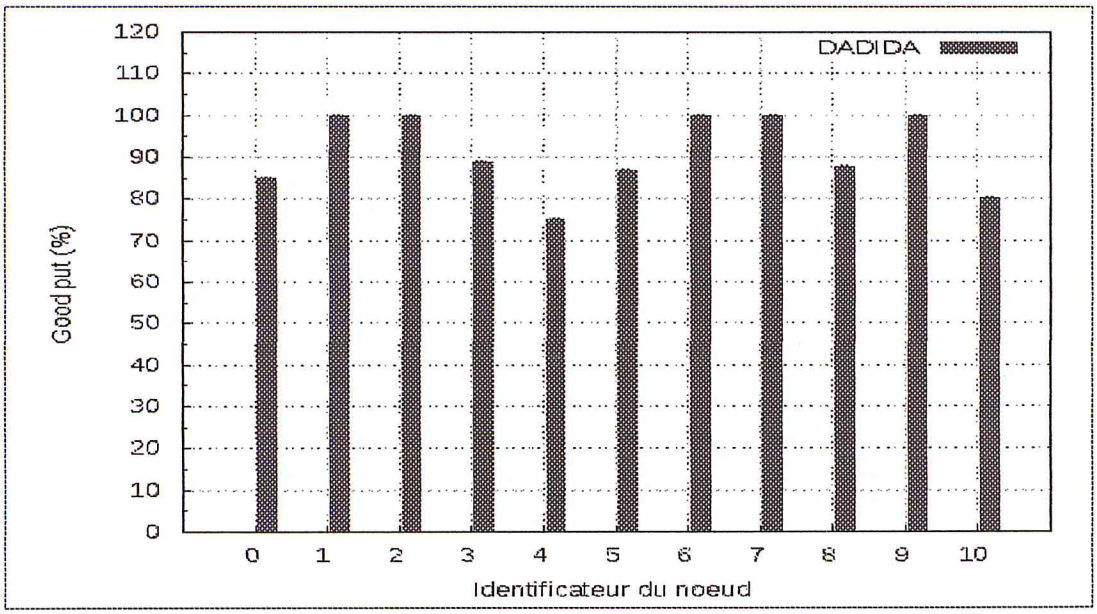

Figure 5.14 « Goodput »

La *figure 5.14* montre le Goodput pour chaque nœud, tel que les nœuds 1, 2, 6 et 9 sont des nœuds feuilles (aucun slot de réception) expliquant une moyenne de goodput 100%. Le nœud 7 obtient la moyenne de 75% qui est la plus basse de tous les autres nœuds, cela est dû au fait qu'il a le plus grand nombre de prédécesseur. Le goodput de la station de base est de 85% qui est correct vus le faible cout de latence, le nombre de prédécesseurs et le nombre de sauts dans cette topologie.

b. Latency

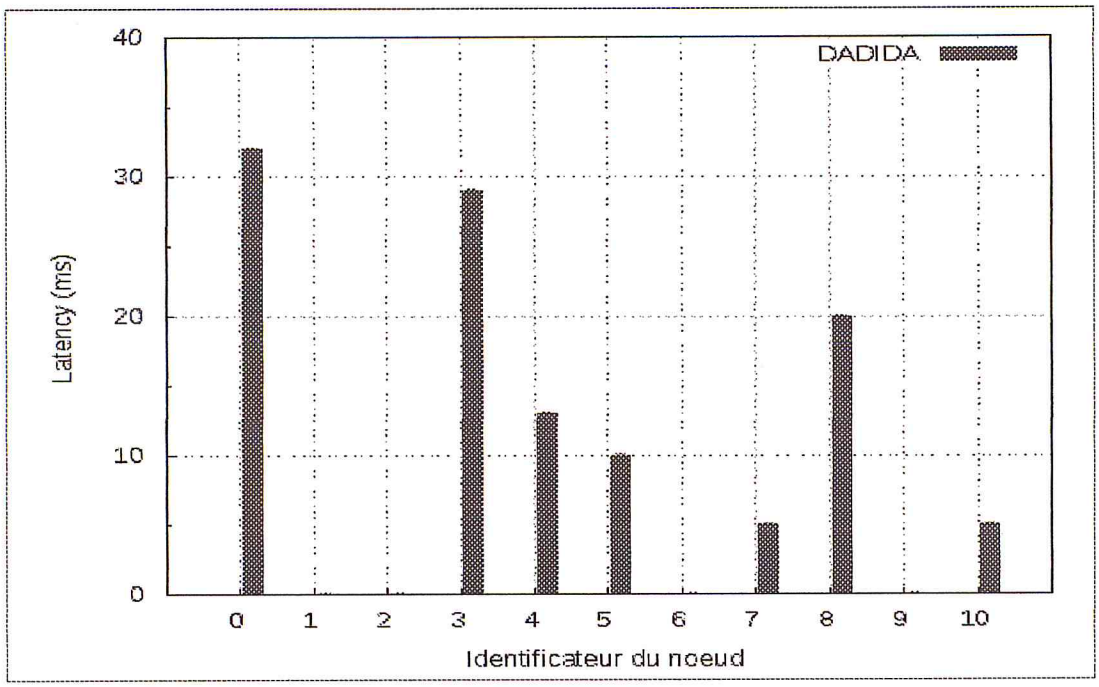

Figure 5.15 « La latence »

Dans la figure 5.15 les nœuds 1, 2, 6 et 9 sont des nœuds feuilles qui enverront donc leurs données sans devoir attendre les slots de leurs prédécesseurs, ensuite viennent les nœuds 7, 10 qui reçoivent les paquets de leur fils lors du premier slot ce qui explique une latence aussi faible, la station de base est le dernier nœud à recevoir les messages ce qui justifie la latence la plus importante entre tous les nœuds de cette topologie vu que la station de base doit attendre que tous les nœuds aient finis d'envoyer leurs données.

c. Duty-cycle

La *figure 5.16* montre le Duty-cycle du temps qu'as passé la carte radio allumée par rapport au temps total de la frame. Tel que les nœuds gardent toujours un très faible taux de duty-cycle vue le faible nombre de slot dans lesquels ils sont actif (un maximum de 3 slots pour le nœud 4). Les nœuds 5, 7, 9, ont un duty cycle plus important, cela s'explique par la séparation des deux slots de réception et d'envoi ainsi que les pertes des messages, ils ont donc dû garder leur carte radio plus longtemps allumée afin de transmettre leur paquets.

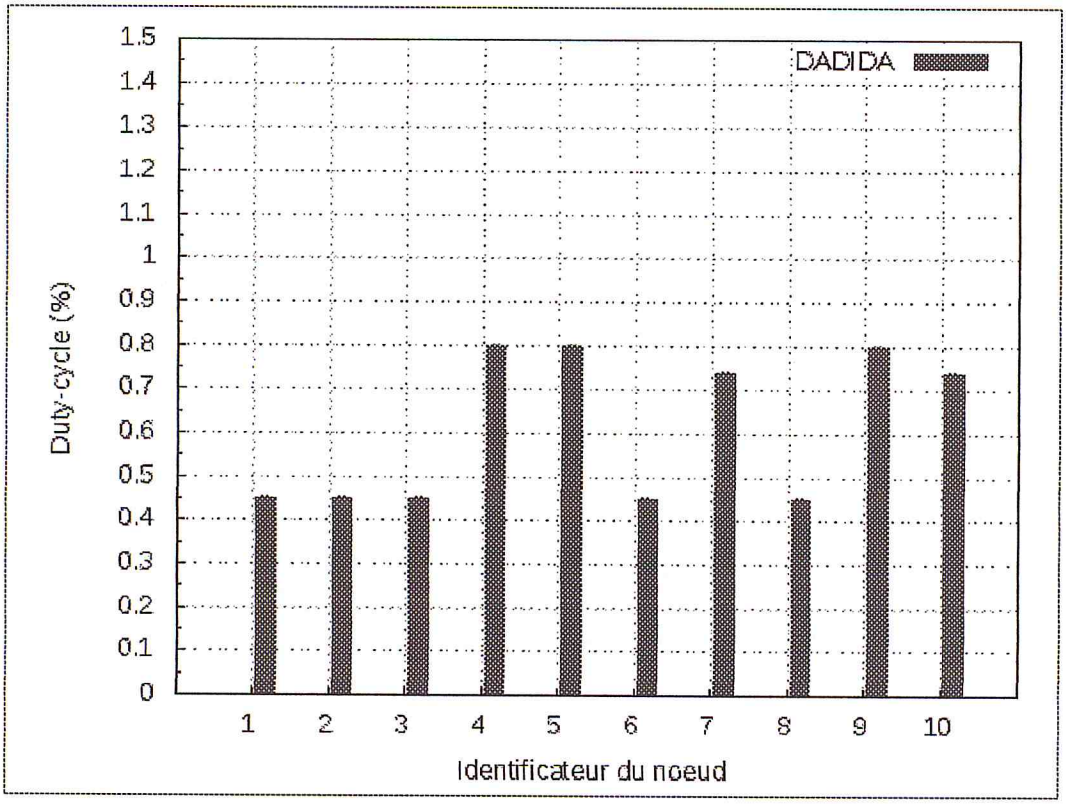

Figure 5.16 « Duty cycle »

5. Conclusion

La durée de vie et les délais de transmission des données d'un *RCSF* sont étroitement liés à la durée de vie du nœud et à la latence qu'il génère. L'architecture d'un réseau et les modes d'acheminement des données sont des facteurs critiques dans la consommation d'énergie et la durée de vie du réseau. Dans l'état-de-l' art il existe différents protocoles traitant la planification de l'agrégation avec des stratégies de planification différentes.

Dans ce chapitre nous avons mis en œuvre le Framework *DADIDA* dans lequel est encapsulé le protocole de planification d'agrégation *DSCA*. Les différents tests réalisés et résultats obtenu montrent que l'algorithme que nous avons implémenté surpassait tous les algorithmes déjà existants, en termes de latence et aussi en termes de durée de vie du réseau.

Conclusions générales et perspectives

Les réseaux de capteurs sans fil font partie des axes de recherches les plus importants. D'après la revue *MIT Technology Review* du MIT (Massachusetts Institute of Technology), les réseaux de capteurs sans fil font partie du TOP-10 des technologies de l'avenir. L'importance de ce domaine vient essentiellement du faible coût des nœuds capteurs et la multitude des applications qui peuvent être réalisée par ce type de réseaux. Comme nous l'avons cité au niveau du premier chapitre, les réseaux de capteurs sans fil peuvent être utilisés dans beaucoup de domaines parmi eux: le domaine militaire, le domaine environnemental, le domaine industriel etc... Cependant, limités en pratique par les problèmes d'énergie, de mémoire et de leur portée asse limitée constituent un frein à leur exploitation et un défi à relever. Plusieurs solutions ont été proposées dans la littérature afin de pallier à ces défis et rendre les réseaux de capteurs plus efficace et utilisable. Le but essentiel de ses solutions est la minimisation de la consommation d'énergie afin d'augmenter la durée de vie du réseau. L'agrégation de données fait partie des techniques les plus utilisées dans les solutions afin de minimiser la consommation d'énergie. Lorsque l'agrégation de données est utilisée, les données redondantes sont supprimées en utilisant des techniques d'agrégation. La suppression de ces données diminue le nombre de paquets acheminés à partir des nœuds capteurs vers la station de base, produisant un impact positif sur la consumation d'énergie.

Depuis quelques années, l'utilisation des réseaux de capteurs a connu une évolution très importante, certains domaines dans lesquels ils sont utilisés ont des contraintes de délai (où les données doivent être reçues par la station de base dans de brefs délais). Avant de faire l'agrégation des données, chaque nœud dans le réseau doit attendre les données de ses prédécesseurs afin d'éliminer les redondances. Par conséquent, l'utilisation de l'agrégation de données peut devenir une source de problèmes pour les applications à contrainte de délai. Afin d'alléger l'impact de l'agrégation de données sur ce type d'applications. Des protocoles de planification de l'agrégation ont été proposés afin minimiser le délai d'acheminement des données tout en prenant en considération le débit du réseau. Afin d'atteindre ces objectifs, les protocoles de planification d'agrégation de données adoptent une approche cross-layer qui prend en considération la couche *MAC* et la couche routage en même temps. La technique d'accès au média utilisé dans ces protocoles est *TDMA* afin d'éliminer les collisions et les retransmissions.

Les protocoles de planification de l'agrégation proposés dans la littérature n'ont jamais été testés ou simulés dans des conditions réelles. Les protocoles proposés généralement sont centralisés et sont testés sur des micro-ordinateurs ordinaires, là où les pertes de paquets lors de l'exécution de protocoles sont totalement négligées. Par conséquent, la simulation de ces solutions dans un simulateur tel que TOSSIM ou Avrora est indispensable, ces derniers prennent en considération les pertes de paquets et reproduisent parfaitement les limites réelles du nœud. Dans ce projet nous avons implémenté le protocole DSCA en utilisant TinyOS. Après l'implémentation de ce protocole, nous l'avons simulé en utilisant Avrora tout en variant le nombre de nœuds. La même implémentation de DSCA a été testée sur un réseau expérimental de capteurs constitué de 11 capteurs MicaZ.

Durant l'implémentation du protocole DSCA, nous avons rencontré plusieurs problèmes. Pour lesquels, nous proposons les perspectives suivantes:

- L'utilisation de TDMA nécessite que les nœuds capteurs doivent être totalement synchronisés. La synchronisation de tous les nœuds de réseau peut causer deux genres de problèmes: (1) la consommation d'énergie du moment que ce processus est gourmand en énergie; (2) la synchronisation de tous le réseau est un processus complexe et ne permet pas le passage à l'échelle. Afin de résoudre ce problème, nous proposons comme perspective de faire une synchronisation localisée uniquement là où les fils synchronisent avec leur parent.

- Tous les protocoles proposés dans la littérature ne prennent pas en considération le changement de topologies, l'ajout ou la suppression des nœuds ou même des fois les pertes de liens. Afin de surmonter ces limitations, nous proposons comme deuxième perspective de proposer une solution qui intègre le routage multi-chemins avec la planification de l'agrégation.

Références bibliographiques

[1] Deprez P, Lampriniere C, Ligot D & Roche S. Réseaux embarqués – « Les WSN et leurs domaines d'application » ENSEIRB, **2008.**

[2] Larbaoui N "La sécurité dans les réseaux sans fils ad hoc", thèse doctorat, université Tlemcen, **2012.**

[3] Challal Y. « systèmes intelligents pour le transport – Réseaux de capteurs sans fil » Compiègne University of Technology, **novembre 2008.**

[4] CHEN X, HU X & ZHU J. "Minimum data aggregation time problem in wireless sensor networks". Lecture Notes in Computer Science Series, vol. 3794. Springer, p133–142, **2005.**

[5] HUANG S C-H, WAN P-J, VU C T, LI Y & YAO F. F. "Nearly constant approximation for data aggregation scheduling in wireless sensor networks" . Wireless Sensor Networks, IEEE INFOCOM , p366–372. **2007.**

[6] Site officiel de TinyOS. [En ligne] http://www.tinyos.net/

[7] Dudek D, Haas C, Kuntz A, Zitterbart M, Krger D, Rothenpieler P, Pfisterer D & Fischer S, "A wireless sensor network for border surveillance," Proc. of the 7th ACM Conference on Embedded Networked Sensor Systems, **Novembre 2009.**

[8] Faye S, « Contrôle du trafic routier urbain par un réseau fixe de capteurs sans fil », Institut Telecom Paris Tech (CNRS),Paris, France, **Mars 2013.**

[9] Hefeeda M & Bagheri M, "Forest fire modeling and early detection using wireless sensor networks," Ad Hoc & Sensor Wireless Networks, vol. 7, p.169–224, **2009.**

[10] Ceperley N. "Water use by Sclerocarya birrea Agroforestry Trees in Sudanian Savanna: Application of Wireless Sensor Networks". European Geosciences Union (EGU) General Assembly, Vienne, Autriche, **2011.**

[11] Ingelrest F, Barrenetxea G, Schaefer G, Vetterli M & Couach O. "SensorScope: Application-Specific Sensor Network for Environmental Monitoring", in ACM Transactions on Sensor Networks, vol. 6, num. 2, p. 1-32, **2010.**

[12] Abdelmoula P.-A & Parlange M. "Measurement of Leaf Area Index for Calculation of Water Consumption by Agroforestry Trees in Burkina Faso», Laboratoire de mécanique des fluides de l'environnement EFLUM , **2012.**

[13] G. Shobha, Ranjana R. Chittal & Kamod Kumar, « Medical Applications of Wireless Networks », *IEEE*, Systems and Networks Communications,. p. 82 – 82, **septembre 2007.**

[14] Beausse C, Bouguerra M, Hong Y-G & Moktad E-A, "Réalisation d'un réseau de capteurs de température sans fil basé sur le protocole ZigBee", ENSEIRB, p 3-5, **2007.**

[15] Réseaux de capteurs sans fil [En Ligne] : http://www.tafats.fr/Techniques/Reseaux_de_capteurs/Reseaux_capteur_text.html/

[16] Global Positioning System [En Ligne]: http://fr.wikipedia.org/wiki/GPS/

[17] Survey A, Dwivedi A. K, Tiwari M. K & Vyas O. P. "Operating Systems for Tiny Networked Sensors" International Journal of Recent Trends in Engineering, Vol. 1, No. 2, **Mai 2009.**

[18] Site Officiel de RetOS [En Ligne] : http://mobed.yonsei.ac.kr/retos/index.html/

[19] Jaein J, Sukun K & Alan B, « Network reprogramming », *TinyOS documentation*, **août 2003.**

[20] David G, Levis P, Von Behren R, Welsh M, Brewer E & Culler D, « The NesC Language : A Holistic Approach to Networked Embedded Systems », *PLDI'03, June 9–11, 2003, San Diego, California, USA.*, **juin 2003**.

[21] Pottie G.J and Kaiser W.J, "Wireless integrated network sensors". Magazine Communications of the ACM, 43(5), p 51-58, **2000.**

[22] VRaghunathan V, Schurgers C, Park S & Srivastava M B. "Energy-aware wireless microsensor networks". IEEE Signal Processing Magazine, p.40-50, **Mars 2002.**

[23] Alippi C, Anastasi G, Galperti C, Mancini F & Roveri M. "Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications". In Proceedings of the 4th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'07), p. 1-6, Pisa, Italy, **October 2007**.

[24] Ye W, Heidemann J & Estrin D. "Medium access control with coordinated adaptive sleeping for wireless sensor networks". IEEE/ACM Transactions on Networking, 12(3), p.493-506, **2004.**

[25] Ali M, Böhm A & Jonsson M. "Wireless sensor networks for surveillance applications - A comparative survey of MAC protocols". In Proceedings of the 4th

International Conference on Wireless and Mobile Communications (ICWMC '08), IEEE Computer Society. , pages 399-403, Washington, DC, USA, **2008.**

[26] Holger K & Willig A. "Protocols and Architectures for Wireless Sensor Networks" . Wiley, **2005.**

[27] KACIMI R, « Techniques de conservation d'énergie pour les réseaux de capteur sans fil », thèse doctorat, Université de Toulouse, **Septembre 2009.**

[28] chakravarthi R, Gomathy C, Sebastian SK, Pushparaj K & Mon V B "A survey on congestion control in wireless sensor networks" Revue internationale de l'informatique et de la communication Vol1, p. 161-164, **janvier-juin 2010.**

[29] Malar R T "Congestion control in wireless sensor networks based multi-path routing in priory rate adjustment technique" international journal of advanced engineering & Application, **Janvier 2010.**

[30] Yu B. Li J & Li y. "Distributed data aggregation scheduling in wireless sensor networks". In INFOCOM. IEEE,p. 2159–2167, **2009.**

[31] Wan P J, Alzoubi K M & Frieder O. "Distributed construction of connected dominating set in wireless ad hoc networks". In INFOCOM. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, p. 1597 - 1604 vol.3. **2002.**

[32] Malhotra B, Nikolaidis I & Nascimento M. "Aggregation convergecast scheduling in wireless sensor networks". Wireless Networks, publication 2,p.319-355.**2010.**

[33] Wang P, Yuan H & Huang L, "Approaching the Optimal Schedule for Data Aggregation in Wireless Sensor Networks" Wireless Algorithms, Systems, and Applications, p.26-35, **2010.**

[34] Xu X, Wang S, Mao X, Tang, S & Li X Y, "An improved approximation algorithm for data aggregation in multi-hop wireless sensor networks". In Proceedings of the 2nd ACM international workshop on Foundations of wireless ad hoc and sensor networking and computing. FOWANC '09. ACM, New York, NY, USA, 47–56. **2009.**

[35] Bagaa M, Lasla N, Ouadjaout A & Challal Y. "Sedan: Secure and efficient protocol for data aggregation in wireless sensor networks". In LCN. IEEE Computer Society, 1053–1060. **2007.**

[36] Zhu J & HU X, "Improved algorithm for minimum data aggregation time problem in wireless sensor networks" Jrl Syst Sci & Complexity, p.626-636, **mars 2008.**

[37] Wan P, Huang S C-H, Wang L, Wan Z & Jia X. "Minimum-latency aggregation scheduling in multihop wireless networks". In MobiHoc, E. W. Knightly, C.-F. Chiasserini, and X. Lin, Eds. ACM, p.185–194. **2009.**

[38] Madden S, Franklin M. J, Hellerstein, J. M & Hong W, "Tag: a tiny aggregation service for ad-hoc sensor networks," SIGOPS Oper. Syst. Rev., vol. 36, no. SI, pp. 131–146, **Decembre 2002.**

[39]: W. Huangfu, Y. Liu, B. Duan, L. Sun, J. Ma, and C. Chen, "Eata: Effectiveness based aggregation time allocation algorithm for wireless sensor networks," Proc. of the IEEE ISCC'08, **Juillet 2008.**

[40] CERIST (Centre de recherche sur l'information scientifique et technique) [En Ligne] http://www.cerist.dz/

[41] Hill J, Szewczyk R, Woo A, Hollar S, Culler D & Pister K, "System architecture directions for networked sensors". ACM SIGPLAN, vol. 35, num. 11, p. 93–104. **2000.**

[42] TinyOs [En Ligne] http://moodle.utc.fr/

[43] Documentation officielle [En Ligne] http://docs.tinyos.net/tinywiki/

[44] Levis P & Gay D, "TinyOS programming". Cambridge university press, 978-0-511-50730-4, Londre, **2009.**

[45] Levis P, Madden S, Polastre J, Szewczyk R, Whitehouse K, Woo A, Gay D, Hill J, Welsh M, Brewer E & Culler D, « TinyOS: An Operating System for Sensor Networks », *Ambient Intelligence*, **2005.**

[46] David G, Levis P, David Culler D & Brewer E, "nesC 1.1 Language Reference Manual ", **Mai 2003.**

[47] Tolle G, Levis P, & Gay D, "Source and Sink Independent Drivers", Core Working Group, TEP: 114, Publiée: **30-Oct-2005**, Modifiée: **10-01-2007.**

[48] Site officiel de Python [En Ligne] http://www.python.org/

[49] Site officiel de NetworkX [En Ligne] http://networkx.lanl.gov/

[50] Site officiel de Gnuplot [En Ligne] http://www.gnuplot.info/

[51] TOSSIM : [En Ligne] http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM/

[52] Avrora [En Ligne] http://compilers.cs.ucla.edu/avrora/

[53] S. R. G, Daw M, Prakash R & Venkatesan S, "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," **2003.**