Optimisation de la durée de vie d'un réseau de capteurs

Amine Abbas

THÈSE

Année 2007

Présentée à

Université de Franche-Comté
UFR Sciences et Techniques
Laboratoire d'Informatique de l'université de Franche-Comté

Pour obtenir le

GRADE DE DOCTEUR DE L'UNIVERSITÉ DE FRANCHE-COMTÉ
Spécialité Informatique, Automatique et Productique

Optimisation de la durée de vie d'un réseau de capteurs

par
Amine ABBAS

Soutenue le 28 juin 2007 devant la commission d'examen :

Rapporteurs	Hamamache Kheddouci	Professeur à l'Université Claude Bernard, Lyon 1
	Christian Michel	Professeur à l'Université Louis Pasteur, Strasbourg 1
Directeur de Thèse	Jacques Bahi	Professeur à l'Université de Franche-Comté
Co-encadrant	Michel Salomon	Maître de conférences à l'Université de Franche-Comté

Dédicaces

Sans votre soutien, ce travail n'aurait jamais vu le jour,

à vous et pour vous.

À mes parents car c'est d'eux que je tiens la détermination et la persévérance qui m'ont permis de mener à bien cette thèse. Que sa réalisation leur soit un témoignage de reconnaissance et d'affection, eux qui dans la simplicité et l'humilité ont su transmettre à leurs enfants les plus grandes et les plus nobles des valeurs morales. Vous être mes parents et je suis très fier d'être votre fils.

À toi très cher père, le plus courageux et le plus juste des hommes que je connaisse, je souhaiterai t'égaler un jour.

À ma chère mère, toi qui m'avait appris que l'homme se mesurait aux obstacles qu'il franchissait.

Pour toi Hadia Dahli, du fond du cœur merci de m'avoir supporté tout au long de cette période, pour ton soutien qui m'a permis d'arriver au terme de ce travail. Ta philosophie qui peut être résumée en une seule phrase : "Acts speak louder than words", deviendra mienne.

À mes frères et à ma soeur.

Je dédie cette thèse.

Remerciements

Ce travail a été effectué au sein de l'équipe Algorithmique Numérique Distribuée (AND) du Laboratoire d'Informatique de l'université de Franche-Comté (site de Belfort).

Je tiens à remercier en tout premier lieu Jacques M. Bahi et Michel Salomon, respectivement directeur de thèse et co-encadrant, pour m'avoir guidé tout au long de cette thèse. Sans leur confiance et patience, ce travail n'aurait sans doute pas abouti.

J'exprime ensuite mes plus vifs remerciements à Hamamache Kheddouci et Christian Michel pour avoir accepté de juger ce travail et d'en être les rapporteurs.

J'adresse toute ma reconnaissance à la ville de Belfort pour m'avoir octroyé une bourse de thèse. Sans ce soutien, ce travail n'existerait sûrement pas.

Je remercie Sylvain Contassot-Vivier pour m'avoir soutenu tout au long de la période allant de mon DEA au jour de ma soutenance.

J'exprime toute mon amitié à Kamel Mazouzi, je te remercie tout particulièrement, tu étais plus qu'un collègue ou un simple ami, je n'oublierai jamais ton soutien moral lors des moments difficiles, je ne trouve pas de mots plus expressifs que le merci si ce n'est de te dire Tanmirt Amga. Par la même occasion et du fond du cœur, merci à toi Jean-Luc Anthoine pour ton amitié sincère.

Mes remerciements vont également à Nacim Betrouni, Pierre-Cyril Heam et Flavien Vernier.

Enfin, j'adresse ma reconnaissance à tous les membres de l'équipe AND que je n'ai pas cités. Qu'ils trouvent ici l'expression de ma gratitude.

Table des matières

Table des figures

Liste des tableaux

Introduction

Le développement et l'évolution technique constante des réseaux informatiques rendent notre monde de plus en plus communiquant. Ainsi, les progrès réalisés ces dernières années dans le domaine de la miniaturisation des systèmes micro-électroniques et des technologies réseaux sans fil se sont notamment traduits par l'émergence d'un nouveau type de réseaux informatiques : les réseaux de capteurs sans fil (ou *Wireless Sensor Networks*). Schématiquement, un réseau de capteurs sans fil est un ensemble de nœuds capteurs capables de recueillir des informations sur leur environnement proche, comme par exemple une température ou un degré de pollution de l'air, et de les transmettre vers un nœud (ou plusieurs nœuds) chargé de les collecter. Comme les réseaux de capteurs répondent à un besoin majeur dans de nombreux domaines, à savoir l'observation d'un phénomène physique, ils sont en pleine expansion. Par exemple, dans le domaine de l'agriculture un réseau de capteurs sans fil permet une meilleure maîtrise de l'apport en eau aux plantations, en collectant des données telles que l'humidité du sol, l'ensoleillement ou les précipitations.

Du point de vue réseau, les réseaux de capteurs s'inscrivent dans le contexte des réseaux ad hoc. Néanmoins, leurs spécificités rendent caduques la plupart de approches existantes pour les réseaux ad hoc. C'est pourquoi, le domaine des réseaux de capteurs est un domaine de recherche très actif. Il faut en particulier noter que la problématique majeure dans le monde des capteurs sans fil est l'énergie. En effet, les nœuds capteurs ont des capacités énergétiques relativement limitées et non renouvelables. De ce fait, et afin de maintenir le réseau fonctionnel le plus longtemps possible, il est capital de définir des mécanismes de gestion de l'énergie et / ou des protocoles de communication économes en énergie. Ainsi, l'objectif de cette thèse est d'étudier les moyens permettant d'assurer la plus grande longévité possible d'un réseau de capteurs.

Au niveau des protocoles, de nombreux travaux ont été effectués. Ces recherches concernent d'une part la couche liaison de données, plus précisément la sous-couche MAC, avec le développement de protocoles d'accès au médium de communication, d'autre part la couche réseau avec les protocoles de routage et de gestion de la topologie. Plutôt que de définir un énième protocole de communication, nous avons choisi de développer une approche au niveau de la couche applicative. Un des avantages étant que cette approche est complémentaire avec les protocoles de communication. Le résultat de notre approche est un algorithme complètement distribué, exécuté par chaque nœud et nécessitant uniquement des communications avec ses voisins

immédiats (*one-hop neighbors*). Notre travail se distingue également par la prise en compte de l'hétérogénéité des nœuds et leur mobilité. Il a été validé théoriquement, par le biais d'une preuve mathématique, et expérimentalement, via des simulations de réseaux de topologies variées.

Ce mémoire de thèse est organisé en deux parties. La première partie est dédiée à un état de l'art sur les réseaux de capteurs en orientant l'étude sur les problèmes liés à l'énergie. La seconde partie présente les travaux réalisés dans le cadre de cette thèse et plus particulièrement l'algorithme proposé pour augmenter la durée de vie d'un réseau de capteurs.

La partie I comporte trois chapitres. Le premier chapitre est consacré à la présentation de généralités sur les réseaux de capteurs, telles que la description de ce qu'est un réseau de capteurs, son objectif ou encore ses applications. Le second chapitre se focalise sur la problématique de la durée de vie d'un réseau de capteurs. Enfin, le dernier chapitre aborde les environnements d'expérimentation en décrivant, d'une part des plateformes matérielles existantes (c'est-à-dire de vrai nœuds capteurs qui peuvent être déployés), d'autre part différents outils de simulation.

Dans la partie II, nous décrivons notre contribution pour prolonger la durée de vie d'un réseau de capteurs. Dans le premier chapitre de cette partie nous exposons le principe de l'approche proposée. Un réseau de capteurs hétérogènes est notamment considéré afin d'illustrer notre propos. L'approche que nous proposons étant basée sur l'équilibrage d'un ratio d'énergies à travers le réseau de capteurs, c'est tout naturellement que nous nous inspirons des techniques d'équilibrage de charge dans les systèmes distribués. L'algorithme de migration de tâches que nous proposons afin d'équilibrer les ratios est défini dans le chapitre qui suit. Il est introduit après une description formelle du problème que l'on cherche à résoudre. La convergence de l'algorithme, c'est-à-dire des valeurs de ratio, est prouvée mathématiquement et expérimentalement. L'étude de la pertinence de l'approche proposée est l'objet des deux derniers chapitres. Plus précisément, dans l'avant dernier chapitre nous nous sommes intéressé à la vitesse de convergence des ratios et au gain en durée de vie que procure l'algorithme pour diverses topologies de réseaux statiques. Nous mettons également en évidence le fait que l'algorithme permet d'assurer l'exécution de toutes les tâches présentes dans le réseau. Le scénario considéré est un réseau de capteurs multimédias utilisé à des fins de détection d'intrusion. Les considérations énergétiques sont présentées en détail : énergie consommée lors de l'exécution d'une tâche et modèle de consommation d'énergie des communications. Finalement, le dernier chapitre aborde les expérimentations dans le contexte des topologies dynamiques. Le simulateur développé est décrit ainsi que quelques résultats.

Première partie

État de l'art sur les réseaux de capteurs

Chapitre 1

Qu'est-ce qu'un réseau de capteurs

1.1 Définition et objectif

Un réseau de capteurs est un ensemble coopérant de nœuds capteurs utilisant des communications sans fil dans le cadre d'un réseau ad hoc [1, 2]. Le déploiement des nœuds peut être contrôlé ou non et est en général relativement dense (un très grand nombre de nœuds, pouvant aller jusqu'à plusieurs milliers de nœuds). La zone dans laquelle les nœuds sont déployés est appelée zone d'intérêt ou d'étude, elle peut éventuellement être inaccessible à l'être humain. Chaque nœud comporte un capteur permettant d'observer un phénomène se produisant à proximité. Dès lors qu'un nœud détecte un phénomène d'intérêt, il produit une donnée qui doit être envoyée par l'intermédiaire de sauts successifs (acheminement multi-sauts) entre nœuds voisins vers un nœud collecteur (également appelé puit - *sink*). Le rôle du nœud collecteur étant de servir de point d'accès aux données. L'utilisateur peut accéder aux données soit directement, soit par l'intermédiaire d'une liaison longue distance, notamment en utilisant Internet ou une liaison satellite. La figure 1.1 illustre ces propos.

Un nœud capteur doit donc être capable de collecter, sérialiser et acheminer des données de façon coopérative vers un (voire plusieurs) nœud(s) collecteur(s). Du point de vue physique, il s'agit d'un dispositif de taille relativement limitée, de quelques centimètres (voire moins d'un centimètre), à plusieurs dizaines de centimètres, soit par exemple le gabarit d'un assistant personnel (PDA) [3]. Au niveau économique, idéalement, afin de pouvoir déployer des réseaux de capteurs relativement denses, le coût unitaire d'un nœud ne devrait pas dépasser quelques dollars. La réalité est bien autre, puisque la plupart des nœuds capteurs qui sont commercialisés actuellement coûtent plusieurs dizaines de dollars, voire plusieurs centaines. Le prix d'un nœud capteur est souvent un bon indicateur de ses performances. Schématiquement, un nœud capteur est composé de quatre unités :

- une unité de capture (produisant une donnée) ;
- une unité de traitement (capacités de calcul, stockage) ;
- une unité de communication/transmission sans fil ;
- une source d'énergie (généralement une batterie).

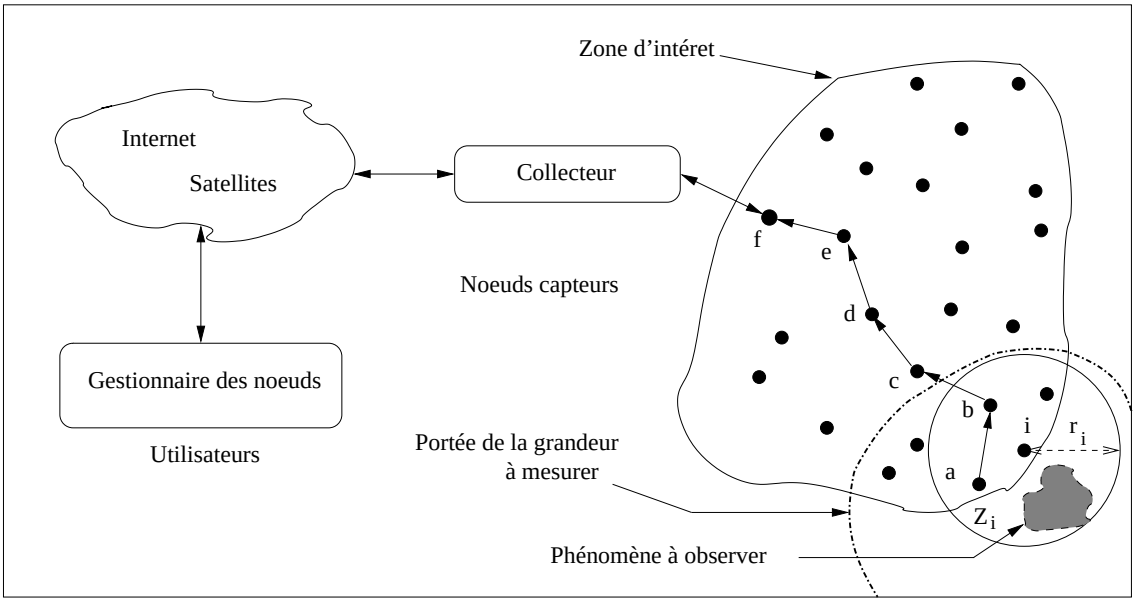


FIG. 1.1 – Description d'un réseau de capteurs.

Un nœud peut, suivant les besoins résultants de l'application visée, comporter des unités supplémentaires à des fins de localisation et de mobilité. Nous reviendrons plus en détails sur l'architecture d'un nœud capteur ultérieurement.

Les nœuds d'un réseau communiquent entre eux de façon autonome, selon une topologie qui peut être statique ou dynamique. Comme le montre la figure 1.1, un nœud i peut communiquer directement avec tout nœud se trouvant dans sa zone de couverture Z_i, définie par son rayon de couverture r_i. Le rayon de couverture dépend bien entendu de la nature du médium sans fil que comporte un nœud. Ainsi, les nœuds a et b qui se trouvent dans la zone de couverture sont « accessibles » en un saut, c'est-à-dire par une communication directe par le nœud i (on parle alors de *one-hop neighbors*). Tandis que toute communication avec le nœud c ou f requiert le passage par des nœuds intermédiaires, induisant une communication multi-saut (on parle alors de *multi-hop neighbors*). Toutes les notions relatives aux communications, à savoir le voisinage et les différentes topologies possibles du réseau sont définies lors du déploiement des nœuds, à l'issue de la phase d'auto-configuration.

Le nœud collecteur est un nœud qui assure la liaison entre le réseau de capteurs et la station de traitement et d'analyse des données récoltées. Du fait de sa fonction, il diffère d'un nœud capteur au niveau des performances de ses composants. Plus précisément, il est notamment doté d'une unité de traitement plus puissante (processeur, capacité de stockage), d'interfaces de communication permettant d'acheminer les données sur de longues distances et d'une autonomie plus importante au niveau énergétique. Par exemple, dans l'article [4] qui décrit l'architecture, le déploiement et les performances d'un réseau de capteurs surveillant l'habitat d'oiseaux marins dans le nord-est des États-Unis, le nœud collecteur est un ordinateur portable. Ce

dernier archive les données collectées par les nœuds capteurs dans une base de données, celle-ci étant répliquée en temps normal toutes les quinze minutes sur une machine du laboratoire Intel localisé à Berkeley grâce à une liaison satellite. Dans certains cas, le nœud collecteur n'est pas uniquement en charge de la centralisation des données, mais peut également commander les nœuds du réseau (diffusion de requêtes de contrôle). Dans ce cas on parle aussi de nœud coordinateur.

1.2 Quelques domaines d'application

L'observation et le contrôle de phénomènes physiques est depuis fort longtemps un besoin dans de nombreux domaines. De fait, la surveillance de grandeurs physiques telles que la température, la pression ou bien la radioactivité est primordiale pour beaucoup d'applications, qu'elles soient à visée industrielle, scientifique, ou autre. Les réseaux de capteurs permettent de répondre à des besoins existants, mais également d'envisager de nouvelles applications. En effet, les caractéristiques des nœuds capteurs, notamment leur coût économique relativement faible (en théorie), l'utilisation d'un support de communication à technologie sans fil, et surtout l'existence d'un large spectre de capteurs, permettent aux réseaux de capteurs de se développer et de se démocratiser de plus en plus. Aussi, le champ d'application des réseaux de capteurs est très vaste [1, 5] :

- l'industrie (surveillance, suivi de processus industriels) ;
- les transports (véhicules « intelligents ») ;
- l'habitat (domotique) ;
- l'environnement ;
- la santé ;
- la défense/sécurité ;
- etc.

Dans la suite, nous allons donner pour quelques domaines d'applications des exemples précis d'utilisation de réseaux de capteurs. Ceux-ci reflètent la très grande variété de besoins auxquels un réseau de capteurs permet de répondre. Ils montrent également que bien que l'architecture globale soit la même pour tout réseau de capteurs, l'application visée n'induit pas toujours les mêmes contraintes sur les différents composants d'un nœud capteur. Par exemple, l'environnement de fonctionnement peut notamment être de nature très différente (parfois hostile à toute présence humaine), certaines applications critiques nécessitent que la latence de la transmission des données soit la plus basse possible (on peut citer la prévention de catastrophes telles que accident chimique ou tremblement de terre), etc.

- **Applications bio-médicales**

  Dans le domaine de la santé, l'usage de capteurs est surtout envisagé à des fins de surveillance des fonctions vitales d'un patient, que ce soit pour établir un diagnostic ou effectuer un suivi [6]. Il s'agit typiquement de collecter des informations d'ordre physiologique telles que le niveau de glucose dans le sang ou les pulsations. Néanmoins, des applications plus ambitieuses sont atten-

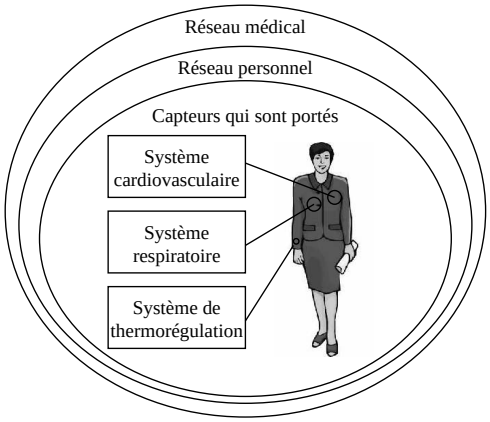


FIG. 1.2 – Réseau de capteurs pour la surveillance à domicile d'un patient.

dues, comme le contrôle d'organes à transplanter ou la détection de tumeurs cancéreuses [7]. Il faut aussi souligner que les premières mises en œuvre de capteurs médicaux consistait en des capteurs isolés. L'exemple classique est la micro-caméra qui permet d'explorer l'appareil digestif d'un malade sans avoir recours à la chirurgie. L'étude de la mise en place de vrais réseaux de capteurs est plus récente, de plus elle cible essentiellement le suivi de patients à leur domicile [6, 8]. Dans [8], les chercheurs proposent une sorte de bulle de capteurs de comportant différents niveaux autour du patient (certains capteurs seront portés par le patient, d'autres seront dans l'environnement où il évolue, cf. figure 1.2), reliée à un réseau médical distant. Ce concept d'habitat intelligent pour la santé, favorisant le maintien à domicile, fait également l'objet de recherches actives en France [9].

La conception d'un nœud capteur médical pose des contraintes propres et strictes, plus ou moins fortes suivant l'application ciblée [7]. Considérons le cas d'un capteur devant être introduit dans le corps humain, il faut alors intégrer des contraintes telles que :

- la taille du nœud et la bio-compatibilité des matériaux utilisés ;
- la perturbation des communications sans fil par les tissus et leur nocivité (l'utilisation des hautes fréquences est ainsi à proscrire) ;
- la toxicité des batteries ;
- etc.

- **Applications liées à l'habitat**

  La sécurité des biens et des personnes (détection d'intrusion, surveillance d'une personne âgée à son domicile, etc.) constitue une application majeure. L'amélioration de la qualité de service et du coût de systèmes tels que le chauffage et son pendant qu'est la climatisation offre une autre opportunité d'utilisation des réseaux de capteurs. On voit immédiatement l'impact positif de cet usage à l'heure du réchauffement de la planète. Rien que pour les États-Unis on estime à plusieurs millions de tonnes de carbone le gain d'une meilleure maîtrise de la température dans les bâtiments.

- **Applications environnementales**

  L'environnement offre un champ d'application très étendu pour les réseaux de capteurs. En effet, des domaines tels que l'agriculture de précision, l'écologie ou encore les sciences naturelles s'appuient sur l'observation de paramètres environnementaux complexes et variés. Les réseaux de capteurs à vocation environnementale se caractérisent en particulier par une zone d'intérêt souvent géographiquement très étendue et fréquemment hostile à la présence humaine. De plus, dans le cadre d'une utilisation pour la prévention ou le suivi de catastrophes touchant l'environnement, qu'elles soient d'origines naturelles (telles que feux de forêt, tsunami, etc.) ou humaine (pollution résultant d'un accident lors du transport de produits chimiques, par exemple) la robustesse du réseau de capteurs est un facteur prépondérant.

  À titre d'exemple, plusieurs travaux [10, 11, 12] ont mis en évidence la pertinence de l'utilisation d'un réseau de capteurs en viticulture. En effet, ils ont montré en déployant des réseaux de capteurs en vraie grandeur que les bénéfices étaient multiples, les principaux étant :
  - une amélioration de la production au niveau rendement et qualité notamment grâce à la prévention des phénomènes de gel ;
  - la préservation de l'environnement avec une meilleure maîtrise des apports de pesticide et d'eau.

  En sciences naturelles divers travaux de recherches ont étudié l'apport d'un réseau de capteurs, que ce soit pour surveiller l'activité d'un volcan [13] ou afin d'observer, sans perturber, des espèces animales dans leur environnement naturel [4].

- **Applications militaires**

  Comme dans le cas d'autres technologies, le domaine de la défense n'est pas étranger au développement des réseaux de capteurs. La principale raison est que ceux-ci offrent de nouvelles possibilités dans le renseignement militaire, permettant d'obtenir des informations plus fines et mieux actualisées. Parmi la multitude d'utilisations possibles, on peut citer la localisation des unités sur le champ de bataille (véhicules, combattants) ou d'un tireur isolé en environnement urbain [14], la détection, la classification et le suivi d'une intrusion [15].

1.3 Spécificités par rapport à un réseau ad hoc

Ainsi que nous l'avons souligné précédemment, les réseaux de capteurs utilisent un support de communication sans fil, généralement corrélé avec une absence d'infrastructure (notamment à cause de la problématique de préservation de l'énergie). C'est pourquoi, un réseau de capteurs peut être vu comme faisant partie de l'architecture réseau de type ad hoc. En effet, les réseaux ad hoc sont des réseaux dits multi-sauts, composés d'hôtes autonomes communiquants au moyen de liaisons sans fil. Dans ces réseaux, l'absence d'infrastructure est palliée par le fait que chaque hôte peut servir de relais (ou routeur) pour acheminer le trafic réseau provenant d'autres nœuds [16].

Les réseaux de capteurs font donc appel à des techniques développées pour les réseaux ad hoc. Cependant, du fait des caractéristiques intrinsèques d'un nœud capteur, la plupart des algorithmes et protocoles définis dans le cadre des réseaux ad hoc ne sont pas utilisables directement. En effet, si on compare les réseaux de capteurs aux réseaux ad hoc, on fait les constatations suivantes :

- le besoin de coopération entre les nœuds est primordial dans les réseaux de capteurs (auto-organisation), afin de réduire la consommation d'énergie et atteindre l'objectif du réseau ;
- la capacité énergétique d'un nœud capteur est très nettement plus limitée, d'où la nécessité d'une bonne gestion de la consommation énergétique ;
- les réseaux de capteurs comportent généralement beaucoup plus de nœuds et ceux-ci sont déployés de manière très dense ;
- les capacités de traitement (calcul et stockage) d'un nœud capteur sont sans commune mesure (très faibles) comparées à celles d'un hôte d'un réseau ad hoc ;
- dans un réseau de capteurs les changements de topologie sont très fréquents, car les nœuds sont plus sujets à défaillance. De plus, des nœuds peuvent être ajoutés au réseau lors d'un redéploiement ;
- au niveau des communications, les réseaux de capteurs utilisent surtout des communications par diffusion (*broadcast*) alors que les réseaux ad hoc ont plus recours aux communications point à point (*unicast*). D'ailleurs, le très grand nombre de nœuds et l'absence de contrôle lors du déploiement font souvent qu'un nœud capteur n'est pas identifié de manière absolue dans le réseau (absence d'identifiant global).

Les contraintes à prendre en compte dans un réseau de capteurs ne sont donc pas du même ordre que dans le cas d'un réseau ad hoc, d'où le besoin de nouveaux algorithmes et protocoles.

1.4 Défis et contraintes

Lorsque l'on cherche à définir de nouveaux algorithmes et / ou protocoles pour réseaux de capteurs, il faut tenir compte de plusieurs contraintes. Nous en évoquons ci-après quelques-unes [1].

1.4.1 Architecture d'un nœud capteur

Schématiquement, quelle que soit l'application considérée, un nœud capteur comporte quatre unités distinctes, à savoir :

- une unité de capture ;
- une unité de traitement ;
- une unité de communication (émission et réception) ;
- une unité fournissant l'énergie.

En plus de ces quatre composantes et pour répondre à des besoins spécifiques résultants de l'application envisagée, un nœud peut contenir des composants supplémentaires tels qu'un système de localisation (global ou local). Ce dernier système peut

en particulier être de type GPS (*Global Positioning System*). Néanmoins, pour des raisons de coût (notamment énergétique) il est souvent remplacé par un protocole de localisation s'appuyant sur un ou plusieurs nœuds servant de balise (*beacon node(s)*). Il existe également des capteurs disposant d'un système leur permettant de bouger dans la zone d'intérêt. La figure 1.3 présente les différentes unités / composantes et leurs interactions.

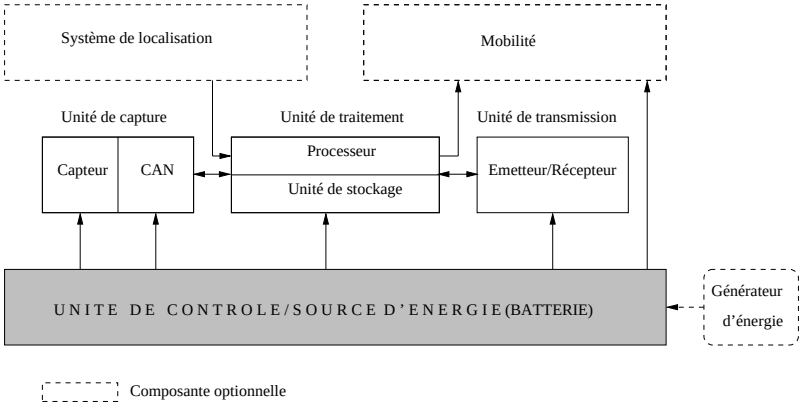


FIG. 1.3 – Architecture d'un nœud capteur

- **Unité de capture**

  C'est cette unité qui va engendrer des données lors de l'observation d'un phénomène d'intérêt. Elle se compose de deux sous-unités : le capteur et un convertisseur analogique/numérique. Le convertisseur permet de transformer la mesure fournie par le capteur en un signal numérique qui peut être traitée par l'unité de traitement.

- **Unité de traitement**

  L'unité de traitement fournit les capacités de calcul et de stockage du nœud capteur. Pour ce faire, elle exécute un système d'exploitation spécifique tel que TinyOS. Naturellement, l'unité de traitement est en charge de l'exécution de tout algorithme / protocole, et plus particulièrement des protocoles de communication. Parmi les autres tâches que peut assurer l'unité de traitement, on peut citer la fusion de données, voire dans certains cas leur analyse.
  À titre d'exemple, voici tout d'abord les caractéristiques du nœud capteur de type MICA2, un COTS *Mote*, car construit à partir de *Commercial Off The Shelf components*. Celui-ci appartient à la famille des *Berkeley Motes* : des nœuds capteurs résultants d'un projet de recherche mené au sein de cette université. Il comporte un microcontrolleur 8 bits ATMegaé128L fonctionnant à 8 MHz et dispose d'une mémoire flash programmable de 128 Ko, ainsi que d'une SRAM et d'une EEPROM, toutes deux d'une taille de 4 Ko. Côté stockage des données issues de mesures, 512 Ko sont prévus. Vu sa taille $58 \times 32 \times 7$ (mm) et sa source d'énergie, deux piles AA, on comprend facilement le pourquoi de

ces caractéristiques. À l'opposé, une plateforme telle que la Stargate SPB400 [17] offre des performances supérieures : processeur Intel PXA255 Xscale à 400 MHz, SDRAM et mémoire flash respectivement de 64 et 32 Mo. Bien qu'utilisable comme nœud capteur, en particulier pour obtenir des données de type multimédia (des images) dans le cadre d'application de type réseau de surveillance. Le nœud Stargate est plutôt prévu par son fabricant pour jouer le rôle de nœud collecteur. MICA2 et Stargate SPB400 sont tout deux des produits de la société Crossbow Technology [18].

- **Unité de communication**

  Les communications entre nœuds capteurs (émission et réception) constituent l'élément clé d'un réseau de capteurs, du fait de sa nature collaborative. Elles se font par l'intermédiaire d'un support (ou médium) de communication sans fil qui peut être de nature optique (comme dans les nœuds Smart Dust), infrarouge ou radiofréquenciel (type WiFi). L'optique est robuste par rapport aux problèmes d'interférences électriques mais suppose que chaque nœud soit en vue directe avec ses voisins (pas d'obstacle), ce qui est évidemment une limitation très forte. L'infrarouge est moins coûteux, de conception aisée et est intégré dans de nombreux équipements (PDA, téléphone portable, ordinateurs), mais souffre du même problème que l'optique.
  Les supports de type radiofréquence (RF) sont les plus usités en raison de leur très large diffusion (WiFi, Bluetooth, GSM, etc.), offrant des bandes de fréquences de puissances variées (bandes ISM telles que 433 MHz, 2,4 GHz, etc.). De plus, les unités de communication utilisant les ondes radio bénéficient des avancées dans les circuits d'émission-réception RF (niveau d'intégration par exemple). La principale limitation a pour origine la consommation énergétique. En effet, le rayon de la zone de couverture est directement lié à la puissance d'émission du signal, or l'énergie consommée est elle-même proportionnelle à la puissance du signal. La solution serait d'augmenter la taille de l'antenne, mais l'encombrement d'un nœud capteur devant être réduit cela n'est pas possible. Le meilleur compromis semble offert par l'ultra haute fréquence (ou UHF, *Ultra high frequency* en anglais) qui correspond à la bande de radiofréquences comprises entre 300 MHz et 3 GHz.

- **Unité fournissant l'énergie**

  L'alimentation électrique est assurée par une batterie qui peut être simplement une pile, par exemple AAA, ou encore une batterie Li-Ion. Sa capacité dépend d'une part de son type, d'autre part de sa taille. Évidemment, ces deux contraintes dépendent des caractéristiques du nœud. L'énergie étant disponible en quantité limitée, c'est une ressource qui doit être utilisée à bon escient. Aussi, l'aspect consommation énergétique doit être appréhendé aux niveaux matériel et logiciel (algorithmes et protocoles) d'un nœud capteur. Enfin, il est à noter qu'une solution envisagée pour augmenter l'autonomie est l'utilisation de cellules solaires.

1.4.2 Tolérance aux pannes et passage à l'échelle

Des nœuds capteurs peuvent tomber en panne pour diverses raisons : destruction due à l'environnement, problème matériel ou logiciel, énergie épuisée. Il est donc clair que quelques nœuds défaillants ne devraient pas affecter le fonctionnement global du réseau, d'où la notion de tolérance aux pannes d'un réseau de capteurs. Garantir un certain niveau de tolérance aux pannes suppose que cet aspect soit pris en compte dans les algorithmes et protocoles.

Le passage à l'échelle est un autre point important dans les réseaux de capteurs, car le nombre de nœuds peut aller de quelques dizaines à plusieurs centaines, voire milliers suivant l'application visée. La robustesse par rapport au nombre de nœuds d'un algorithme ou d'un protocole est donc une caractéristique très recherchée. Un algorithme sera d'autant plus utilisé qu'il sera peu sensible à l'évolution du nombre de nœuds du réseau.

1.4.3 Topologie d'un réseau de capteurs

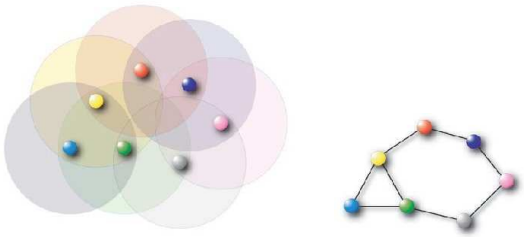
La topologie associée à la formation d'un réseau de capteurs est souvent indéterminée, évoluant au cours du temps. Elle peut être vue comme le résultat de trois phases :

1. **Déploiement des nœuds capteurs**
   Dans cette phase, les nœuds sont déployés à travers la zone d'intérêt soit de façon contrôlée, soit aléatoirement (dispersion depuis un avion par exemple). Le placement manuel permet un contrôle relatif de l'organisation des nœuds, mais n'est possible que pour des réseaux de taille limitée et uniquement dans le cadre d'un environnement de fonctionnement non hostile.

2. **Post-déploiement, fonctionnement normal du réseau**
   Après déploiement, la topologie peut être amenée à évoluer en raison de changements propres à chaque nœud, tels que :
   – modification de la position (en particulier si le nœud est mobile) ;
   – perturbation des communications (obstacle, interférences, etc.) ;
   – épuisement de l'énergie ;
   – défaillance.

3. **Redéploiement, ajout de nœuds capteurs**
   Le réseau de capteurs doit être capable d'intégrer, aisément, de nouveaux nœuds. Un des buts de l'ajout de nœuds peut être de pallier la défaillance d'un trop grand nombre de nœuds. En tout état de cause, le réseau doit pouvoir se réorganisé rapidement et avec un coût énergétique limité.

Remarquons que dans un réseau de capteurs constitué de nœuds disposant tous de la même interface de communication sans fil (rayons de couverture identiques), la matrice d'incidence associée au graphe modélisant les communications est symétrique (voir les figures 1.4(a) et (b)).



(a) (b)

FIG. 1.4 – Réseau de capteur et graphe de communication associé.

1.5 Architecture de communication

Akyildiz *et al.* [1, 19] ont proposé qu'à l'image de n'importe quel autre type de réseau et comme le montre la figure 1.5, les algorithmes et protocoles mis en œuvre dans les réseaux de capteurs soient structurés suivant le modèle ISO-OSI (*Open System Interconnect*). Toutefois, leur proposition induit le passage d'un modèle plan à un modèle tridimensionnel. Les plans supplémentaires rendent compte de la distribution de contraintes telles que la consommation énergétique sur toutes les couches. Un autre modèle d'architecture pour réseaux de capteurs sans fil est celui dénommé Zigbee [20, 21] et qui s'appuie sur la norme IEEE 802.15.4 que nous aborderons plus tard. À proprement parler, Zigbee est le nom d'un consortium [22] à l'image de WiFi pour les normes 802.11. D'autres modèles, comme celui proposé dans [23], s'éloignent plus nettement du modèle OSI classique.

Dans la suite, nous allons surtout nous intéresser aux trois couches les plus basses, orientées communications. Il est à souligner qu'au niveau de la couche application on trouve essentiellement des protocoles permettant à l'utilisateur d'interagir avec les nœuds du réseau de capteurs, en particulier par le le biais de requêtes. Enfin, en ce qui concerne la couche transport, de nouveaux protocoles sont nécessaires car les protocoles conventionnels tels que TCP et UDP ne sont pas très adaptés aux réseaux de capteurs. Parmi les protocoles de transport développés pour assurer un transport fiable et peu coûteux en énergie on peut citer ESRT [24] et k-RTP [25].

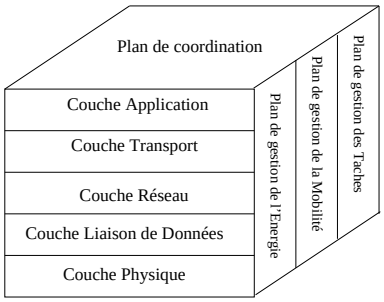


FIG. 1.5 – Architecture de communication - Modèle OSI pour les réseaux de capteurs

1.5.1 Couche physique

La couche physique est en charge de la gestion des transmissions radio, pour ce faire elle définit la modulation des ondes radioélectriques (diverses fréquences radio sont disponibles, permettant de répondre à différents niveaux d'exigence en particulier pour ce qui est de la propagation des ondes), l'encodage et la signalisation de la transmission. Dans les protocoles réseaux sans fil actuels (802.11a, 802.11b, 802.11g, Bluetooth, etc.), la couche physique offre deux types de transmission à modulation de fréquence associés à une modulation de phase. Les techniques de transmission à modulation de fréquence s'appuient sur la technique dite « d'étalement de spectre ».

La technique d'étalement de spectre, d'origine militaire, s'est imposée du fait de ses nombreux avantages. Il s'agit notamment d'une limitation des problèmes dûs aux interférences, la possibilité de faire cohabiter dans une même bande de fréquences plusieurs transmissions (partage de la bande passante), la résistance aux interceptions, etc. On distingue deux techniques d'étalement de spectre, à savoir :

- d'une part la technique à saut de fréquence, dénotée FHSS, pour *Frequency Hopping Spread Spectrum*. Celle-ci consiste à découper la bande de fréquences en canaux de 1 MHz, puis à transmettre en changeant de fréquence d'émission périodiquement (toutes les 300 à 400 ms) suivant une séquence préétablie ;
- d'autre part la technique à séquence directe, dite DSSS, pour *Direct Sequence Spread Spectrum*. Cette méthode divise également la bande de fréquences en canaux, mais beaucoup plus larges (d'une largeur de 22 MHz en 802.11). De plus il n'y a pas de saut lors de la transmission des données. Comme les canaux se chevauchent, pour compenser le bruit des canaux voisins, on a recours à la technique du *chipping*. Elle consiste à remplacer chaque bit par une séquence *Barker* de bits, afin de pouvoir effectuer des contrôles d'erreurs. Par exemple, la norme 802.11 définit une séquence de 11 bits (10110111000) pour coder un 1 et son complément pour représenter un 0.

Finalement, le débit en bits par seconde d'une transmission va résulter de la technique de modulation de phase choisie. Plusieurs techniques sont à disposition, offrant des débits plus ou moins élevés :

- BPSK (*Binary Phase-Shift Keying*) - encodage d'un bit à chaque changement de phase, ce dernier consistant en une rotation de 180° ;
- QPSK (*Quadrature PSK*) - même type d'encodage, mais l'utilisation d'une série de quatre rotations de 90° permet d'obtenir un débit deux fois plus élevé.

Afin d'améliorer les débits, diverses optimisations ont été intégrées dans les différentes normes. On peut noter en 802.11b la mise en œuvre de la méthode d'encodage CCK (*Complementary Code Keying*) à la place des séquences *Baker*. L'utilisation en 802.11a de la bande de fréquences des 5 GHz, offrant 8 canaux distincts, fait appel à la technique de modulation de fréquence dite OFDM (*Orthogonal Frequency Division Multiplexing*). Celle-ci tire partie des canaux en envoyant des données en parallèle sur plusieurs canaux. La norme 802.11g utilise également la technique OFDM, mais est compatible avec 802.11b (même bande de fréquences ISM et technique d'encodage).

Nous allons maintenant nous focaliser sur une norme de réseau sans fil de type

WPAN (*Wireless Personal Area Network*), i.e. une norme IEEE 802.15.x, comme le Bluetooth (IEEE 802.15.1). Comparé aux réseaux sans fil de type WLAN (normes 802.11x), un WPAN se caractérise en particulier par une portée des communications moindre. La norme IEEE 802.15.4 [26] a été définie pour répondre à certaines caractéristiques des réseaux de capteurs : en général un faible débit et une faible consommation électrique. Au niveau de la couche physique, 27 canaux sont disponibles, répartis dans trois bandes de fréquences. Ainsi que le montre le tableau 1.1, elles offrent respectivement des débits de 20 Kbps, 40 Kbps et 250 Kbps.

PHY	Bande de fréquences	Nb de canaux	Modulation fréquence	Modulation phase	Débits (Kbps)
868 MHz	868,0 à 868,6 MHz	1	DSSS	BPSK	20
915 MHz	902,0 à 928,0 MHz	10	DSSS	BPSK	40
2,4 GHz	2,4 à 2,4835 GHz	16	DSSS	O-QPSK	250

TAB. 1.1 – Norme IEEE 802.15.4 - Caractéristiques des transmissions.

Si on compare sur quelques critères la norme 802.15.4 aux normes 802.11b et 802.15.1 (cf. tableau 1.2), on constate qu'elle est clairement adaptée à la plupart des réseaux de capteurs. Elle est particulièrement prévue pour les réseaux de très grande taille, comportants plusieurs milliers de nœuds. Bien entendu, il est probable que dans un réseau de capteurs les standards Zigbee et WiFi soient amenés à cohabiter. A priori, il semblerait ne pas y avoir de problème particulier, à condition de faire attention au choix des fréquences utilisées [27].

Norme IEEE	Zigbee 802.15.4	WiFi 802.11b	Bluetooth 802.15.1
Portée (mètres)	1 à 100	1 à 100	1 à 10
Durée de vie (jours)	100 à 1000	0,5 à 5	1 à 7
Nombre de nœuds	> 64000	32	7
Débits (Kbps)	20 à 250	11000	720

TAB. 1.2 – Comparaison des normes 802.15.4, 802.11b et 802.15.1.

1.5.2 Couche liaison de données

La couche liaison de données permet d'assurer la fiabilité des communications, qu'elles soient point à point ou multipoints. Cela se fait par le biais de mécanismes tels que le contrôle d'accès au support / médium de communication ou le contrôle des erreurs de transmission. La couche liaison de données se divise en deux sous-couches :

- la couche LLC (*Logical Link Control*), standardisée dans la norme IEEE 802.2, commune à toutes les normes définies ultérieurement (802.11x, 802.15.x). C'est elle qui établie les connexions logiques entre les nœuds ;
- la couche MAC (*Medium Access Control*), qui répond à l'enjeu majeur qu'est la désignation du nœud qui a le droit d'émettre à un instant donné.

Idéalement, un protocole MAC doit permettre un accès équitable au médium de communication. Cela signifie que si un seul nœud veut transmettre, il devrait pouvoir disposer de toute la bande passante. Parmi les autres caractéristiques recherchées on a la simplicité et l'absence de synchronisation globale. De nombreux protocoles MAC sont disponibles. On peut tout d'abord les classer suivant le type de contrôle, à savoir centralisé ou distribué. Naturellement, dans le cadre d'un réseau de capteurs une approche distribuée est à privilégier. Ensuite, la classification peut être raffinée suivant la manière dont l'accès au médium est géré :

- accès contrôlé ou garanti (*contention-free protocols*) ;
- accès aléatoire ou par compétition (*contention-based protocols*)

Certains protocoles centralisés peuvent être vus comme une méthode hybride. Il s'agit notamment d'approches par passage de jeton et de réservation à la demande.

- **Protocoles à accès contrôlé**

  Cette famille contient des protocoles offrant un accès par segmentation ou multiplexage au médium de communication. Plus précisément, l'accès est partitionné en autant de parties que de nœuds. Différentes techniques de multiplexage ont été définies, chacune correspondant à une méthode de segmentation. La segmentation peut se faire par :
  - le temps - multiplexage TDMA (*Time Division Multiple Access*) ;
  - la fréquence - multiplexage FDMA (*Frequence Division Multiple Access*) ;
  - code - multiplexage CDMA (*Code Division Multiple Access*) ;
  - longueur d'onde - WDMA (*Wavelength Division Multiple Access*).

  Certaines méthodes hybrides associent CDMA et TDMA ou FDMA.

  Le principe du TDMA consiste en une allocation temporelle périodique du canal à chaque nœud du réseau. Ainsi, chaque nœud dispose de la totalité de la bande passante pendant un temps très court. Pour ce qui est du FDMA, la technique la plus ancienne, elle découpe la bande de fréquences en autant de sous-bandes (d'intersection nulle) que de nœuds. Le WDMA reprend cette approche, puisque c'est le pendant du FDMA dans le domaine optique. La dernière technique, appelée CDMA, est la plus récente. Elle est basée sur l'étalement du spectre (utilisation de DSSS) : la largeur de l'occupation spectrale du signal est multipliée par un gain de codage. Cet élargissement améliore la réception du signal, autorisant sous certaines conditions la cohabitation de plusieurs signaux sur la même bande de fréquences. Le partage de l'accès au médium de communication se fait donc suivant un procédé de codage : à chaque nœud correspond un code (ou clé) à l'aide duquel le message est codé avant d'être émis.

Parmi les inconvénients possibles des protocoles par accès contrôlé il y le problème de savoir qui définit l'allocation et le fait d'avoir parfois besoin de synchronisations. Cependant, l'approche CDMA présente des avantages (résistance aux interférences, faible consommation, etc.) qui font qu'elle est relativement en vogue dans les systèmes de téléphonie type GSM et UMTS. De fait, les systèmes de téléphonie actuels reposent sur des stations de base qui assurent la synchronisation des nœuds (affectation des codes). Dans le cadre des réseaux ad hoc, l'absence d'infrastructure, la dynamicité de la topologie du réseau et son auto-organisation au déploiement rendent l'utilisation du CDMA nettement plus problématique.

- **Protocoles à accès aléatoire**

  Dans un protocole à accès aléatoire, un nœud doit pouvoir envoyer un paquet de données en disposant complètement du canal et sans coordination préalable avec les autres nœuds. L'absence de coordination aboutit à la possibilité d'avoir des collisions (nœuds émettants en même temps). Résoudre ce problème nécessite de savoir d'une part détecter une collision, d'autre part de savoir la gérer.
  Historiquement, le premier protocole à accès aléatoire est ALOHA. Dans ce protocole, tout nœud ayant procédé à une émission (d'un paquet de données) qui s'est traduite par une collision doit attendre un temps tiré aléatoirement avant de réémettre le paquet concerné. Cette gestion sommaire ne permet évidemment pas d'obtenir des performances satisfaisantes sur des réseaux de grande taille. Une première amélioration, dénommée CSMA (pour *Carrier Sense Multiple Access*), a été l'ajout avant toute émission d'une phase d'écoute du support de communication pour savoir si il est libre. Cela permet de réduire les collisions, mais pas celles provenant de nœuds émettant en même temps.
  Aussi, pour que la gestion des collisions ne soit plus prise en charge par une couche de niveau supérieur, comme dans ALOHA, un mécanisme de détection de collision a été ajouté. Le résultat est un protocole de type CSMA/CD (*Collision Detection*), le plus connu d'entre eux étant le protocole Ethernet (IEEE 802.3). Néanmoins, le CSMA/CD suppose qu'un nœud puisse écouter et émettre en même temps, or dans les réseaux sans fil cela n'est pas possible. D'autre part, un nœud qui écoute peut ne pas capter des transmissions que reçoit un de ses voisins (problème du nœud caché, cf. figure 1.6). Finalement, l'adaptation du CSMA/CD aux contraintes du sans fil a abouti au CSMA/CA (*Collision Avoidance*). Comme son nom l'indique, ce type de protocole intègre des mécanismes qui cherchent à éviter les collisions :
  – accès au médium et reprise sur collision utilisant des temporisateurs ;
  – sécurisation des communications point à point par acquittement positif et réservation optionnelle du canal par une « poignée de main » avant chaque transmission. Ce dernier mécanisme permet également d'éviter les collisions avec les nœuds cachés.

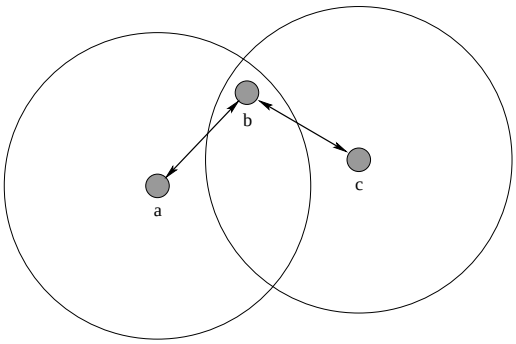


FIG. 1.6 – Problème du nœud caché : a ne sait pas que b peut communiquer avec c, et réciproquement.

Divers protocoles MAC ont été proposés pour les réseaux ad hoc, dont on rappelle que les réseaux de capteurs peuvent être vus comme un cas particulier. Néanmoins, les spécificités d'un réseau de capteurs, évoquées précédemment dans la section 1.3, font que la définition d'un protocole MAC adéquat est un domaine de recherche très actif. En effet, les protocoles MAC pour réseaux ad hoc, en particulier pour les MANETs (*Mobile Ad hoc NETworks*) qui sont les plus proches des réseaux de capteurs, se caractérisent par :

- la qualité de service (*QoS*) comme principal objectif (la garantie d'un débit minimal pour les applications) ;
- la prise en compte de la consommation énergétique comme n'étant que secondaire, puisqu'a priori la source d'énergie d'un nœud (ordinateur portable, PDA, etc.) peut être remplacée ;
- la gestion de réseaux de taille moindre ;
- etc.

Dans le cas d'un réseau de capteurs, le protocole MAC doit mettre l'accent sur :

- la maîtrise de l'énergie consommée ;
- la tolérance aux pannes successives à la défaillance de certains nœuds ;
- un accès équitable et efficace de tout nœud au support de communication.

Pour ce faire, certains protocoles utilisent des modes de fonctionnement économes en énergie tels que l'alternance de périodes de veille et d'activité, recourant à un schéma de type TDMA. Une autre approche est l'usage de l'acquittement négatif (s'appuyant sur des temporisateurs) pour détecter les erreurs de transmission.

Dans les articles de synthèse [28] et [29], les auteurs présentent divers protocoles MAC pour réseaux de capteurs. Les premiers protocoles tels que S-MAC [30] étaient surtout orientés conservation d'énergie. Par la suite, comme le temps de réponse d'un réseau de capteurs est capital dans certaines applications, ce fut le meilleur compromis entre énergie consommée et le couple débit/latence des communications [31] qui était recherché. Nous verrons plus en détails certains protocoles MAC pour réseaux de capteurs dans le chapitre suivant, où l'on étudiera les solutions proposées à différents niveaux de l'architecture réseau pour une meilleure gestion de la consommation énergétique.

1.5.3 Couche réseau

- **Auto-organisation**

  L'absence d'infrastructure (préservation de l'énergie) et le déploiement (voire redéploiement) aléatoire de nœuds capteurs posent le problème de la construction de la topologie du réseau et de son maintien dans le temps. D'autant plus si le réseau est dense et comporte beaucoup de nœuds. Les nœuds doivent donc s'organiser pour faire émerger un schéma de coopération pertinent et adéquat pour atteindre l'objectif du réseau. Cette organisation, résultat de la phase d'auto-organisation, doit être produite sans aucun contrôle, ni connaissance préalable. L'auto-organisation peut notamment consister à créer une topologie réseau, à former des clusters de nœuds ou encore à affecter un identifiant unique à chaque nœud. L'objectif de l'auto-organisation dans les réseaux de capteurs est d'économiser de l'énergie. Bien entendu, la phase d'auto-organisation a elle-même un coût énergétique, mais le bénéfice énergétique attendu doit en principe largement compenser cette surconsommation.
  La formation de clusters consiste à partitionner un réseau en construisant des groupes de nœuds géographiquement proches, avec chacun à sa tête un nœud « chef » (*clusterhead*). On distingue d'une part des algorithmes de formation actifs, d'autre part des algorithmes passifs. Dans le premier cas, les clusters sont construits en échangeant des informations spécifiques entre nœuds, alors que dans le second cas c'est le trafic réseau qui est utilisé. Divers algorithmes de formation de clusters ont été proposés, ont peut citer ACE [32] ou encore [33]. La structuration en clusters et le routage sont souvent liés. C'est par exemple le cas de l'algorithme de routage LEACH [34]. Celui-ci s'appuie sur l'organisation hiérarchique associée à la structuration en clusters pour minimiser les communications. En effet, les nœuds d'un même cluster routent les données vers le chef, à charge pour ce dernier de les transmettre au nœud collecteur. Comme un chef de cluster va consommer plus d'énergie que les autres nœuds, les auteurs proposent que tout nœud devienne à un moment ou à un autre responsable du cluster. Le choix du nœud chef se fait alors de manière probabiliste, avec la probabilité associée à chaque nœud qui augmente avec le temps passé depuis la dernière fois où il a été élu chef du cluster.
  Pour ce qui est de l'auto-adressage, de nombreux algorithmes ont été développés, ceux-ci peuvent être classés en deux catégories [35]. La différence entre les deux catégories est l'utilisation ou non d'une table d'allocation d'adresses. Les algorithmes n'utilisant pas de table d'allocation sont décentralisés, ce qui est une caractéristique très intéressante dans le cadre des réseaux de capteurs. Le principe de cette catégorie d'algorithme est de générer aléatoirement une adresse, puis de détecter d'éventuelles collisions d'adresses. La différence entre les algorithmes se situe alors au niveau du mécanisme de détection des collisions. Néanmoins, il faut noter que dans beaucoup de réseaux de capteurs il est difficile d'adresser de manière unique chaque nœud.

- **Routage**

  Dans tout réseau, l'acheminement des données d'un point à un autre se fait par l'intermédiaire d'un algorithme de routage. L'objectif de cet algorithme est de déterminer un chemin optimal par rapport à un critère de performance bien précis (par exemple, la longueur moyenne des routes). Pour ce faire, il doit tenir compte des caractéristiques du réseau considéré et plus particulièrement des contraintes inhérentes à la nature du médium de communication. Par exemple, dans les réseaux sans fil la portée et les interférences sont des contraintes de tout premier plan.
  Beaucoup d'algorithmes / protocoles de routage ont été proposés dans le cadre des réseaux ad hoc. Ceux-ci peuvent être classés suivant différents critères [36] : nature des informations de routage échangées, approche centralisée ou distribuée, quand et comment les routes sont calculées, etc. Ainsi, si on considère le dernier critère (le calcul des routes), on distingue :
  – le routage pro-actif, qui consiste à calculer à l'avance les routes en échangeant périodiquement des paquets de contrôle. Dans ce cas, à chaque nœud est associée une table de routage ;
  – le routage réactif, caractérisé par le calcul à la demande des routes, ce qui permet de minimiser le trafic réseau.
  Si on compare les deux approches, on constate que le routage pro-actif permet d'avoir des routes immédiatement disponibles, mais au prix d'un trafic de contrôle important. Dans le cas du routage réactif la découverte des routes se fait souvent par inondation (chaque nœud diffuse les informations de routage à tous ses voisins). Notons qu'il est également possible de diffuser les données par inondation, on parle alors de routage par inondation. Néanmoins, comme le trafic réseau associé est très important, on gâche beaucoup de bande passante et d'énergie. Enfin, routages pro-actif et réactif ne sont pas très adaptés à des réseaux de très grande taille (le trafic de contrôle devient trop important de même que les tables de routage). Aussi, dans ce contexte un routage hybride, s'appuyant sur une structuration en clusters du réseau, est souvent adopté : le routage est pro-actif au sein des clusters, tandis que le routage est réactif entre les clusters.
  Dans le cadre des réseaux de capteurs, les spécificités de ces derniers doivent être prises en compte dans le routage :
  – envoi en continu ou à la demande (requête) ;
  – capacité de stockage et en énergie limitées ;
  – adressage problématique des nœuds.
  Les articles [37] et [38] font un état de l'art très complet des techniques de routage pour les réseaux de capteurs. Il s'agit de techniques spécifiques ou définies pour les réseaux ad hoc, notamment mobiles, et répondant aux contraintes des réseaux de capteurs. Tout deux distinguent 4 familles de protocoles de routage :

  1. routage orienté données ou à plat (ou direct) ;
  2. routage hiérarchique ;

3. routage géographique ;
4. routage orienté flux de données et qualité de service.

Il faut cependant souligner que certains rares protocoles ne sont pas classés par les deux articles dans la même famille. Enfin, dans [37] certains des protocoles de la quatrième famille peuvent être également vus comme faisant partie d'une des trois familles précédentes.

Dans le routage orienté données ou à plat, les nœuds capteurs répondent à des requêtes de demande de données. Le réseau de capteurs peut donc être vu comme une base de données. Pendant l'envoi de la requête, des informations sur le chemin suivi sont stockées dans les nœuds. Aucun n'adressage n'est utilisé, en revanche les données sont décrites par des attributs, de sorte que seuls les nœuds possédant la donnée requise répondent. Parmi les nombreux algorithmes de cette famille on peut nommer SPIN [39] et Directed Diffusion [40].

Le routage hiérarchique, qui on le rappelle repose sur la structuration en clusters, est une solution au défi que représente le passage à l'échelle. De fait, l'utilisation de clusters permet de réduire la consommation énergétique et le nombre de paquets de données circulant dans le réseau. D'une part les communications sont dans un premier temps limitées à chaque cluster, puisque toutes les données sont d'abord envoyées au chef du cluster. D'autre part, celui-ci va dans un second temps réduire les paquets en agrégeant et fusionnant les données, avant de les transmettre. LEACH [34], déjà évoqué précédemment, PEGASIS ou TEEN [41] et APTEEN [42] sont des exemples d'algorithmes de routage de ce type.

Le routage géographique s'appuie lui sur l'information de localisation de chaque nœud. Plus précisément, à partir de ce type d'information il est possible de calculer la distance entre deux nœuds et donc d'estimer le coût énergétique de la communication associée. L'objectif du routage est alors de trouver le meilleur chemin, au sens de l'énergie consommée, entre le nœud source et la destination. L'algorithme de routage géographique GAF [43] (*Geographic Adaptive Fidelety*), qui a été proposé pour les réseaux ad hoc mobiles (MANETs), utilise par exemple la position GPS.

Dans la dernière famille de méthodes de routage, celles orientées flux de données et qualité de service, on trouve des approches modélisant le routage comme un problème de flot ou cherchant à obtenir le meilleur compromis entre l'énergie consommée et le délai de transit (*end-to-end delay*).

Chapitre 2

La problématique de la durée de vie d'un réseau de capteurs

2.1 Introduction

Les réseaux de capteurs sont caractérisés par l'absence d'infrastructure globale, chaque nœud (éventuellement mobile) ayant une provision d'énergie limitée sans aucune possibilité de disposer de ressources énergétiques supplémentaires. Aussi, la durée de vie d'un nœud est fortement liée à la manière dont l'énergie est exploitée par ses composants logiciels et matériels, i.e. comment cette énergie est dépensée. Une utilisation raisonnée et adéquate de l'énergie est donc cruciale pour maintenir en activité aussi longtemps que possible chaque nœud capteur et par voie de conséquence le réseau de capteurs. Comme l'exploitation et la gestion de l'énergie dans les réseaux de capteurs, qu'elle soit au niveau des algorithmes / protocoles mis en œuvre ou des composants physiques (processeur, mémoire, etc.), est une problématique majeur, elle fait l'objet de nombreux travaux de recherches. L'objectif de toutes ces études est d'augmenter la durée de vie des nœuds constituant le réseau considéré.

L'énergie n'est pas consommée de façon uniforme dans le temps, dans le sens où les nœuds d'un réseau ne disposent pas tous de la même quantité d'énergie à un instant donné. De fait, d'une part les nœuds sont généralement hétérogènes et d'autre part un nœud peut tenir différents rôles durant son existence au sein du réseau. Lors du déploiement, la répartition aléatoire des nœuds dans l'espace fait que certains nœuds sont plus actifs que d'autres, d'où une consommation variant d'un nœud à un autre. Les différents paramètres d'un réseau de capteurs : topologie, emplacement et rôles de chaque nœud, le mode de routage des données, la stratégie d'accès au médium de communication, etc., ont tous un impact sur la consommation de l'énergie. Il est donc indispensable de disposer pour chaque paramètre d'une approche permettant de maîtriser la consommation énergétique.

Dans ce chapitre, nous allons tout d'abord présenter quelques définitions possibles pour la notion de durée de vie d'un réseau. Ensuite, nous décrirons

quelques approches, à différents niveaux du modèle OSI, optimisant la consommation énergétique.

2.2 La notion de durée de vie d'un réseau

Il est important de définir de manière précise ce que l'on entend par durée de vie d'un réseau de capteurs. Dans la littérature, il existe différentes définitions possibles, selon des propriétés et des contraintes diverses liées au réseau que l'on désire étudier. Ainsi, le réseau est supposé avoir atteint sa fin de vie dès lors :

- qu'un nœud du réseau est défaillant. La durée de vie de ce nœud est donc également celle du réseau de capteurs [44, 45].
- que le pourcentage de nœuds encore actifs est inférieur à un seuil fixé [46, 47] ;
- que le réseau de capteurs ne couvre plus complètement la zone d'intérêt [43, 48, 49] (présence d'une zone où le phénomène physique considéré ne pourra plus être observé).

Suivant la configuration du réseau, la perte d'un nœud est suffisante pour provoquer son dysfonctionnement, dans d'autres cas le réseau pourra tolérer plusieurs pertes de nœuds. Par dysfonctionnement, on entend surtout le partitionnement du réseau en composantes connexes distinctes.

Pour notre part, nous considérerons la durée de vie d'un réseau de capteur selon la première définition, à savoir comme étant égale à celle du premier nœud défaillant.

2.2.1 Un nœud défaillant

Cette première approche consiste à considérer le réseau en vie et donc opérationnel, tant que tous les nœuds disposent de suffisamment d'énergie pour exécuter les tâches qui leurs restent à accomplir. Elle est souvent utilisée dans le cadre de réseaux présentant une consommation énergétique homogène pour tous les nœuds (ceux-ci meurent quasiment en même temps). Dans cette approche, il n'est pas nécessaire de considérer l'impact de la perte d'un nœud sur la connectivité du réseau. Par exemple, il n'y aucune phase de reconstruction des tables de routage.

2.2.2 Un nombre de nœuds actifs insuffisant

Une alternative à l'approche précédente est de considérer un réseau comme actif tant qu'une certaine proportion de ses nœuds est encore en fonctionnement. De plus, tous les nœuds ayant encore de l'énergie doivent faire partie d'une seule et même composante connexe. Dans ce contexte, la disparition d'un nœud doit être intégrée. Il est impératif de redéfinir les tables de routage en tenant compte uniquement des nœuds encore actifs. Une phase de réorganisation / auto-organisation est donc indispensable. Dans ce cas de figure, il est possible de continuer à exploiter le réseau de capteurs. Comme le montre le figure 2.1, l'évolution du réseau peut être vue comme une répétition de périodes d'auto-organisation et d'activité, plus une phase de détection des nœuds manquants ou bien d'ajout de nœuds.

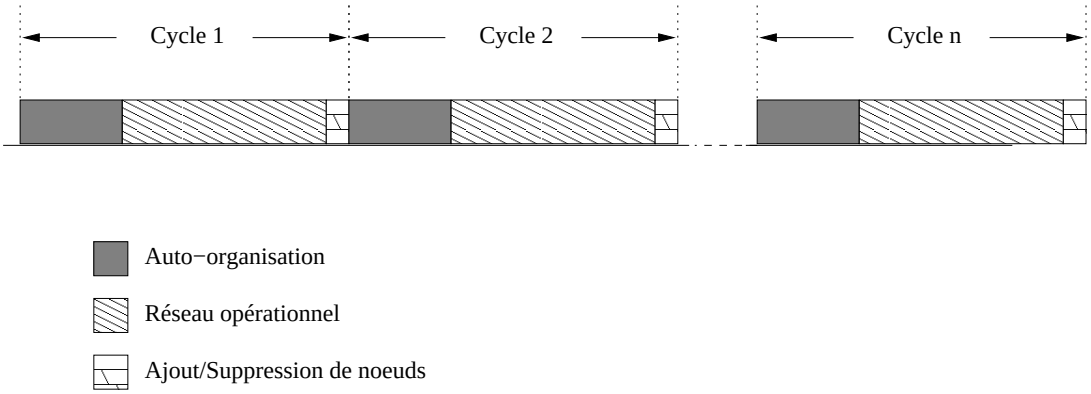


FIG. 2.1 – Les différentes phases rythmant la vie du réseau.

2.3 Optimisation des aspects énergétiques

2.3.1 Couche physique

Au niveau de la couche physique, qui est essentiellement du domaine de l'électronique, la recherche de la meilleure performance énergétique est un défi majeur pour les fondeurs de composants. L'autre axe de recherche important est le développement de nouvelles batteries.

- **Composants électroniques**

  Les fondeurs de CPU, tel qu'Intel ou IBM, cherchent à produire des composants offrant le meilleur rapport puissance / Watt consommé [50]. Pour ce faire, il s'agit d'une part de contrôler la tension d'alimentation et la fréquence de fonctionnement [51], d'autre part d'utiliser des techniques de gravure induisant moins de pertes d'énergie. Par exemple, les architectures dites EESA (*Energy Efficient System Architecture*) d'Intel font passer les parties inactives (n'ayant aucun traitement en cours) du composant dans un état de basse consommation [52]. Au niveau des techniques de gravure, les chercheurs d'IBM se concentrent sur la lithographie par faisceaux d'électrons [53, 54], une technique comparable à l'actuelle lithographie optique, mais induisant une consommation en énergie inférieure.

  Au niveau des mémoires, on peut citer le développement par Ramtron et Texas Instrument du premier prototype de mémoire de 512 Ko FRAM (ferroe-lectric RAM) [55, 56] d'une finesse de gravure de 130 nm. La FRAM est une mémoire non volatile dont la construction reste proche de celle des DRAM. On retrouve donc le design 1T-1C de la DRAM, chaque cellule étant composée d'un transistor et d'un condensateur qui retient le courant. À cette différence que le condensateur d'une FRAM utilise des matériaux ferroélectriques pour retenir les données. Il prend généralement la forme d'un film en céramique de PZT (Titano-Zirconiate de Plomb). Ce type de mémoire est donc simple à

fabriquer, mais moins dense que ses concurrentes. Sa non-volatilité vient du fait que contrairement à la DRAM qui connaît des fuites et doit donc être rafraîchi plusieurs fois par seconde, demandant donc constamment de l'énergie, le condensateur de la FeRAM n'a pas ce problème. Il ne requiert de l'énergie que pour la lecture ou l'écriture de données.

- **Batteries**

  Ces dernières années, le gain en autonomie des appareils nomades a été surtout le fait des gains énergétiques obtenus au niveau des composants. Du côté des batteries, les évolutions sont plus lentes à se dessiner et à se mettre en place. À l'heure actuelle, on distingue quatre technologies de batteries (la plus récente est encore à l'état de prototype) :

  1. **Nickel Métal-hydrure (NiMh)** [57]

     Technologie récente qui s'inscrit parmi les plus utilisées dans les appareils mobiles. Cela se justifie par l'excellent rapport prix / durée de vie que présente ce type de batteries. Néanmoins, elles présentent l'inconvénient d'être fragiles, sensibles à la surcharge et de requérir des chargeurs spéciaux.

  2. **Lithium-ion (Li-ion)** [58]

     Cette technologie de batteries offre de meilleures performances que la précédente. Une batterie de ce type n'est pas sensible au problème d'effet mémoire et peut être rechargée plusieurs fois.

  3. **Lithium-ion polymère (Li-poly)** [59]

     Apparues en 1999, les batteries Lithium ion polymère sont une variante de la technologie Lithium ion. Les performances sont sensiblement les mêmes, mais l'électrolyte est remplacé par un polymère gélifié qui permet de donner toutes les formes possibles à la batterie.

  4. **La pile à combustible** [60]

     C'est a priori la technologie de batteries du futur. En effet, en produisant du courant électrique à partir de l'hydrogène, ce type de batterie (ou pile) atteint des performances bien supérieures aux batteries classiques et permet des usages tout à fait différents : plus besoin de brancher la batterie sur le secteur pour la recharger, il suffit de la charger en utilisant des plaquettes solides, plus sûres et plus pratiques à manipuler. Mis à part les piles à combustible à l'hydrogène, il existe également des piles à combustible au méthanol. Néanmoins, il faut souligner que cette technologie n'existe qu'à l'état de prototype.

  Les articles [61] et [62] présentent deux modélisations mathématiques d'une batterie, celles-ci ont été proposées pour étudier son comportement dans le temps. Les auteurs s'intéressent plus particulièrement aux modèles sous-jacents de décharge de la puissance, afin d'en déduire des métriques pour estimer la durée de vie de dispositifs micro-électroniques.

Les réseaux de capteurs intègrent un nombre important de nœuds, souvent hétérogènes. La consommation d'énergie est de plus en plus difficile à maîtriser du fait que la réduction de la tension d'alimentation conduit à réduire également la fréquence et donc les performances [63], cela est dû principalement aux différents états que peut prendre un nœud capteur tout au long de sa durée de vie dans le réseau [64]. Par conséquent, pour diminuer la consommation d'énergie, il devient primordial de gérer dynamiquement la tension et la fréquence au plus près de l'activité des nœuds du réseau considéré [65]. Certaines études ont permis de modéliser un nœud capteur et d'avoir des heuristiques qui permettent de prédire les performances liées à la consommation d'énergie, avec des modèles qui permettent d'évaluer la consommation énergétique au niveau d'un nœud capteur durant les différentes phases de sa vie au sein du réseau soit : à l'initialisation, lors d'éventuelles attentes passives, d'acquisitions de données, de réceptions de paquets de données, de transmissions de paquets de données. La table 2.2 donne la durée de vie estimée d'un nœud capteur en fonction de l'état de son interface radio.

Durée de vie d'un noeud capteur		
Etat de l'interface de transmission	Taux d'utilisation	Espérance de vie de la batterie
Ecoute à plein temps	100%	3 Jours
Ecoute active en mode économie énergie	100%	6.65 Jours
Ecoute périodique	10%	65 Jours
Mode veille non actif	0.01%	Plusieurs années

FIG. 2.2 – Durée de vie d'un nœud suivant le mode d'écoute.

Les valeurs présentées dans le tableau 2.3 correspondent aux consommations énergétiques de l'interface de communication de type TR1000 de RF-Monolithics, qui a une portée de transmission d'une vingtaine de mètres [66, 67]. Ce dispositif a un débit de transfert de données de l'ordre de $2,4$ Kbps et utilise une modulation de type *On-Off Keying* (OOK) [68]. Ces valeurs mettent en évidence une importante consommation d'énergie lorsque l'interface radio d'un nœud capteur est active [69].

Mode Com.	Consommations (mW)
Emission	14.88
Réception	12.50
Idle	12.36
Etteint	0.016

FIG. 2.3 – Consommation énergétique associée à l'interface radio TR1000 selon son mode.

2.3.2 Couche liaison de données

Nous nous intéressons plus particulièrement à la sous-couche MAC. De nombreux travaux ont été menés afin d'améliorer les performances énergétiques de la sous-couche MAC, certains correspondant à la transposition de protocoles pour réseaux ad hoc dans le contexte des réseaux de capteurs. Une adaptation pas évidente, étant donné les contraintes matérielles qui différencient les deux types de réseaux.

Le TDMA tel que décrit dans [70] est le plus classique des modes de gestion de l'accès au médium de communication . Il présente l'avantage d'être en mesure d'être adopté par les réseaux de capteurs à topologies fixes. L'envoi de données entre nœuds voisins est possible en exclusion mutuelle, la gestion du médium de communication est assurée par un ordonnanceur d'états (*scheduler*) qui prend en compte plusieurs contraintes, en étant fortement dépendant de l'orientation de l'application envisagée. De nouvelles variantes distribuées du TDMA ont été proposées, moins coûteuses en énergie [71] ou encore avec prise en charge des collisions [72, 73]. En effet, grâce à l'ordonnanceur des activités, les nœuds se mettent en mode veille durant leurs périodes d'inactivités, ce qui abouti à une diminution considérable de la consommation d'énergie [74].

Le S-MAC (*Sensor-MAC*) [30] est un mécanisme de gestion de l'accès au médium de communication pour réseaux de capteurs. Il gère l'interface radio dans le but de maximiser la durée de vie des nœuds. Le S-MAC impose des périodes de mise en veille aux nœuds afin de pallier au problème de l'écoute du canal qui est coûteuse en énergie. Les périodes de mise en veille sont fixes pour tous les nœuds, l'énergie est donc uniformément consommée entre les nœuds du réseau et la dynamique du réseau est trivialement imposée. Les nœuds sont actifs durant de courtes périodes sans tenir compte du trafic et de l'hétérogénéité des taux d'activités entre les nœuds. S-MAC conserve mieux l'énergie que le standard IEEE 802.11, ainsi les nœuds survivent plus longtemps.

Le T-MAC (*Timeout-MAC*) [75] est dérivé du S-MAC, il apporte des améliorations pour de meilleures performances au niveau de l'énergie consommée. En effet, S-MAC impose une période de mise en veille identique pour tous les nœuds, ce qui pénalise considérablement le débit pour les réseaux ayant un trafic de données important et induit aussi une perte d'énergie considérable en augmentant les périodes d'activités. Le S-MAC est donc gourmand en ressources énergétiques dans le cas des réseaux à fort trafic de données. Le T-MAC améliore le S-MAC en utilisant une stratégie adaptative de mise en veille dédiée, i.e. la durée des périodes de mise en veille n'est pas commune à tous les nœuds du réseau. L'idée est de mettre un nœud en veille si la durée de sa période d'inactivité est supérieure à un seuil. S-MAC et T-MAC ont les mêmes performances dans un environnement réseau à trafic constant, par contre T-MAC induit des performances nettement supérieures pour ce qui est de la maîtrise de la consommation énergétique dans un environnement à trafic réseau variable.

Le P-MAC (*Pattern MAC*) [76] étend les mécanismes S/T-MAC et pallie les problèmes sous-jacents en ajustant dynamiquement la longueur de la période de

réveil des nœuds selon les activités environnantes. En d'autres mots, les périodes de veille d'un nœud dépendent non seulement de son propre trafic mais aussi de celui de ses nœuds voisins. En effet, la politique de mise en veille en T-MAC est fortement pénalisante, car elle met prématurément des nœuds en état de veille. Le P-MAC permet de réduire la consommation de l'énergie suite à l'écoute passive du canal. Des motifs des activités sont générés localement au niveau d'un nœud et sont échangés avec les nœuds voisins. Malgré sa consommation en énergie due aux échanges de motifs, P-MAC fournit de meilleures performances que les mécanismes S/T-MAC. Des performances qui traduisent une conservation énergétique plus intéressante dans le cas des réseaux à trafic variable.

Dans [77], les auteurs ont développé un ordonnanceur pour réseau de capteurs à communications asymétriques selon l'architecture AROS (*Asymetric communication and ROuting in Sensor networks*) [78]. Dans [79], les auteurs ont étudié le compromis entre la consommation d'énergie et des performances du réseau comme le débit et les délais d'attente pour les protocoles de type RASMAC (*Route-Aware Sensor MAC*) [79]. Les deux modes basés sur la contention et sur l'ordonnancement fournissent un ensemble riche de schémas de fonctionnement basés sur la préservation des ressources énergétiques.

Le CDMA (*Code-Division Multiple Access*) [80] tient compte des envois multiples sans pertes de messages et sans risques de collisions, grâce à l'introduction de codes (Hadamard [81]). Ce mode d'exploitation est plus gourmand en terme d'énergie mais présente des avantages que son prédécesseur TDMA n'offre pas, notamment dans les réseaux de capteurs à Qualité de Service requise ou temps réel. Il a été développé pour la téléphonie sans fil mais s'adapte pour les réseaux de capteurs. Des travaux d'adaptation ont été réalisés dans [82], en tenant compte des aspects énergétiques qui ne sont pas aussi influants que dans le contexte des réseaux ad hoc. L'approche décrite dans [83], solutionne le problème du nœud proche-distant rencontré dans le modèle CDMA, problème ayant un impact non négligeable sur la consommation d'énergie.

Le CSMAC [31] (*Cdma Sensor MAC*) proposé par Bulusu *et al.* est un protocole MAC qui est issu a la fois de la technique de modulation du TDMA et de celle du CDMA [84], fusionnant les avantages des deux approches tout en l'orientant optimisation d'énergie et absence de collision. Contrairement aux protocoles MAC précédents, comme le S-MAC qui est orienté conservation d'énergie plutôt que latence des communications [85, 30], le protocole CSMAC est conçu selon un compromis entre la consommation énergétique, la latence, l'exactitude et la tolérance aux pannes.

2.3.3 Couche réseau

Durant la phase d'auto-organisation du réseau, le processus d'établissement des routes est biaisé par des considérations énergétiques. Lors de la transmission de données le coût énergétique est proportionnel au carré de la distance qui sépare les nœuds communiquants. De plus, la puissance d'envoi doit être multipliée par un facteur lors de la présence d'obstacles entre les nœuds. De ce fait, les communications multi-sauts sont moins coûteuses que des communications directes distantes. En fait, le routage direct de données serait le plus adapté si tous les nœuds étaient très près du / des nœud(s) collecteur(s) [86]. Cependant, la plupart du temps les nœuds capteurs sont dispersés aléatoirement, dans ce cas le cheminement des données est inévitablement de type multi-sauts. Dans le chapitre précédent nous avons vu que les protocoles de la couche réseau pouvaient être classés en quatre catégories (cf. section 1.5.3). Dans la suite, nous allons présenter quelques protocoles, notamment orientés données, en insistant sur la façon dont ils appréhendent la consommation d'énergie.

Les données sont habituellement transmises avec une redondance significative. Cette redondance de données se traduit par une perte énergétique relativement considérable, aussi les algorithmes de routage de données doivent être en mesure de prendre en charge l'agrégation des données avant de les acheminer. Le collecteur envoie des requêtes à destination des nœuds et reçoit les données observées des régions ciblées. En considérant les types des requêtes formulées qui s'avèrent être dédiées, des requêtes basées sur des attributs sont inévitables afin d'indiquer les propriétés des données.

SPIN (*Sensor Protocols for Information via Negotiation*) [39] est le premier protocole orienté données. Il consiste à établir un échange de requêtes-interrogations entre nœuds dans le but d'éliminer les données redondantes significatives, permettant une nette amélioration de la consommation d'énergie. Dans [40] une variante basée sur la diffusion directe a été développée. D'autres protocoles basés sur la diffusion directe ont été ensuite proposés et développés [87, 88, 89] ou basés sur les mêmes concepts tels que *COUGAR* [90] ou *ACQUIRE* [91]. Nous allons présenter quelques uns de ces protocoles de façon détaillée afin de souligner l'apport en terme de conservation d'énergie.

Dans SPIN, quand un nœud i a des données à envoyer à un voisin j, il va nommer ses données en utilisant des descripteurs de haut niveau. Avant transmission, les descripteurs sont échangés entre les nœuds capteurs par un mécanisme de publication des données. Ainsi i annonce ses données à j. Si j n'a jamais reçu les données, il répond par un acquittement positif, dès lors i peut les transmettre. j annonce à son tour les données qu'il a à transmettre et ainsi de suite. Ce protocole permet donc d'économiser de l'énergie, car s'assurer que les données n'ont pas été transmises est beaucoup moins coûteux que de les envoyer à perte. De ce fait, SPIN est un protocole de routage de données qui prend en charge une gestion efficace des ressources énergétiques. Un des avantages de SPIN est que les changements de topologie sont localisés puisque chaque nœud doit connaître seulement ses voisins. En terme de gain énergétique, SPIN consomme 3,5 fois moins d'énergie que les méthodes par inondation (*flooding*) [92].

Le *flooding* et le *gossiping* [93] sont deux mécanismes classiques de relais de données pour réseaux de capteurs. Ils ne requièrent ni la mise en place d'une stratégie de routage, ni la maintenance de la topologie. Dans le *flooding*, chaque nœud capteur reçoit une trame de données et la diffuse dans son voisinage, ce phénomène de propagation par diffusion assure l'acheminement jusqu'au nœud cible. Le *gossiping* est une version améliorée, la différence étant qu'à la réception d'une trame de données celle-ci est uniquement diffusée vers quelques nœuds choisis aléatoirement dans le voisinage. Le phénomène de propagation est donc légèrement différent de celui du *flooding* et consomme relativement moins d'énergie. Bien que le *flooding* soit simple à implanter, il présente deux inconvénients [39] :

- un problème d'implosion dû à la duplication des données envoyées ;
- un problème de recouvrement (deux nœuds captant dans une même région, envoient des paquets de données identiques, ce qui se traduit par une forte consommation d'énergie).

Le *gossiping* pallie au problème d'implosion par la sélection des nœuds destination.

La diffusion dirigée (*Directed Diffusion*) proposée dans [94, 95], suggère l'utilisation de paires attribut-valeur pour les données et les requêtes. Les nœuds ont également la capacité de faire l'agrégation de données. Dans [90] un récapitulatif des protocoles à diffusion dirigée est présenté en détail. Les réparations de chemin sont également possibles dans la diffusion dirigée. Quand un chemin entre un nœud et le nœud collecteur est coupé, un nouveau chemin alternatif est construit. Ganesan *et al.* proposent dans [96] l'utilisation de chemins multiples construits à l'avance, de telle sorte qu'en cas d'échec d'un chemin, une route alternative est choisie sans coût énergétique additionnel.

Dans COUGAR, le réseau est vu comme une base de données répartie [90]. L'idée est d'employer des requêtes déclaratives afin de soustraire le traitement des requêtes des fonctions de la couche réseau : le choix des nœuds capteurs appropriés, etc. Il utilise également l'agrégation de données pour économiser de l'énergie. ACQUIRE (*ACtive QUery forwarding In sensoR nEtworks*) [91] est un protocole similaire. Différentes modélisations mathématiques ont été définies pour ce dernier [87, 89] afin de prédire sont coût énergétique.

Le principal but du routage hiérarchique est de maîtriser la consommation d'énergie des nœuds capteurs en imposant des communications multi-sauts au sein d'un cluster (groupe de nœuds). L'agrégation et la fusion de données sont également adoptées afin de diminuer le nombre de messages transmis au nœud collecteur. Quant à la formation des clusters, elle est basée sur la réservation d'énergie [97, 98]. LEACH [86] est l'une des premières approches de routage hiérarchique pour réseaux de capteurs.

D'autres travaux de recherche ont été proposés pour améliorer du point de vue de l'énergie consommée des algorithmes de routage de données existants. Dans [99], trois approches ont été présentées. La première concerne le routage unicast, un mécanisme de routage qui s'adapte aux changements fréquents de topologie ainsi qu'à la faible puissance des batteries. Ce mécanisme, appelé *Energy Conserving Dynamic Source Routing* (EC-DSR), modifie le protocole de routage *Dynamic Source Routing* (DSR) [100] en considérant la stabilité des nœuds voisins. L'objectif étant

de fournir un routage multicast efficace, consommant moins de ressources. Un nouveau protocole de routage multicast, appelé *Source Routing-based Multicast Protocol* (SRMP) [101], utilise le concept de *source routing* pour minimiser la charge sur le réseau. SRMP permet une forte connectivité ainsi qu'une stabilité des liens entre les nœuds tout en minimisant la consommation d'énergie [102].

Chapitre 3

Environnements d'expérimentation

3.1 Introduction

Toute proposition d'un nouvel algorithme ou protocole doit être validée par des expérimentations. Dans le cas de travaux de recherche portant sur des problématiques concernant les réseaux de capteurs, deux approches sont possibles :

- déployer un vrai réseau de capteurs ;
- développer une simulation en utilisant un environnement parmi les nombreux qui sont disponibles.

Le déploiement d'un vrai réseau de capteurs est tout à fait envisageable, comme l'ont montré beaucoup de travaux [4, 10, 13]. Le principal avantage vient du fait qu'un vrai réseau permet d'appréhender de manière très fine l'environnement de fonctionnement (par exemple la perturbation des communications radio), celui-ci étant idéalisé dans une simulation. L'inconvénient majeur est que le choix d'un nœud capteur précis va fixer un certain nombre de caractéristiques physiques telles que les capacités de calcul, de stockage ou de transmission. La simulation permet elle de faire des tests en faisant varier plus facilement tous les paramètres possibles (taille du réseau, placement des nœuds, caractéristiques physiques, etc.). Une autre différence entre un vrai réseau et une simulation se situe sur le plan financier. En effet, le coût actuel d'un nœud capteur n'est pas du tout négligeable. Par exemple, un nœud capteur MICA2 [18] coûte plus de 150 $, quelle que soit la bande de fréquences choisie pour les communications (433 MHz ou autre). Idéalement, il faudrait tout d'abord passer par une phase de simulation pour faire de la prédiction de performances, étudier l'impact des différents paramètres du réseau de capteurs et les affiner. Puis, déployer un vrai réseau pour se confronter à la réalité du terrain.

Nous allons tout d'abord présenter quelques nœuds capteurs. En particulier quelques plateformes matérielles disponibles dans le cadre de la plateforme nationale CNRS réseaux de capteurs et réseaux auto-organisés (RECAP) [103]. La section qui suivra sera consacrée à la description de quelques environnements de simulation, notamment les plus connus à savoir NS-2 [104], GloMoSim [105] et OMNeT++ [106].

3.2 Les plateformes matérielles existantes

Comme le montre le tableau 3.1, de nombreuses sociétés commercialisent actuellement des nœuds capteurs. Ceux-ci sont de taille variable et adaptés à la mesure de diverses grandeurs physiques. Ce large panel de nœuds capteurs permet de répondre à un éventail étendu d'applications. Il est intéressant de noter qu'au niveau architecturale l'unité de capture est souvent un module distinct du reste du nœud capteurs. Cette approche permet aisément de faire évoluer les capacités d'observation du nœud capteur (grâce à l'interchangeabilité des unités de capture disponibles) et à moindre frais.

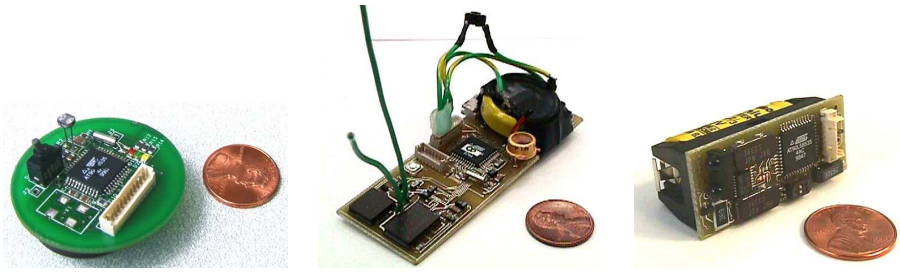
Société	Technologie	Applications visées
Crossbow Technology	Nœuds capteurs modulaires (WiFi et Zigbee)	Environnement, sécurité
Digital Sun http ://digitalsun.com	Nœuds capteurs statiques (Zigbee)	Contrôle d'un système d'arrosage automatique
Dust Network http ://www.dust-inc.com	Réseau maillé de capteurs (Zigbee)	Logistique, surveillance
Intel http ://www.intel.com	Nœuds capteurs modulaires (WiFi, Bluetooth)	Logistique, suivi de productions agricoles, suivi d'espèces animales
Millennial Net http ://www.millennial.net	Réseau maillé de capteurs (Zigbee)	Habitat, logistique
Senera http ://www.senera.com	Nœuds capteurs (vibrations, corrosion, ...)	Surveillance de ponts, tunnel et routes
...	...	...

TAB. 3.1 – Quelques plateformes matérielles.

Nous allons maintenant décrire plus en détails quelques nœuds capteurs de la famille des *Berkeley Motes*, ainsi que l'*Intel Mote* [10, 12]. Parmi les *Berkeley Motes* on s'intéresse plus particulièrement aux nœuds de type MICA [4, 18].

- **Berkeley Motes**

  Cette famille de nœuds capteurs a été conçu à l'Université de Californie à Berkeley, dans le cadre du projet *Smart Dust* financé par la *Defense Advanced Research Projects Agency* (DARPA), une agence du ministère de la défense des États-Unis. Ce projet a pris fin courant 2001 en ayant abouti à la conception de divers nœuds capteurs, la figure 3.1 en présente trois de type *COTS* (*Commercial Off-The-Shelf*). Les deux premiers nœuds présentés, WeC et RF, utilisent des communications radio de fréquence 916,5 MHz avec une portée de 20 mètres et un débit respectif de 5 et 10 Kbps. Tout deux intégraient des capteurs de luminosité et de température, plus d'autres capteurs pour le nœud RF (pression, accéléromètres, etc.). Quand au nœud IrDA, il se caractérise par l'utilisation d'un support de communication sans fil de type optique (technique infrarouge). Ces capteurs, relativement « anciens », ne sont naturellement plus d'actualités.



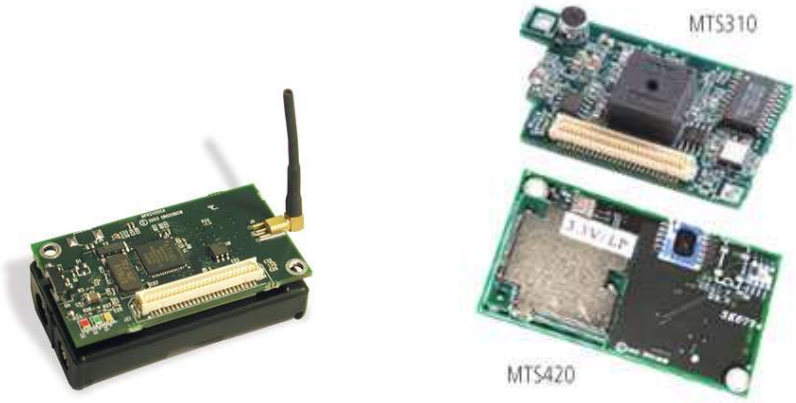
WeC *mote* RF *mote* IrDA *mote*

FIG. 3.1 – Quelques nœuds capteurs de type *COTS Dust* de l'UC Berkeley.

La génération actuelle des nœuds capteurs ayant succédés aux nœuds du projet *Smart Dust* est commercialisée par les sociétés Crossbow et Dust Networks (les sociétés proposent des nœuds capteurs distincts). Il s'agit notamment, pour la société Crossbow, de nœuds capteurs de la famille MICA : MICAz et MICA2. Les nœuds MICA2 et MICA2Dot datent de 2002, ils suivirent la première version commercialisée en 2001 (celle-ci correspondait à la quatrième génération de nœuds *Berkeley Motes*). L'année suivante fut introduit MICAz, enfin 2004 vit l'arrivée de la première version du nœud Telos. Nous évoquons plus précisément le nœud MICAz, car c'est celui-ci qui a été retenu pour constituer le réseau de capteurs de la plateforme nationale CNRS (RECAP). Différents laboratoires (le LIP6, le LIFL et le LAAS) disposent d'un tel réseau qui est composé de 28 nœuds MICAz, avec certains de ces nœuds qui peuvent jouer le rôle de nœud collecteur (en utilisant un module supplémentaire adéquat).
Une caractéristique intéressante des nœuds de type MICA est l'interchangeabilité de l'unité de capture, ce qui donne accès à une large gamme de capteurs.
Pour ce qui est des unités de traitement et de communication, le nœud présente les caractéristiques suivantes :

- Unité de traitement
  - processeur : Atmel ATMega 128L basse consommation ;
  - mémoire flash pour l'OS : 128 Ko ;
  - mémoire pour stocker les mesures : 512 Ko ;
  - EEPROM de configuration : 4 Ko ;
  - consommation de courant : 8 mA en activité, moins de 15 μA en veille ;
- Unité de communication (IEEE 802.15.4)
  - composant : Texas Instruments CC2420
  - bande de fréquences : 2,4 à 2,4834 GHz ;
  - sensibilité de réception : -90 dBm (min.), -94 dBm ;
  - débit : 250 Kbps ;
  - portée radio : en extérieur de 75 à 100 m, en intérieur de 20 à 30 m ;
  - consommation de courant : 19,7 mA en réception, de 11 à 17,4 mA en émission, 20 μA en inactivité et 1 μA en veille.

L'OS utilisé est TinyOS et l'énergie est tirée de 2 batteries AA. Sa taille est de $58 \times 32 \times 7$ mm pour un poids de 18 g (sans batteries). La figure 3.2 présente un nœud MICAz avec batteries et deux unités de capture.



MICAz MPR2400CA Unités de capture MTS310 et MTS420

FIG. 3.2 – Noeud MICAz 2,4GHz avec batteries et deux unités de capture.

- **Intel Mote**

  Le nœud capteur Imote2 (voir la figure 3.3), également commercialisé par Crossbow Technology, est le résultat d'une collaboration entre l'Université de Californie Berkeley et l'Intel Research Berkeley laboratory. Si on le compare au nœud précédent, on constate qu'il se différencie essentiellement par une unité de traitement plus puissante. L'augmentation des capacités de traitement permet d'envisager de traiter des données multimédia (traitement d'image, de sons ou de vidéos), surtout que le processeur intègre un coprocesseur MMX. Les caractéristiques de l'unité de traitement de l'Imote2 sont ainsi :
  - processeur : Intel PXA271 XScale 13-416 MHz basse consommation ;
  - mémoire flash : 32 Mo ;
  - SRAM : 256 Ko ;
  - SDRAM : 32 Mo ;
  - consommation de courant : de 31 à 66 mA en activité (suivant la fréquence de fonctionnement et l'activité), moins de 390 μA en veille ;

  L'unité de communication est quant à elle identique à celle du nœud MICAz, de même que l'OS. Concernant l'énergie, elle est puisée dans 3 batteries AAA. Sa taille est de $36 \times 48 \times 9$ mm pour un poids de 12 g (sans batteries).

3.3 Les outils de simulation

3.3.1 Critères d'évaluation

L'alternative au déploiement d'un réseau de capteurs en vrai grandeur consiste à utiliser un environnement de simulation. Afin de faciliter le choix d'un simulateur pour évaluer un algorithme / protocole pour réseau de capteurs, plusieurs aspects sont à considérer. En effet, plusieurs critères peuvent entrer en ligne de compte :

- précision de la modélisation ;

FIG. 3.3 – Nœud capteur Imote2 sans batteries.

- performance du moteur de simulation ;
- passage à l'échelle (simulation avec beaucoup de nœuds) ;
- facilité d'utilisation (prise en main, description de scénarios, automatisation) ;
- facilité d'analyse des résultats ;
- plateforme d'exécution et type de licence.

3.3.2 NS-2 et SensorSim

De tous les simulateurs, NS-2 (*Network Simulator*) est le plus connu et le plus communément utilisé. Il s'agit d'un simulateur à évènements discrets orienté objet dont le développement a débuté en 1989, la version 2 date quant à elle de 1995. Il est open source et multi-plateformes (FreeBSD, Linux, Solaris, Windows, MAC). Le développement s'effectue en OTCL (*Object TCL*) et C++ :

- OTcl est utilisé pour la configuration des simulations ;
- C++ pour créer les classes de base (calcul de routes, etc.).

En fait, le noyau du simulateur (intégrant la plupart des protocoles réseaux) est écrit en C++, tandis que l'interface de programmation est en OTCL.

NS-2 cible surtout la simulation d'architectures réseau suivant le modèle OSI, proposant divers protocoles pour chaque couche du modèle, y compris la couche physique. D'ailleurs, vu sa popularité il intègre un très grand nombre de protocoles, que ce soit pour des réseaux sans fil ou non. Des générateurs de trafic et des modèles de consommation d'énergie sont d'autre part disponibles. Néanmoins, on peut noter que les modèles de couche physique sont simplistes et que certains protocoles comportent parfois des erreurs. Pour ce qui est de la simulation de réseaux de capteurs, divers projets s'y sont attelés. Par exemple, le projet SensorSim [107] de l'Université de Californie Los Angeles visait à créer un simulateur spécifique aux réseaux de capteurs sur la base de NS-2. Pour cela, il faut en particulier ajouter des modèles de capteurs et de batteries.

Au niveau des performances, elles sont fortement liées au nombre de paquets échangés entre les nœuds. En théorie, NS-2 peut gérer plus de 15000 nœuds, mais en pratique il est difficile d'aller au-delà de 1000 nœuds. Pour finir, le résultat d'une simulation est essentiellement composé d'un fichier retraçant l'ensemble des envois,

réceptions et suppressions de paquets. L'exploitation des résultats est facilité par l'outil graphique *Network AniMator* (NAM), celui-ci permet en outre d'éditer des scénarii simples.

3.3.3 `GloMoSim` et `QualNet`

Comme son nom complet l'indique, *Global Mobile Information Systems Simulation Library*, il s'agit d'un environnement de simulation consistant en une bibliothèque d'APIs. En effet, tout comme NS-2, GloMoSim reprend l'architecture réseau par couche du modèle OSI. Or les différentes couches (application, transport, réseau, liaison de données, réception de paquets, ondes radio, mobilité) interagissent par le biais d'APIs standardisées. Bien que conçu dans la perspective de pouvoir simuler des réseaux quelconques, à l'heure actuelle seule la simulation de réseaux sans fil est possible. Une caractéristique intéressante de GloMoSim est qu'il est explicitement prévu pour faciliter le passage à l'échelle :

- d'une part, il est théoriquement capable de simuler un réseau comportant jusqu'à 100000 nœuds ;
- d'autre part, comme le noyau du simulateur repose sur la bibliothèque parallèle Parsec, l'exécution parallèle d'une simulation est envisageable.

Les performances réelles sur une station de travail sont de l'ordre de l'heure de calcul pour une simulation comportant 5000 nœuds.

Le simulateur GloMoSim est multi-plateformes et utilisable uniquement à des fins de recherche académique. Pour un usage commercial, il faut utiliser QualNet, son pendant payant. Ce dernier bénéficie d'une documentation plus fournie, d'un support technique et d'une interface graphique. Notons que le projet européen BISON (*Biology-Inspired techniques for Self-Organization in dynamic Networks* - 2003 à 2006) avait retenu QualNet après une étude comparative.

3.3.4 `JiST / SWANS`

Il s'agit d'un environnement de simulation écrit en Java pour simuler des réseaux ad hoc sans fil [108]. Plus précisément, JiST (*Java in Simulation Time*) est le moteur de simulation à évènements discrets, tandis que SWANS (*Scalable Wireless Ad hoc Network Simulator*) est le « simulateur ». Une étude comparative des performances du couple JiST / SWANS par rapport à NS-2 et GloMoSim a mis en évidence pour de mêmes réseaux (500 et 5000 nœuds) :

- une occupation mémoire qui est nettement moindre. Ainsi, pour le réseau de 500 nœuds JiST / SWANS occupe un peu plus de 1,1 Ko, contre 5,8 et 5,9 Ko pour GloMoSim et NS-2 ;
- de meilleurs temps de calcul, puisqu'il requiert 43 secondes, contre 82 secondes pour GloMoSim et surtout 7136 secondes pour NS-2.

Enfin, JiST / SWANS offre la possibilité de simuler des réseaux largement plus denses (une simulation de 1 millions de nœuds est possible sur un ordinateur fonctionnant à 2 GHz et disposant de 2 Go de RAM). Il semble surtout souffrir de

sa relative jeunesse qui se traduit par un manque de modèles (la première version date de 2004). Un avantage majeur, inhérent à l'utilisation de Java, est qu'il est facilement portable sur un très grand nombre de plateformes.

3.3.5 OMNeT++

OMNeT++ (*Objective Modular Network Testbed in C++*) est un simulateur à évènements discrets, orienté objet, issu du travail de thèse d'András Varga. Il est open source et multi-plateformes, l'utilisation à des fins commerciales nécessite une licence payante (le simulateur s'appelle alors OMNEST).

À l'image de NS-2 et GloMoSim, le modèle d'architecture réseau considéré est le modèle OSI. OMNeT++ est un simulateur performant car entièrement écrit en C++, capable de gérer des réseaux relativement denses (plusieurs milliers de nœuds). D'autant qu'il permet l'exécution parallèle de simulations (utilisation de LAM-MPI). En plus de C++, le simulateur fait appel à son propre langage de configuration (NED) pour décrire à un haut niveau une simulation. En effet, OMNeT++ utilise un architecture hiérarchique par composants pour construire un réseau, offrant une grande flexibilité. Chaque composant de base est spécifié par un fichier de configuration, puis programmé en C++. Ils peuvent ensuite être éventuellement regroupés pour décrire des composants de plus haut niveau, le sommet de la hiérarchie étant constitué par le réseau.

Comme pour NS-2, un très grand nombre de modèles / protocoles réseau sont disponibles. Par exemple, le *Mobility Framework* [109] permet de simuler des réseaux ad hoc mobiles (implémentation de la norme IEEE 802.11b). En ce qui concerne les réseaux de capteurs, on peut citer la modélisation proposée par Mallanda *et al.* [110]. Du point de vue utilisateur, OMNeT++ dispose d'une interface graphique grâce à laquelle il est possible d'éditer et de suivre l'évolution d'une simulation (analyse des échanges entre nœuds, etc.). Naturellement, cette interface graphique ralentit l'exécution, aussi pour des réseaux de grande taille on peut basculer vers une interface textuelle. L'exploitation des résultats d'une simulation (suite de valeurs pour un même paramètre ou valeur scalaire obtenue à la fin) est relativement aisée, puisque des outils sont fournis pour produire des graphiques.

3.4 Conclusion

Tout nouvel algorithme ou protocole pour réseau de capteurs doit être validé par des expérimentations. Déployé un vrai réseau est tout à fait possible, mais suppose de figer un certain nombre de caractéristiques du réseau de capteurs. Un autre frein important est le coût, surtout si le réseau de capteurs considéré comporte beaucoup de nœuds. Par conséquent, nous avons opté pour la simulation.

De nombreux environnements de simulation ont été proposés. Nous en avons présenté certains dans la section précédente (voir le tableau récapitulatif 3.2), à savoir NS-2, GloMoSim, JiST / SWANS et OMNeT++, mais il en existe bien d'autres

tels que Shawn [111]. Divers critères permettent de les différencier : performances (en particulier la densité des réseaux supportés), modèles existants, etc. Nous avons choisi OMNeT++ car il est performant et que le développement de simulations est relativement facile. De plus, divers modèles / protocoles sont disponibles et ce simulateur commence à avoir un certain succès dans la communauté.

NS-2	
Site Web	http ://www.isi.edu/nsnam/ns/
Plateforme	Unix (Linux, solaris, Mac OS X incertain), Microsoft Windows (pas d'expérience d'installation)
Licence	Gratuite
SensorSIM	
Site Web	http ://nesl.ee.ucla.edu/projects/sensorsim/
Plateforme	Unix (Linux, solaris, Mac OS X incertain) Microsoft Windows
Licence	Gratuite
GlomoSim	
Site Web	http ://pcl.cs.ucla.edu/projects/glomosim/
Plateforme	Unix
Licence	Gratuit pour les universitaires.
JSim	
Site Web	http ://chief.cs.uga.edu/ jam/jsim/
Plateforme	Java
Licence	Gratuit
Jist / SWANS	
Site Web	http ://jist.ece.cornell.edu/
Plateforme	Java
Licence	Gratuit
SimPy	
QualNet	
Site Web	http ://www.scalable-networks.com/products/qualnet.php
Plateforme	Microsoft Windows, Linux, Solaris
Licence	Commerciale. Des réductions sont appliquées pour la recherche.
OMNeT++	
Site Web	http ://www.omnetpp.org/
Plateforme	Microsoft Windows (avec Cygwin), Unix
Licence	Gratuit pour les universitaires et pour toute utilisation non lucrative.

TAB. 3.2 – Récapitulatif de quelques simulateurs.

Deuxième partie

Contribution à l'optimisation de la durée de vie d'un réseau de capteurs

Chapitre 4

Une contribution au niveau de la couche application

4.1 Introduction

Les nœuds capteurs ayant une capacité énergétique limitée, pour avoir un réseau fonctionnel le plus longtemps possible, une bonne gestion de l'énergie est primordiale. Pour réduire la consommation énergétique d'un nœud capteur et donc prolonger sa durée de vie, on a vu dans le chapitre 2 que l'on peut intervenir à différents niveaux, en particulier :

- développer des composants physiques consommant moins d'énergie ;
- contrôler son activité en alternant périodes de veille et d'activité ;
- définir des protocoles de communication économes en énergie.

Le dernier point a donné lieu à de nombreux travaux. Ceux-ci concernent d'une part la couche de liaison par la définition de protocoles d'accès au médium de communication (MAC), d'autre part la couche réseau avec les protocoles de routage et de gestion de la topologie. Rappelons que nous considérons que la durée de vie d'un réseau correspond à celle du premier nœud défaillant. Ainsi, un réseau sera dit hors service dès lors qu'un de ses nœuds aura épuisé son énergie.

Plutôt que de définir un énième protocole de communication, nous avons choisi de développer une approche au niveau applicatif. Un des avantages est que cette approche est complémentaire avec les protocoles de communication. On se place de plus dans un contexte où les nœuds sont toujours en activité. Au niveau applicatif, chaque nœud est en charge d'un certain nombre de tâches. Celles-ci peuvent être hétérogènes, i.e. de différents types suivant l'application visée : calculs, réseau, stockage de données, voire un mélange de plusieurs types. Réduire la consommation énergétique au niveau applicatif d'un nœud reviendrait donc à réduire le nombre de tâches qu'il doit exécuter. Naturellement, cela n'est pas possible pour tous les nœuds, puisque cela affecterait fortement le fonctionnement du réseau (nœuds égoïstes refusant de router des paquets, etc.). L'idée est plutôt de faire en sorte que le coût énergétique de l'exécution de la charge d'un nœud soit proportionnelle à son niveau d'énergie restant, avec un coefficient de proportionnalité identique pour tous les nœuds.

4.2 Principe de l'approche

Considérons un réseau de 8 nœuds hétérogènes, avec 3 catégories différentes de nœuds. Un tel réseau est présenté par la figure 4.1, où les nœuds 3, 4 et 8 sont des nœuds dont les capacités sont supérieures à celles des nœuds 1, 6 et 7, mais inférieures à celles des nœuds 2 et 5. Supposons d'autre part que chaque nœud est en charge de différents types de tâches :

- des tâches de traitement de type 1 qui correspondent à du calcul intensif ;
- des tâches de traitement de type 2 qui comportent des phases de calcul et d'accès en mémoire ;
- des tâches réseau pour acheminer les données à travers le réseau.

On constate que le nœud 4 joue un rôle majeur, puisque s'il disparaît le réseau ne forme plus une seule et unique composante connexe. Ce rôle de passerelle fait évidemment que ce nœud est en charge de beaucoup de tâches réseau, d'où une consommation en énergie nettement plus importante, comme le montre le cas A. Afin de réduire l'énergie consommée par le nœud 4 et donc prolonger la durée de vie du réseau, nous proposons que les nœuds fassent une sorte de « deal » entre eux, de manière à décharger le nœud 4 de certaines de ses tâches. Plus précisément, ainsi que le met en évidence le cas B, il s'agit de faire migrer des tâches du nœud 4 vers d'autres nœuds qui sont moins sollicités. Dans le cas B on observe que toutes les tâches de traitement de type 2 du nœud 4 ont été distribuées à des nœuds voisins qui vont donc consommer plus d'énergie. C'est en particulier le cas des nœuds 2 et 5. Pour cela, nous allons définir un algorithme de migration de tâches qui sera exécuté par chaque nœud. De sorte qu'un nœud dont la charge est proportionnellement plus consommatrice en énergie (par rapport à sa provision d'énergie) que celle de ses voisins leurs envoie des tâches.

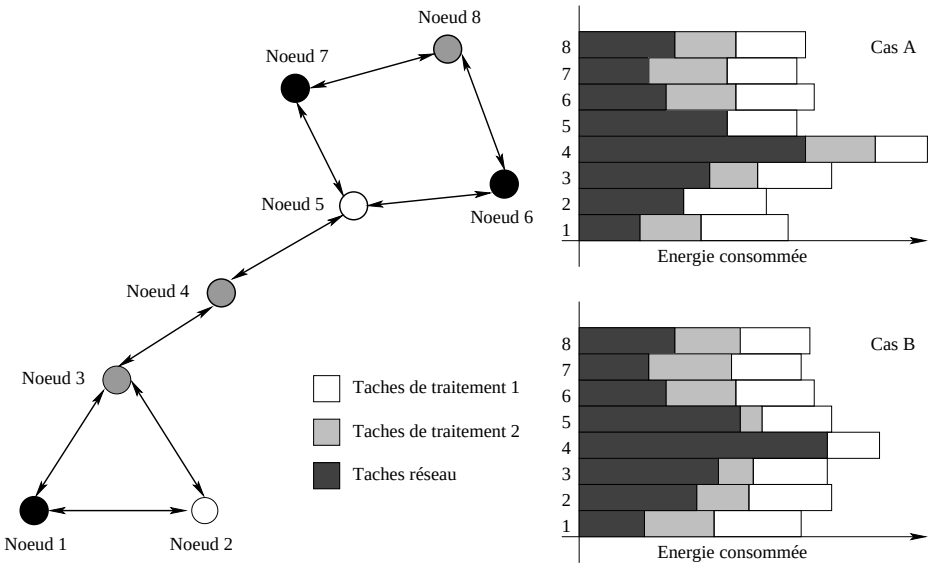


FIG. 4.1 – Réseau mettant en évidence le principe de l'approche.

4.3 Fondements scientifiques

Nous nous plaçons dans le contexte où un nœud i du réseau peut uniquement communiquer avec les nœuds se trouvant dans son rayon de couverture r_i. L'ensemble des nœuds avec lesquels le nœud i peut communiquer directement constitue son voisinage que l'on désignera par V_i. On rappelle que les communications entre nœuds distants d'un même réseau sont assurées par sauts successifs (multi-sauts). Cela signifie que les nœuds intermédiaires doivent faire suivre les données jusqu'au nœud destinataire.

Dans le réseau représenté par le figure 4.2 on voit que le nœud i ne peut pas joindre de façon directe tous les autres nœuds du réseau. Néanmoins, le réseau est complètement connecté au sens de la définition 2. Dans ce cas de figure, les communications entre le nœud i (source) et les nœuds $\{j, k, l, m\} \notin V_i$ (destinataires) se font grâce aux nœuds relais qui se trouvent dans le cercle de rayon r_i (voisinage du nœud i). Le rôle d'un nœud relais est de transmettre les paquets de données entre la source et la destination en l'absence d'une route directe entre ces derniers. Lors de l'initialisation du réseau, les nœuds s'auto-organisent et construisent leurs tables de routage (ils optent pour une politique de routage), afin de permettre à des nœuds non voisins de communiquer.

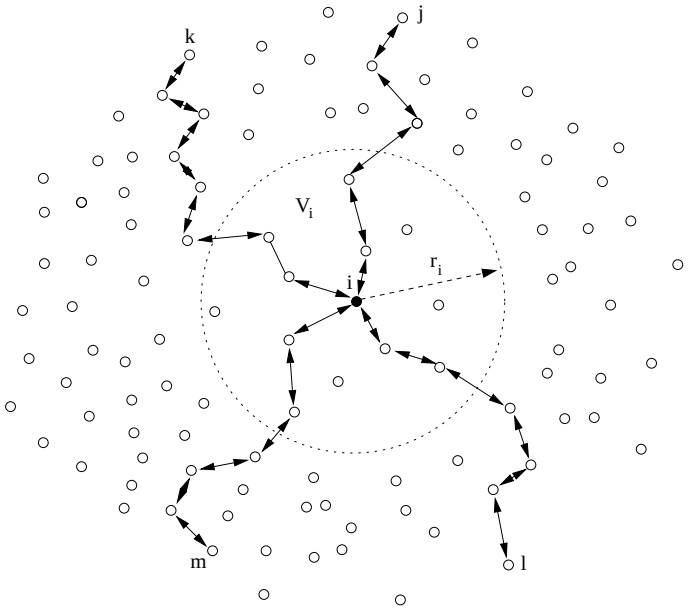


FIG. 4.2 – Communications distantes multi-sauts.

La mobilité éventuelle des nœuds et leur fonctionnement décentralisé donne lieu à plusieurs problématiques non traitées dans les réseaux de type filaire. En effet, le réseau étant sans infrastructure prédéfinie, chaque nœud doit exécuter en plus de sa charge certaines tâches réseau (relais et routage). Par conséquent, chaque nœud se doit de trouver le meilleur équilibre entre l'exécution de ses propres tâches et celles relevant du réseau.

Définition 1 ***Graphe connexe***

- *Un graphe est dit connexe si et seulement si il existe au moins un chemin entre chaque paire de sommets (le chemin n'est pas implicitement direct).*
- *Un graphe connexe est un graphe dans lequel chaque paire de sommets est reliée par une chaîne.*
- *Un graphe qui n'est pas connexe peut être décomposé en composantes connexes.*

Remarque *:*

Que se passe-t-il si le graphe G n'est pas connexe ?

Il apparaît alors comme un ensemble de graphes connexes mis les uns à coté des autres. Chacun de ces graphes est un sous-graphe particulier de G, appelé « composante connexe ».

Définition 2 ***Connexité du réseau***

Intuitivement, nous définissons la connectivité du réseau comme suit :

1. $\forall k = \{1, 2, \ldots\}$ $\exists$ *sous-ensemble du réseau (graphe)* $G^{(k)} = \{\phi_1^{(k)}, \phi_2^{(k)}, \ldots\} \subset G$ *tel que* $\phi_i^{(k)}$ *est un graphe connexe.*
2. *si* $\bigcup_{k=1}^{\infty} G^k$ *est un graphe connecté (connexe), la matrice d'incidence associée* B *a pour propriété :* $\lim_{k\to\infty} B^k = 0$

Cette définition traduit l'état global du réseau tout au long de sa durée de vie : le réseau n'est constitué que d'une seule composante connexe.

Hypothèses 1

On considère un réseau de capteurs ayant les caractéristiques suivantes :

- *il est constitué de N nœuds hétérogènes ;*
- *les interfaces de communications sont identiques pour tous les nœuds ;*
- *chaque nœud a une source d'énergie non renouvelable ;*
- *déploiement aléatoire et densité dans le sens de la définition 2*
- *etc.*

Maximiser la durée de vie d'un réseau de capteurs revient à développer des modèles et des mécanismes au niveau de divers couches du modèle OSI (section 1.5, figure 1.5), afin d'optimiser l'utilisation de la ressource énergétique. Si chaque nœud consomme à bon escient l'énergie pour accomplir ses tâches, cela permettra d'augmenter la durée de vie globale du réseau. Ces mécanismes doivent impliquer la coopération entre les nœuds, de manière à éviter l'apparition d'un comportement égoïste de certains d'entre eux. En effet, pour économiser de l'énergie, des nœuds pourraient réduire le nombre de tâches dont ils ont la charge et en particulier celles liées à la dynamique du réseau comme les tâches de relais et de routage. Il en résulte que des nœuds égoïstes peuvent nettement pénaliser le fonctionnement global d'un réseau, puisqu'ils privilégient leur survie indépendamment du fonctionnement global du réseau. Des travaux ont montré que si un réseau comporte de 10 à 15% de nœuds égoïstes ses performances seront fortement dégradées.

Notre objectif est de développer une approche qui permette d'une part de prolonger la durée de vie d'un nœud donné, d'autre part d'assurer l'accomplissement des différentes tâches. Pour ce faire, nous allons définir un indicateur qui sera calculé par chaque nœud et qui servira pour contrôler la migration de tâches entre nœuds voisins. La figure 4.3 présente, dans le cas d'un réseau de 7 nœuds, la quantité d'énergie dont dispose chaque nœud i à l'initialisation ($E_i^{(0)}$) et l'énergie dont il aura besoin pour exécuter ses tâches (ψ_i). On observe par exemple que le nœud 2 a une provision d'énergie supérieure à l'énergie qu'il consommerait en exécutant ses tâches. En revanche, d'autres nœuds n'auront pas assez d'énergie, à savoir les nœuds 3, 4, 5 et 6. Ceux-ci devront donc envoyer des tâches à des voisins ayant encore de l'énergie disponible pour exécuter celles-ci. Si on considère le rapport des énergies ψ_i sur $E_i^{(0)}$ (≥ 0) d'un nœud i on constate que :

- si sa valeur est supérieure à 1, le nœud ne sera pas en mesure d'exécuter toute sa charge car il aura une provision d'énergie insuffisante ;
- si sa valeur est inférieure ou égale à 1, le nœud disposera de suffisamment d'énergie. En outre, il pourra d'autant plus exécuter des tâches supplémentaires (provenant de ses voisins) que son rapport d'énergies est éloigné de 1.

C'est précisément ce rapport, que nous appellerons également ratio, qui constituera l'indicateur que nous allons utiliser pour contrôler la migration de tâches entre nœuds voisins. Ainsi, la différence de ratio entre deux nœuds voisins permettra de définir quel nœud enverra des tâches à son voisin et en quelle quantité. Le but est d'obtenir au final un ratio identique pour tous les nœuds du réseau, puisque dans ce cas ils participeront proportionnellement de manière égale à l'exécution de tâches.

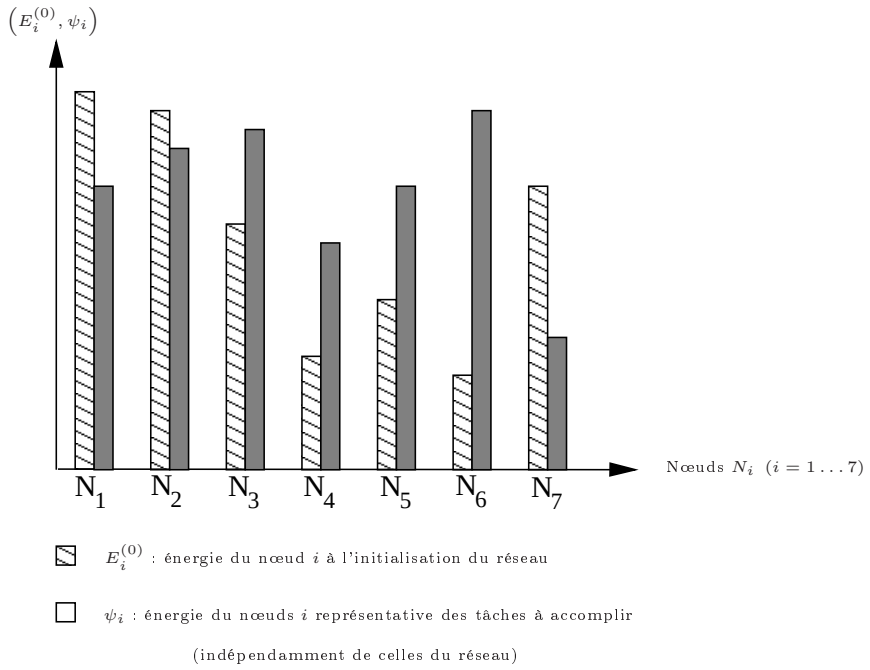


FIG. 4.3 – Mise en évidence du problème - niveaux d'énergie des nœuds.

L'approche que nous proposons pouvant être vue comme une méthode d'équilibrage d'un ratio d'énergies à travers le réseau, c'est tout naturellement que nous allons nous inspirer des techniques d'équilibrage de charge dans les systèmes distribués. Celles-ci ont été proposées dans le cadre du calcul distribué, pour répondre à la problématique de la répartition des calculs sur différents processeurs, dans la perspective d'obtenir le résultat le plus rapidement possible. L'équilibrage de charge a été surtout étudié dans le contexte des topologies de réseaux statiques, ce n'est que récemment que des solutions ont été proposées pour des topologies dynamiques [112]. De plus, on distingue d'une part des méthodes d'équilibrage centralisées et d'autre part des méthodes décentralisées. Bien entendu, nous nous focalisons sur ces dernières, puisque dans le cas d'un réseau de capteurs un algorithme décentralisé est préférable. Parmi les méthodes décentralisées existantes, nous nous inspirons de l'algorithme de diffusion de premier ordre proposé par Cybenko [113]. Cet algorithme procède itérativement pour équilibrer au fur et à mesure la charge et converger vers un état stable. La vitesse de convergence, i.e. le nombre d'itérations nécessaires pour atteindre l'état stable, dépend de la matrice de diffusion. Différentes méthodes de construction de la matrice de diffusion ont été définies, nous avons opté pour la méthode introduite par Boillat [114] car il s'agit d'une méthode de construction locale. En définitive, nous allons proposer un algorithme de migration de tâches complètement décentralisé, qui sera exécuté de manière synchrone par tous les nœuds du réseau de capteurs considéré.

Chapitre 5

Un algorithme de migration de tâches

Dans ce chapitre, nous allons introduire formellement l'algorithme que nous proposons et nous étudierons sa validité.

5.1 Formalisation du problème

L'évolution temporelle du réseau considéré est supposée être en temps discret, avec s qui désigne la longueur d'un pas de temps. Aussi, chaque instant $t \geq 0$ satisfait $t = k \times s$ où $k \in \mathbb{N}$, pour simplifier dans la suite nous utiliserons k en lieu et place de t. On fait également l'hypothèse que le réseau est constitué de N nœuds $i \in \{1, \ldots, N\}$ et on considère M sous-ensembles (ou classes) de tâches, avec un coût d'exécution d'unitaire (d'une tâche) différent d'un sous-ensemble à un autre. Comme les nœuds du réseau peuvent être hétérogènes, l'exécution de tâches d'une même classe aura un coût énergétique qui variera suivant les caractéristiques de chaque type de nœud. Finalement, le tableau 5.1 présente les autres symboles de notation qui seront utilisés.

L'énergie consommée par le nœud i durant le pas de temps k est donnée par $\xi_i^{(k)} = \sum_{j=1}^{M} N_{i,j}^{(k)} e_{i,j}$. Par conséquent, la provision d'énergie dont dispose le nœud i à l'instant $k > 0$ vérifie :

$$E_i^{(k)} = E_i^{(0)} - \sum_{l=0}^{k-1} \xi_i^{(l)} = E_i^{(0)} - \sum_{l=0}^{k-1} \sum_{j=1}^{M} N_{i,j}^{(l)} e_{i,j} \ . \tag{5.1}$$

Symbole	Signification
Ω	L'ensemble des nœuds hétérogènes du réseau ($\Omega = \{1, 2, \ldots, N\}$)
N	Le nombre de nœuds du réseau ($N = \lvert\Omega\rvert$)
$E_i^{(k)}$	L'énergie dont dispose le nœud i à l'instant k
$T_{i,j}^{(k)}$	Le nombre de tâches de classe j dont dispose le nœud i à l'instant k
$e_{i,j}$	L'énergie nécessaire au nœud i à l'exécution d'une seule tâche de classe j
$N_{i,j}^{(k)}$	Le nombre de tâches de la classe j exécutées par le nœud i durant l'intervalle de temps k

TAB. 5.1 – Symboles de notation.

Comme on cherche à optimiser la provision d'énergie au pas de temps k, la seule façon d'atteindre cet objectif est selon l'équation (5.1) d'adapter le nombre de tâches que le nœud i exécute $\left(\sum_{l=0}^{k-1}\sum_{j=1}^{M} N_{i,j}^{(l)}\right)$. Évidemment, diminuer ce nombre pour chacun des nœuds n'est pas approprié, puisque ce scénario mènerait à une baisse de la qualité de service (QoS). C'est pourquoi, notre objectif est d'équilibrer la charge des nœuds en fonction de l'énergie dont chacun dispose. Plus précisément, on cherche un instant S tel que $\forall k \geq S$ on ait :

$$\frac{\sum_{j=1}^{M} T_{1,j}^{(k)} e_{1,j}}{E_1^{(k)}} = \cdots = \frac{\sum_{j=1}^{M} T_{N,j}^{(k)} e_{N,j}}{E_N^{(k)}} \tag{5.2}$$

où $T_{i,j}^{(k)}$ représente la charge restante, $T_{i,j}^{(k)} = T_{i,j}^{(0)} - \sum_{l=0}^{k-1} N_{i,j}^{(l)}$. L'équation (5.2) exprime le fait que le rapport entre l'énergie nécessaire pour exécuter la charge restante et l'énergie disponible est la même pour tous les nœuds. Cela garantit d'un côté que chaque nœud participe à l'exécution de tâches en fonction de sa ressource énergétique et d'un autre côté l'équité de notre approche. Dans la suite, le ratio du nœud i sera désigné par $x_i^{(k)} = \frac{\sum_{j=1}^{M} T_{i,j}^{(k)} e_{i,j}}{E_i^{(k)}}$.

Notre approche définit clairement un système convergeant vers un point fixe. En effet, soit $X^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)}\right)$ le vecteur de ratios, alors $\forall k \geq S$:

$$X^{(k+1)} = X^{(k)} = X^{(S)} = \left(x_1^{(S)}, x_2^{(S)}, \ldots, x_N^{(S)}\right) = (x^*, \ldots, x^*) = X^* \ . \tag{5.3}$$

5.2 Définition de l'algorithme

Chaque nœud va exécuter un algorithme qui sera « guidé » d'une part par la quantité d'énergie dont il dispose, d'autre part par sa charge. Ces informations, combinées avec la connaissance du ratio de chaque nœud voisin à un saut (*one-hop neighbors*), permettent de savoir si le nœud doit se délester de tâches. Comme notre approche aboutit à un système convergeant vers un point fixe, c'est naturellement un algorithme itératif. Afin d'avoir un algorithme complètement distribué, la détection de convergence sera contrôlée localement (la différence de ratio entre le nœud et ses voisins immédiats - à un saut - est inférieure à un ϵ fixé). Il est important de noter que notre approche est robuste et insensible aussi bien vis à vis de la perte de liens de communications que de l'évolution de la topologie du réseau.

Soit $V_i^{(k)}$ l'ensemble des voisins à un saut du nœud i au pas de temps k. Chaque nœud va alors exécuter l'algorithme 5.1, où $ea_i^{(l)}$ et $D_{i,j}^{(l)}$ représentent respectivement :

- l'énergie consommée par l'exécution de l'algorithme. Il s'agit de l'énergie consommée par les calculs et les communications ;
- le nombre de tâches de la classe j qui ont été engendrées pendant le pas de temps l. C'est un processus poissonnien.

Le principe de l'algorithme est que le nœud l'exécutant échange son ratio avec celui de chacun de ses voisins, puis d'utiliser ces grandeurs pour déduire vers quel(s)

Algorithme 5.1 Algorithme optimisant la durée de vie par migration de tâches

$l = k$
Convergence = faux
repeat
  if $V_i^{(l)} \neq \emptyset$ **then**
    Échanger $x_i^{(l)}$ avec chaque $v \in V_i^{(l)}$ (i.e. envoyer $x_i^{(l)}$ et recevoir son ratio $x_v^{(l)}$)
    $x_i^{(l+1)} = x_i^{(l)} + \sum_{v \in V_i^{(l)}} A_{iv}^{(l)} \cdot \left(x_v^{(l)} - x_i^{(l)} \right)$
    if $x_i^{(l+1)} - x_i^{(l)} > \epsilon$ **then**
      Envoyer des tâches aux nœuds voisins $\left(v \in V_i^{(l)} \right)$ suivant l'algorithme 5.2
    else
      Convergence = vrai
    end if
  else
    Convergence = vrai
  end if
  $l = l + 1$
  $E_i^{(l)} = E_i^{(l-1)} - \sum_{j=1}^{M} N_{i,j}^{(l-1)} e_{i,j} - ea_i^{(l-1)}$
  $T_{i,j}^{(l)} = T_{i,j}^{l-1} - N_{i,j}^{(l-1)} + D_{i,j}^{(l-1)}$
  $x_i^{(l)} = \frac{\sum_{j=1}^{M} T_{i,j}^{(l)} e_{i,j}}{E_i^{(l)}}$
until Convergence est vrai

voisin(s) et en quelle quantité envoyer des tâches. En fait, chaque nœud va uniquement envoyer des tâches à ses voisins qui ont un ratio inférieur au sien. Des migrations de tâches vont s'opérer à chaque itération jusqu'à détection de la convergence : la différence de ratio (ligne 7) est inférieure à ϵ. Les coefficients de pondération de la quantité de ratio échangée, $A_{iv}^{(k)}$, correspondent aux composantes de la matrice de diffusion $A^{(k)}$. Cette matrice est construite à chaque itération de manière à pouvoir appréhender l'évolution de la topologie du réseau. La provision d'énergie du nœud i diminue à la suite de l'exécution de tâches et de l'algorithme (ligne 16), alors que le nombre tâches décroît également mais peut éventuellement augmenter (ligne 17).

5.2.1 Migration de tâches

Comme l'algorithme 5.1 est décentralisé, il ne requiert que des communications locales (chaque nœud ne communique qu'avec ses voisins à un saut). Voyons maintenant le nombre de tâches qui devront être envoyées à un nœud voisin ayant une valeur de ratio inférieure.

À partir de la ligne 6 de l'algorithme on obtient pour le nœud i :

$$x_i^{(k+1)} = x_i^{(k)} + \sum_{v \in V_i^{(k)}} A_{iv}^{(k)} \cdot \left(x_v^{(k)} - x_i^{(k)} \right) \quad . \tag{5.4}$$

Considérons maintenant un voisin v quelconque dans $V_i^{(k)}$, on peut alors faire les observations suivantes :

1. $x_v^{(k)} < x_i^{(k)}$ fait décroître le ratio de la quantité $\left| A_{iv}^{(k)} \cdot \left(x_v^{(k)} - x_i^{(k)} \right) \right|$. En conséquence, le nœud i doit envoyer au nœud v suffisamment de tâches pour induire cette décroissance de ratio ;
2. si $x_v^{(k)} = x_i^{(k)}$ il n'y pas de migration de tâches ;
3. $x_v^{(k)} > x_i^{(k)}$ est le dual du premier cas.

Soit $\alpha_{i,v,j}^{(k)}$ le nombre de tâches de classe j devant être envoyées par le nœud i au nœud v au pas de temps k. Alors l'équation (5.4) peut être réécrite comme suit :

$$x_i^{(k+1)} = x_i^{(k)} + \sum_{v \in V_i^{(k)}} \operatorname{sgn}\left(x_v - x_i\right) \cdot \frac{\sum_{j=1}^{M} \alpha_{\phi_1(i,v),\phi_2(i,v),j}^{(k)} e_{i,j}}{E_i^{(k)}} \tag{5.5}$$

où

$$\begin{aligned} \phi_1(i,v) &= \frac{\left(1 - \operatorname{sgn}\left(x_v - x_i\right)\right) \cdot i + \left(1 + \operatorname{sgn}\left(x_v - x_i\right)\right) \cdot v}{2} \quad , \\ \phi_2(i,v) &= \frac{\left(1 + \operatorname{sgn}\left(x_v - x_i\right)\right) \cdot i + \left(1 - \operatorname{sgn}\left(x_v - x_i\right)\right) \cdot v}{2} \quad . \end{aligned} \tag{5.6}$$

Bien entendu, on ne s'intéresse qu'au premier cas. Le nombre de tâches à envoyer par le nœud i au nœud v au pas de temps k peut alors être déduit des équations (5.4)

et (5.5) :

$$\frac{\sum_{j=1}^{M} \alpha_{i,v,j}^{(k)} e_{i,j}}{E_i^{(k)}} = A_{iv}^{(k)} \cdot \left(x_i^{(k)} - x_v^{(k)} \right) \quad . \tag{5.7}$$

De cette équation on obtient :

$$\sum_{j=1}^{M} \alpha_{i,v,j}^{(k)} e_{i,j} = A_{iv}^{(k)} \cdot \left(\sum_{j=1}^{M} T_{i,j}^{(k)} e_{i,j} - E_i^{(k)} x_v^{(k)} \right) = \sum_{j=1}^{M} \left(A_{iv}^{(k)} \cdot \gamma_{i,v,j}^{(k)} \right) \cdot e_{i,j} \quad . \tag{5.8}$$

Se pose maintenant le problème du calcul des coefficients $\gamma_{i,v,j}^{(k)}$.

Comme nous allons le voir, différentes approches sont possibles. Avant tout, nous devons régler un problème majeur, à savoir éviter que le nombre de tâches à envoyer soit supérieur au nombre dont un nœud dispose. La solution consiste à ajouter des contraintes garantissant que le nombre de tâches dont le nœud dispose est toujours un nombre positif ou nul, quelle que soit la classe. Pour ce faire, on encadre $\gamma_{i,v,j}^{(k)}$ comme suit :

$$0 \leq \left(\alpha_{i,v,j}^{(k)} = A_{iv}^{(k)} \cdot \gamma_{i,v,j}^{(k)} \right) \leq \frac{T_{i,j}^{(k)}}{|V_i^{(k)}|} \quad \Leftrightarrow \quad 0 \leq \gamma_{i,v,j}^{(k)} \leq \frac{T_{i,j}^{(k)}}{A_{iv}^{(k)} \cdot |V_i^{(k)}|} \quad . \tag{5.9}$$

- **Méthode de calcul 1**

  Supposons que pour tout $j \in \{2, \ldots, M\}$ on ait $\gamma_{i,v,j}^{(k)} = 0$, hypothèse valide à partir de (5.9), on obtient implicitement :

  $$\gamma_{1,v,j}^{(k)} e_{1,j} = \sum_{j=1}^{M} \gamma_{i,v,j}^{(k)} e_{i,j} \tag{5.10}$$

  ce qui définit une borne supérieure pour $\gamma_{1,v,j}^{(k)} e_{1,j}$. Toujours à partir de l'encadrement (5.9), on peut maintenant faire l'hypothèse que $\gamma_{i,v,j}^{(k)} = \frac{T_{i,j}^{(k)}}{A_{iv}^{(k)} |V_i^{(k)}|}$ pour tout $j \in \{2, \ldots, M\}$. On aboutit ainsi à une borne inférieure pour $\gamma_{1,v,j}^{(k)} e_{1,j}$:

  $$\gamma_{1,j}^{(k)} e_{1,j} = \sum_{j=1}^{M} \gamma_{i,j}^{(k)} e_{i,j} - \sum_{l=2}^{M} \frac{T_{i,l}^{(k)}}{A_{iv}^{(k)} |V_i^{(k)}|} e_{i,l} \tag{5.11}$$

  Des équations (5.10) et (5.11) on peut définir un encadrement de $\gamma_{1,v,j}^{(k)} e_{1,j}$, puis de $\gamma_{1,v,j}^{(k)}$:

  $$\sum_{j=1}^{M} \gamma_{i,v,j}^{(k)} e_{i,j} - \sum_{l=2}^{M} \frac{T_{i,l}^{(k)}}{A_{iv}^{(k)} |V_i^{(k)}|} e_{i,l} \leq \quad \gamma_{i,v,1}^{(k)} e_{i,1} \quad \leq \sum_{j=1}^{M} \gamma_{i,v,j}^{(k)} e_{i,j} \tag{5.12}$$

  $$\frac{\sum_{j=1}^{M} \gamma_{i,v,j}^{(k)} e_{i,j} - \sum_{l=2}^{M} \frac{T_{i,l}^{(k)}}{A_{iv}^{(k)} |V_i^{(k)}|} e_{i,l}}{e_{i,1}} \leq \quad \gamma_{i,v,1}^{(k)} \quad \leq \frac{\sum_{j=1}^{M} \gamma_{i,v,j}^{(k)} e_{i,j}}{e_{i,1}} \tag{5.13}$$

On peut alors combiner les encadrements (5.9) et (5.13) en un seul, aboutissant à :

$$\max\left(0, \frac{\sum_{j=1}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} - \sum_{l=2}^{M}\frac{T_{i,l}^{(k)}}{A_{iv}^{(k)}|V_i^{(k)}|}e_{i,l}}{e_{i,1}}\right) \leq \gamma_{i,v,1}^{(k)} \leq \min\left(\frac{\sum_{j=1}^{M}\gamma_{i,v,j}^{(k)}e_{i,j}}{e_{i,1}}, \frac{T_{i,1}^{(k)}}{A_{iv}^{(k)}|V_i^{(k)}|}\right)$$

Pour finir $\gamma_{i,v,1}^{(k)}$ est tiré aléatoirement dans l'intervalle défini par l'encadrement précédent.
Dès lors que $\gamma_{i,v,1}^{(k)}$ est fixé, on peut calculer $\gamma_{i,v,2}^{(k)}$ en reprenant la même approche. L'encadrement associé que l'on obtient est :

$$\max\left(0, \frac{\left(\sum_{j=1}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} - \gamma_{i,v,1}^{(k)}e_{i,1}\right) - \sum_{l=3}^{M}\frac{T_{i,l}^{(k)}}{A_{iv}^{(k)}|V_i^{(k)}|}e_{i,l}}{e_{i,2}}\right) \leq \gamma_{i,v,2}^{(k)}$$

$$\gamma_{i,v,2}^{(k)} \leq \min\left(\frac{\sum_{j=1}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} - \gamma_{i,v,1}^{(k)}e_{i,1}}{e_{i,2}}, \frac{T_{i,2}^{(k)}}{A_{iv}^{(k)}|V_i^{(k)}|}\right)$$

En appliquant récursivement ce schéma, on dérive l'expression suivante pour tout indice σ de classe de tâches, $\sigma \in \{1, \ldots, M-1\}$:

$$\max\left(0, \frac{\left(\sum_{j=1}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} - \sum_{m=1}^{\sigma-1}\gamma_{i,v,m}^{(k)}e_{i,m}\right) - \sum_{l=\sigma+1}^{M}\frac{T_{i,l}^{(k)}}{A_{iv}^{(k)}|V_i^{(k)}|}e_{i,l}}{e_{i,\sigma}}\right) \leq \gamma_{i,v,\sigma}^{(k)}$$

$$\gamma_{i,v,\sigma}^{(k)} \leq \min\left(\frac{\sum_{j=1}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} - \sum_{m=1}^{\sigma-1}\gamma_{i,v,m}^{(k)}e_{i,m}}{e_{i,\sigma}}, \frac{T_{i,\sigma}^{(k)}}{A_{iv}^{(k)}|V_i^{(k)}|}\right)$$

Tous les termes $\gamma_{i,v,1}^{(k)}, \ldots, \gamma_{i,v,\sigma}^{(k)}, \ldots, \gamma_{i,v,M-1}^{(k)}$ sont tirés aléatoirement d'après le schéma récursif donné ci-dessus. Le dernier terme, $\gamma_{i,v,M}^{(k)}$, se calcule lui comme suit :

$$\gamma_{i,v,M}^{(k)} = \frac{\sum_{j=1}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} - \sum_{m=1}^{M-1}\gamma_{i,v,m}^{(k)}e_{i,m}}{e_{i,M}} \tag{5.14}$$

- **Méthode de calcul 2**

Pour calculer les coefficients $\gamma_{i,v,j}^{(k)}$ on utilise le schéma suivant :

$$\begin{aligned}
\sum_{j=1}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} &= \sum_{j=1}^{M}T_{i,j}^{(k)}e_{i,j} + \sum_{j=1}^{M}\frac{1}{M}\left[-E_i^{(k)}x_v^{(k)}\right] \\
\sum_{j=2}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} &= \sum_{j=2}^{M}T_{i,j}^{(k)}e_{i,j} + \sum_{j=2}^{M}\frac{1}{M-1}\left[\left(T_{i,1}^{(k)} - \gamma_{i,v,1}^{(k)}\right)e_{i,1} - E_i^{(k)}x_v^{(k)}\right] \\
\sum_{j=3}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} &= \sum_{j=3}^{M}T_{i,j}^{(k)}e_{i,j} + \sum_{j=3}^{M}\frac{1}{M-2}\left[\left(T_{i,1}^{(k)} - \gamma_{i,v,1}^{(k)}\right)e_{i,1} + \left(T_{i,2}^{(k)} - \gamma_{i,v,2}^{(k)}\right)e_{i,2} - E_i^{(k)}x_v^{(k)}\right] \\
&\cdots \\
\sum_{j=\sigma}^{M}\gamma_{i,v,j}^{(k)}e_{i,j} &= \sum_{j=\sigma}^{M}T_{i,j}^{(k)}e_{i,j} + \sum_{j=\sigma}^{M}\frac{1}{M-(\sigma-1)}\left[\sum_{l=1}^{\sigma-1}\left(T_{i,l}^{(k)} - \gamma_{i,v,l}^{(k)}\right)e_{i,l} - E_i^{(k)}x_v^{(k)}\right]
\end{aligned}$$

Cela implique l'expression suivante pour $\gamma_{i,v,\sigma}^{(k)}$, $\sigma \in \{1,\dots,M\}$:

$$\begin{aligned}\gamma_{i,v,\sigma}^{(k)} e_{i,\sigma} &= T_{i,\sigma}^{(k)} e_{i,\sigma} + \frac{1}{M-(\sigma-1)} \left[\sum_{l=1}^{\sigma-1} \left(T_{i,l}^{(k)} - \gamma_{i,v,l}^{(k)} \right) e_{i,l} - E_i^{(k)} x_v^{(k)} \right] \\ \gamma_{i,v,\sigma}^{(k)} &= T_{i,\sigma}^{(k)} + \frac{1}{e_{i,\sigma} \cdot (M-(\sigma-1))} \left[\sum_{l=1}^{\sigma-1} \left(T_{i,l}^{(k)} - \gamma_{i,v,l}^{(k)} \right) e_{i,l} - E_i^{(k)} x_v^{(k)} \right]\end{aligned}$$

Comme les bornes inférieure et supérieure sont les mêmes que pour la méthode de calcul précédente, l'équation se réécrit de la façon suivante :

$$\begin{aligned}\gamma_{i,v,\sigma}^{(k)} = \min\Bigg(& \max \left(0, T_{i,\sigma}^{(k)} + \frac{1}{e_{i,\sigma} \cdot (M-(\sigma-1))} \left[\sum_{l=1}^{\sigma-1} \left(T_{i,l}^{(k)} - \gamma_{i,v,l}^{(k)} \right) e_{i,l} - E_i^{(k)} x_v^{(k)} \right] \right), \\ & \min \left(\frac{\sum_{j=1}^{M} \gamma_{i,v,j}^{(k)} e_{i,j} - \sum_{m=1}^{\sigma-1} \gamma_{i,v,m}^{(k)} e_{i,m}}{e_{i,\sigma}}, \frac{T_{i,\sigma}^{(k)}}{A_{iv}^{(k)} |V_i^{(k)}|} \right) \Bigg)\end{aligned} \tag{5.15}$$

- **Méthode de calcul 3**
  Pour finir, nous avons décidé de définir le calcul des coefficients $\gamma_{i,v,j}^{(k)}$ à partir de la charge du nœud i, puisque ces coefficients y sont directement reliés. Le coefficient d'indice j est défini proportionnellement à partir de la charge en tâches de classe j et de la charge globale. On a ainsi :

$$\sum_{j=1}^{M} \gamma_{i,v,j}^{(k)} e_{i,j} = \sum_{j=1}^{M} \frac{T_{i,j}^{(k)}}{\sum_{l=1}^{M} T_{i,l}^{(k)}} \cdot \left(\sum_{j=1}^{M} T_{i,j}^{(k)} e_{i,j} - E_i^{(k)} x_v^{(k)} \right) \tag{5.16}$$

  et le coefficient $\gamma_{i,v,j}^{(k)}$ vérifie :

$$\gamma_{i,v,j}^{(k)} = \frac{T_{i,j}^{(k)}}{e_{i,j} \cdot \sum_{l=1}^{M} T_{i,l}^{(k)}} \cdot \left(\sum_{j=1}^{M} T_{i,j}^{(k)} e_{i,j} - E_i^{(k)} x_v^{(k)} \right) \quad . \tag{5.17}$$

  Finalement, en tenant à nouveau compte des bornes inférieure et supérieure on aboutit au calcul suivant pour chaque coefficient $\gamma_{i,v,\sigma}$, $\sigma \in \{1,\dots,M\}$:

$$\gamma_{i,v,\sigma}^{(k)} = \min \left(\begin{array}{l} \max \left(0, \frac{T_{i,\sigma}^{(k)}}{e_{i,\sigma} \cdot \sum_{l=1}^{M} T_{i,l}^{(k)}} \cdot \left(\sum_{j=1}^{M} T_{i,j}^{(k)} e_{i,j} - E_i^{(k)} x_v^{(k)} \right) \right), \\ \min \left(\frac{\sum_{j=1}^{M} \gamma_{i,v,j}^{(k)} e_{i,j} - \sum_{m=1}^{\sigma-1} \gamma_{i,v,m}^{(k)} e_{i,m}}{e_{i,\sigma}}, \frac{T_{i,\sigma}^{(k)}}{A_{iv}^{(k)} |V_i^{(k)}|} \right) \end{array} \right) \quad . \tag{5.18}$$

La première méthode de calcul est stochastique, tandis que les deux autres sont déterministes. À l'issue d'expérimentations préliminaires, nous avons choisi de ne considérer pour la suite que la troisième méthode de calcul. Il reste un seul petit problème à régler : les coefficients $\alpha_{i,v,j}^{(k)}$ sont des réels. Or comme tout nombre de tâches doit être à valeur discrète, il faut introduire une phase de discrétisation. Ce point sera discuté plus en détail dans le chapitre suivant.

L'algorithme d'envoi de tâches (à un nœud voisin v par le nœud i) que l'on obtient, tenant compte des remarques précédentes, est l'algorithme 5.2.

Algorithme 5.2 Envoi de tâches

1: **for all** nœud $v \in V_i^{(k)}$ **do**
2: **if** $\left(x_v^{(k)} - x_i^{(k)}\right) < 0$ **then**
3: Calculer $\gamma_{i,v,j}^{(k)}, j \in \{1, \ldots, M\}$
4: Discrétiser tous les cœfficients $\alpha_{i,v,j}^{(k)} = A_{iv}^{(k)} \cdot \gamma_{i,v,j}^{(k)}$
5: **for all** j tel que $1 \leq j \leq M$ **do**
6: Envoyer $\beta_{i,v,j}^{(k)}$ tâches au nœud v ($\beta_{i,v,j}^{(k)}$ représente la discrétisation de $\alpha_{i,v,j}^{(k)}$)
7: **end for**
8: **end if**
9: **end for**

5.2.2 Matrice de diffusion

Les algorithmes de diffusion sont bien connus dans le domaine de l'équilibrage de charge [113]. Dans ce contexte, ils furent d'abord introduits pour des réseaux statiques, puis plus récemment pour les réseaux dynamiques [112] (prise en compte de la rupture de liens de communication). Notre approche qui consiste en un algorithme d'équilibrage, non plus de charge mais de ratio, reprend le principe des schémas de diffusion [113]. En effet, comme on peut le constater en regardant l'équation (5.4), nous avons utilisé une loi de diffusion anisotropique :

$$x_i^{(k+1)} = \left(1 - \sum_{v \neq i} A_{iv}^{(k)}\right) \cdot x_i^{(k)} + \sum_{v \neq i} A_{iv}^{(k)} \cdot x_v^{(k)} \quad , \tag{5.19}$$

équation qui peut être réécrite matriciellement sous la forme $X^{(k+1)} = A^{(k)} \cdot X^{(k)}$.

Comme les coefficients de la matrice sont recalculés à chaque itération, la fraction de la différence de ratio qui sera transférée entre les nœuds i et v peut varier au cours du temps (tenant éventuellement compte des caractéristiques du lien de communication entre i et v). La loi de diffusion étant supposée être conservative on a : $A_{iv}^{(k)} = A_{vi}^{(k)}$. D'autre part, dans le cas où il n'y pas de lien entre deux nœuds i et v, les coefficients correspondants sont fixés à zéro. Enfin, chaque coefficient de la diagonale, représentant la fraction de la différence de ratio qui n'induira pas de migration de tâches doit vérifier : $A_{ii}^{(k)} = 1 - \sum_{v \neq i} A_{iv}^{(k)}$, $A_{ii}^{(k)} \geq 0$. Le résultat est que $A^{(k)}$ est une matrice symétrique, doublement stochastique.

La forme de la matrice conditionne fortement le comportement de notre approche. En pratique, la matrice est construire grâce à l'algorithme 5.3, celui-ci utilise pour le calcul des éléments non diagonaux la méthode suggérée par Boillat [114].

5.3 Validité de l'approche

Dans cette section, nous allons tout d'abord prouver mathématiquement la convergence de notre algorithme d'équilibrage de ratio. La convergence sera ensuite mise en évidence expérimentalement, en considérant différentes topologies réseaux.

Algorithme 5.3 Calcul de la matrice de diffusion $A^{(k)}$

1: **for all** i tel que $1 \leq i \leq N$ **do**
2: **for all** v tel que $1 \leq v \leq N$ **do**
3: **if** $v \in V_i^{(k)}$ **then**
4: $A_{iv}^{(k)} = \frac{1}{\max(d(i),d(v))+1}$
5: **else**
6: $A_{iv}^{(k)} = 0$
7: **end if**
8: **end for**
9: $A_{ii}^{(k)} = 1 - \sum_{v=1, v\neq i}^{N} A_{iv}^{(k)}$
10: **end for**

5.3.1 Preuve mathématique

Remarque 1 *Par construction, les matrices de diffusion $A^{(k)}$ sont symétriques, doublement stochastiques.*

Condition 2 *Le graphe de connexion est supposé infiniment souvent connecté, i.e. qu'il existe une sous-séquence $\{k_p\}_{p\in\mathbb{N}}$ telle que aux temps k_p le graphe de connexion est connecté.*

Théorème 3 *Sous la condition précédente, l'algorithme 5.1 converge vers le vecteur uniforme $X^* = (x^*, \ldots, x^*)$.*

Preuve 1 *Considérons*

$$f_i^{(k)}\left(x_1^{(k)}, \ldots, x_N^{(k)}\right) = x_i^{(k)} + \sum_{v\neq i} A_{iv}^{(k)} \cdot \left(x_v^{(k)} - x_i^{(k)}\right)$$

on obtient alors successivement

$$\begin{aligned}\left|f_i^{(k)}\left(X^{(k)}\right)\right| &= \left|x_i^{(k)} + \sum_{v\neq i} A_{iv}^{(k)} \cdot \left(x_v^{(k)} - x_i^{(k)}\right)\right| \\ &\leq \left|\left(1 - \textstyle\sum_{v\neq i} A_{iv}^{(k)}\right) \cdot \max_i x_i^{(k)} + \max_i x_i^{(k)} \cdot \textstyle\sum_{v\neq i} A_{iv}^{(k)}\right| \\ &\leq \max_i \left|x_i^{(k)}\right|\end{aligned}$$

En notant la norme maximale par $\left|X^{(k)}\right|_\infty = \max_i \left|x_i^{(k)}\right|$ on a :

$$\left|X^{(k)}\right|_\infty \leq \left|X^{(k-1)}\right|_\infty \leq \cdots \leq \left|X^{(0)}\right|_\infty \ ,$$

aussi la séquence $\left\{\left|X^{(k)}\right|_\infty\right\}_k$ (et a fortiori $\left\{\left|X^{(k)}\right|_\infty\right\}_{k_p}$) est bornée.

Le théorème de Bolzano-Weierstrass implique qu'elle contient une sous-séquence convergente. Sans perte de généralité, on suppose que

$$\lim_{k_p \to +\infty} X^{(k_p)} = X^*$$

De plus, la sous-séquence $\left\{\left|X^{(k)}\right|_\infty\right\}_k$ *converge car elle est monotone décroissante, d'où :*

$$\lim_{k\to+\infty}\left|X^{(k)}\right|_\infty = \lim_{k_p\to+\infty}\left|X^{(k_p)}\right|_\infty = X^*$$

La condition 2 implique que la suite de matrices $A^{(k_p)}$ *est convergente, et comme il s'agit de matrices doublement stochastiques on a :*

$$\lim_{k_p\to+\infty}\left(A^{(k_p)}\right)^{k_p} = \begin{pmatrix} \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \ddots & \vdots \\ \frac{1}{N} & \cdots & \frac{1}{N} \end{pmatrix}$$

et

$$\left|Q\cdot X^* - \left(A^{(k_p)}\right)^{(k_p)}\cdot X^{(k_p)}\right|_\infty \leq$$
$$\left|\left(Q - \left(A^{(k_p)}\right)^{(k_p)}\right)\cdot X^*\right|_\infty + \left|\left(A^{(k_p)}\right)^{(k_p)}\cdot\left(X^* - X^{(k_p)}\right)\right|_\infty$$

Alors

$$\lim_{k_p\to+\infty}\left|Q\cdot X^* - \left(A^{(k_p)}\right)^{(k_p)}\cdot X^{(k_p)}\right|_\infty = 0$$

Un résultat classique pour les matrices non négatives [115] garantit que $A^{(k_p)}\cdot Q = Q\cdot A^{(k_p)} = Q$*. Grâce à ce résultat on peut établir que :*

$$\lim_{k_p\to+\infty}\left|Q\cdot X^* - X^{(k_p+1)}\right|_\infty = 0 \ .$$

En effet, on a :

$$\left|Q\cdot X^* - \left(A^{(k_p)}\right)^{(k_p)}\cdot X^{(k_p)}\right|_\infty =$$
$$\left|\left(A^{(k_p)}\right)^{(k_p-1)}\cdot\left(Q\cdot X^* - X^{(k_p+1)}\right)\right|_\infty$$

Par conséquent

$$|Q\cdot X^*|_\infty = \lim_{k_p\to+\infty}\left|X^{(k_p+1)}\right|_\infty = |X^*|_\infty$$

Comme $\left|x_i^{(k+1)} - x_i^*\right| = f_i^{(k)}\left(X^{(k)} - X^*\right) \leq \max_i\left|x_i^{(k)} - x_i^*\right|$ *on obtient :*

$$0 \leq \left|X^{(k+1)} - X^*\right|_\infty \leq \left|X^{(k_p)} - X^*\right|_\infty$$

Le développement précédent montre que la séquence $\left\{\left|X^{(k)} - X^*\right|_\infty\right\}_k$ *est monotone décroissante, et donc convergente. Finalement :*

$$\lim_{k\to+\infty}\left|X^{(k)} - X^*\right|_\infty = \lim_{k_p\to+\infty}\left|X^{(k_p)} - X^*\right|_\infty = 0$$

ce qui signifie que $\lim_{k\to+\infty} X^{(k)} = X^* = x^*\cdot(1,\ldots,1)^T$ *où* $x^* = c$ ■.

5.3.2 Validation expérimentale

Appliquons l'algorithme 5.1 à un réseau de 5 nœuds numérotés de 0 à 4, le réseau étant supposé complètement connecté (cela implique que pour tout i, v et k on a $A_{iv}^{(k)} = \frac{1}{5}$). La numérotation des nœuds commence à zéro pour des raisons pratiques (rappelons qu'on utilise le simulateur OMNeT++).

Nous faisons également l'hypothèse que les nœuds sont hétérogènes, avec deux types de nœuds :

1. nœud capteur dont les capacités sont limitées ;
2. nœud de type ordinateur portable ayant des capacités énergétique et de calcul supérieures.

La figure 5.1 décrit un tel réseau, il est composé de deux nœuds de type 1 et trois de type 2. On a donc $i \in \Omega = \{0, \ldots, 4\}$ et $\forall i, V_i^{(k)} = \Omega - \{i\}$ pour tout k.

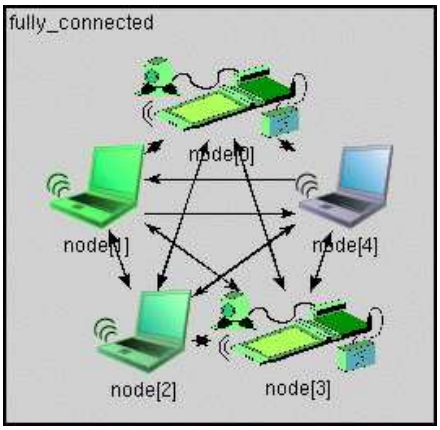


FIG. 5.1 – Réseau considéré.

D'autre part, chaque nœud est supposé pouvoir traiter des tâches provenant de trois classes de tâches distinctes, sachant qu'à une classe de tâches correspond un coût énergétique pour exécuter une tâche élémentaire de ce type. Cela signifie donc que $M = 3$ et $j \in \{1, 2, 3\}$:

- classe $T_{i,1}$, le coût énergétique pour exécuter une de ses tâches est $e_{i,1} = 10$;
- classe $T_{i,2}$, le coût énergétique pour exécuter une de ses tâches est $e_{i,2} = 20$;
- classe $T_{i,3}$, le coût énergétique pour exécuter une de ses tâches est $e_{i,3} = 30$.

Le nombre de tâches de chaque classe dont dispose chaque nœud est défini de telle sorte que l'on ait pour un nœud de type 1 un total de 50 tâches (réparties aléatoirement dans l'une des trois classes). Tandis qu'un nœud de type 2 aura le double de tâches, soit $\sum_{j=1}^{3} T_{i,j} = 100$ pour i de type 2. Le nombre total de tâches, sans tenir compte de la classe, est donc :

$$\sum_{i=0}^{4} \left(\sum_{j=1}^{3} T_{i,j} \right) = 400$$

Nœud i	Type	Énergie $E_i^{(0)}$	Ratio $x_i^{(0)}$	$T_{i,1}^{(0)}$	$T_{i,2}^{(0)}$	$T_{i,3}^{(0)}$
0	1	1634,02	0,610534	19,0648	12,1081	18,8271
1	2	4499,08	0,412993	44,0023	26,1865	29,8112
2	2	2614,76	0,828776	4,3860	74,5274	21,0866
3	1	1501,18	0,625097	18,5608	19,0401	12,3991
4	2	664,415	3,31985	18,2312	42,9619	38,8069

TAB. 5.2 – Valeurs initiales des différents nœuds du réseau.

La table 5.2 présente pour chaque nœud son énergie, son ratio et la répartition des tâches par classe, tous à l'instant $k = 0$.

Supposons maintenant qu'au cours du temps l'énergie de chaque nœud et le nombre global de tâches sur le réseau n'évolue pas, c'est-à-dire qu'il n'y a ni consommation, ni production de tâches. Connaissant les valeurs initiales de chaque nœud du réseau on peut alors aisément calculer le ratio x^* vers lequel doit converger le ratio de chaque nœud. En effet, le réseau peut être vu comme un méta-nœud dont le nombre de tâches est égal à la somme de toutes les tâches présentes sur les différents nœuds, alors que son énergie est égale à la somme de l'énergie disponible à travers le réseau. C'est pourquoi, le point fixe recherché satisfait :

$$x^* = \frac{\sum_{i=0}^{4} \left(\sum_{j=1}^{3} T_{i,j}^{(0)} e_{i,j} \right)}{\sum_{i=0}^{4} E_i^{(0)}} \tag{5.20}$$

Si on utilise les valeurs numériques de la table 5.2 on aboutit à : $x^* \approx 0,74833$

Les expérimentations menées confirment la validité de notre approche. Plus précisément, comme le montrent les courbes d'évolution des ratios présentées dans les figures 5.2, 5.3 et 5.4, l'algorithme d'équilibrage de ratio converge systématiquement. Ces figures correspondent respectivement à une topologie de type graphe complet (l'exemple considéré), linéaire et mixte, avec dans les deux premiers cas 5 nœuds et 20 nœuds dans le dernier cas. La table 5.3 présente les ratios obtenus dans le cadre du réseau complètement connecté, i.e. celui de la figure 5.1, pour deux valeurs d'epsilon ($\epsilon = 10^{-4}$ et 10^{-8}).

Nœud i	Ratio $x_i^{(k)}$ à $\epsilon = 10^{-4}$	Ratio $x_i^{(k)}$ à $\epsilon = 10^{-8}$
0	0,748492	0,748334
1	0,748107	0,748332
2	0,748491	0,748334
3	0,748492	0,748334
4	0,748492	0,748334

TAB. 5.3 – Ratios trouvés pour le réseau de 5 nœuds complètement connecté.

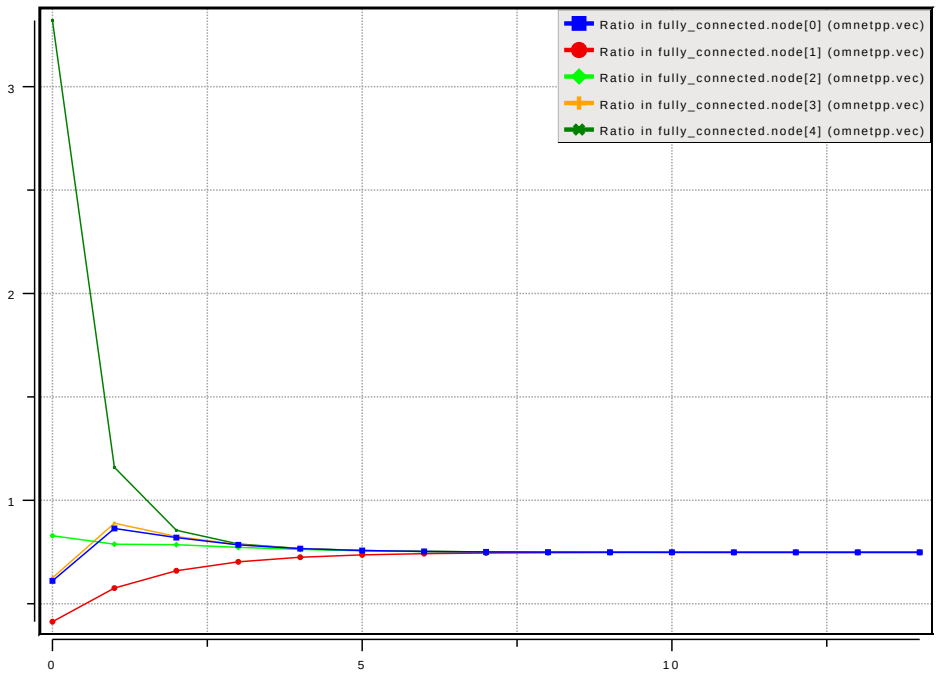


FIG. 5.2 – Courbe d'évolution des ratios suivant le nombre d'itérations dans le cas du réseau complètement connecté.

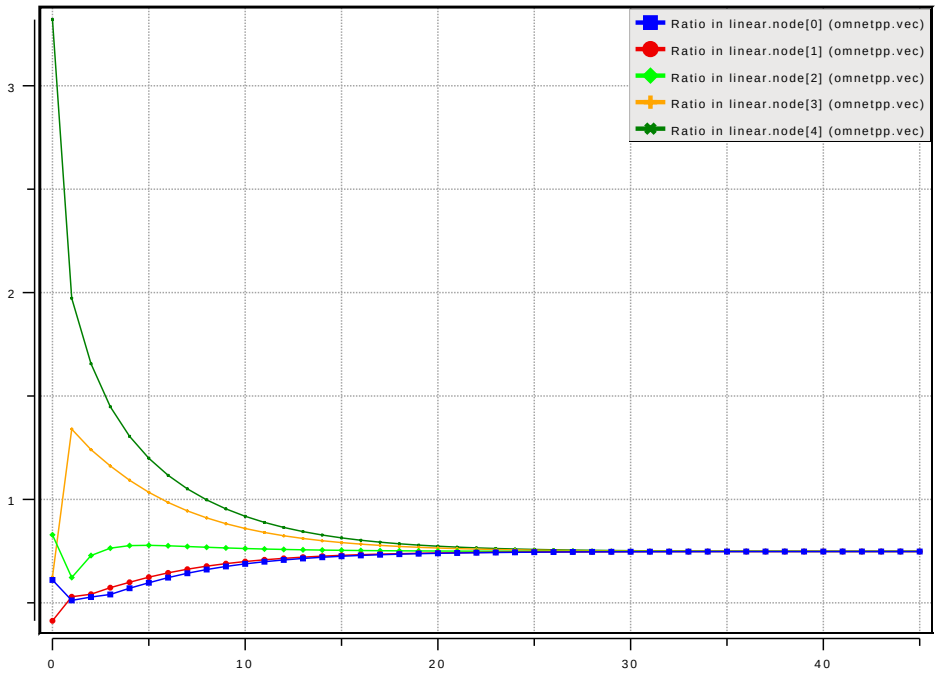


FIG. 5.3 – Courbe d'évolution des ratios suivant le nombre d'itérations dans le cas du réseau linéaire.

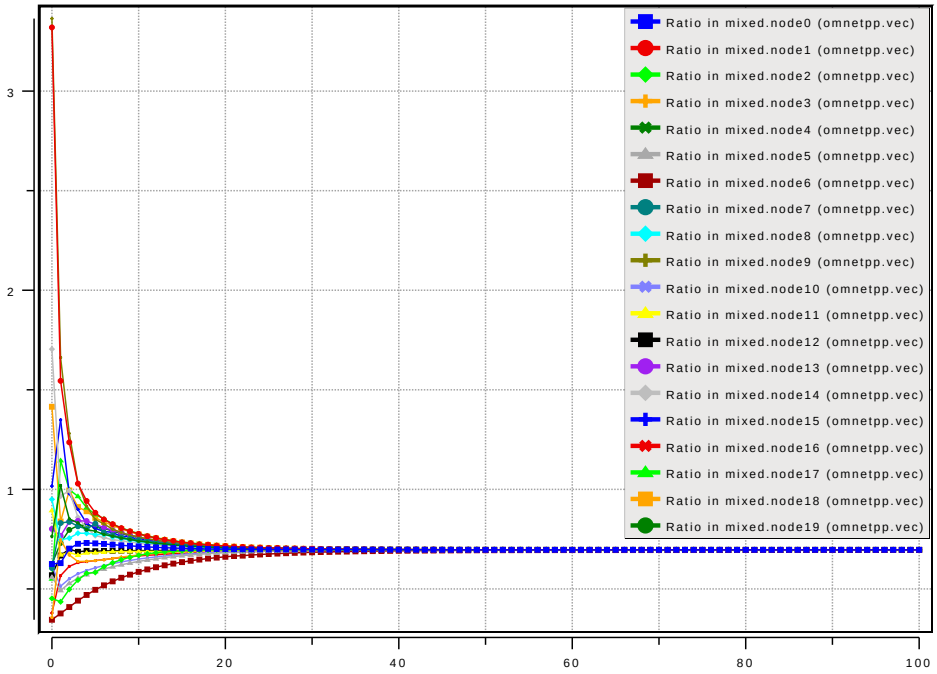


FIG. 5.4 – Courbe d'évolution des ratios suivant le nombre d'itérations dans le cas du réseau mixte.

Chapitre 6

Simulation de réseaux statiques

6.1 Description du scénario considéré

On considère un réseau de nœuds capteurs hétérogènes dont la vocation est la surveillance d'un périmètre, avec pour objectif la détection de toute intrusion. Chaque nœud est supposé surveiller une partie de la zone d'intérêt au moyen d'un capteur multimédia (webcam). Les données produites et à traiter consistent en des images de résolution 320 × 240 pixels en 24 couleurs prises à intervalle régulier. Les tâches associées à ces données et qui devront être exécutées par les différents nœuds sont de type compression d'image, détection de mouvements à partir d'un couple d'images, stockage d'image, etc.

Nous avons retenu ce scénario, car grâce aux travaux de Margi *et al.* [17, 116] nous disposions d'une analyse fine et précise de la consommation énergétique de deux plateformes pouvant jouer le rôle de nœud capteur multimédia. Il s'agit d'une part d'un nœud *Stargate* SPB-400 commercialisé par Crossbow, d'autre part d'un ordinateur portable de type DELL Latitude C600 (voir la figure 6.1). Pour les communications, nous faisons l'hypothèse que le réseau sans fil est de type WiFi, norme 802.11b. En effet, tous les nœuds utiliseront une carte PCMCIA Lucent *WaveLAN Silver* (11 Mbps) pour laquelle un modèle de consommation énergétique est disponible.

(a) Stargate SPB-400

(b) Ordinateur portable DELL Latitude C600

FIG. 6.1 – Nœuds capteurs multimédias.

6.2 Implémentation de l'algorithme

Dans cette section, nous allons tout d'abord décrire le principe de la méthode retenue pour discrétiser les coefficients $\alpha_{i,v,j}^{(k)}$. Nous nous intéresserons ensuite aux aspects énergétiques, en commençant par l'énergie consommée par les communications, puis celle résultant de l'exécution de tâches sur chaque type de nœud.

6.2.1 Discrétisation de la migration de tâches

Toutes les méthodes de calcul du nombre de tâches à échanger entre nœuds voisins, présentées précédemment dans la sous-section 5.2.1 induisent des valeurs réelles. On peut aisément comprendre qu'envoyer $1,35$ tâches de la classe x n'est pas réaliste. Une procédure de discrétisation des coefficients $\alpha_{i,v,j}^{(k)} = A_{iv}^{(k)} \cdot \gamma_{i,v,j}^{(k)}$, $j \in \{1,\ldots,M\}$ est donc requise. D'ailleurs, cette procédure a été explicitement prévue dans l'algorithme 5.2 (ligne numéro 4). On rappelle que la valeur discrète de $\alpha_{i,v,j}^{(k)}$ est notée $\beta_{i,v,j}^{(k)}$.

La discrétisation se fait en deux étapes. Une première étape consiste à arrondir chaque coefficient $\alpha_{i,v,j}^{(k)}$ à la valeur entière la plus proche. Dans un second temps, on compare la valeur discrète de la somme des coefficients réels à la somme de leurs équivalents discrets. On peut ainsi savoir si la discrétisation aboutit à l'envoi de trop tâches ou inversement conduit à un manque de tâches échangées. En conséquence, certains coefficients discrets sont « corrigés » de manière à ce que les deux sommes soient égales. Cette correction revient à incrémenter ou à décrémenter le nombre de tâches à envoyer.

Formellement, la procédure de discrétisation se décrit comme suit :

1. Discrétisation par arrondi des coefficients $\alpha_{i,v,j}^{(k)}$
   - $\beta_{i,v,j}^{(k)} = \lceil \alpha_{i,v,j}^{(k)} \rceil$ si $\alpha_{i,v,j}^{(k)} - \lfloor \alpha_{i,v,j}^{(k)} \rfloor > 0,5$ et $T_{i,j}^{(k)} \geq \lceil \alpha_{i,v,j}^{(k)} \rceil$;
   - $\beta_{i,v,j}^{(k)} = \lfloor \alpha_{i,v,j}^{(k)} \rfloor$ sinon.

   On calcule également pour chaque coefficient l'écart entre sa valeur réelle et son équivalent discret :
   $$\delta_{i,v,j}^{(k)} = \beta_{i,v,j}^{(k)} - \alpha_{i,v,j}^{(k)}$$
   Les coefficients $\delta_{i,v,j}^{(k)}$ seront utilisés dans la seconde étape pour déterminer le ou les coefficient(s) discret(s) qui sera / seront « corrigé(s) ».

2. Correction éventuelle des coefficients discrets On compare la somme arrondie des nombres réels de tâches à la somme de leurs équivalents discrets :
   $$\Gamma_{i,v}^{(k)} = \texttt{round}\left(\sum_{j=1}^{M} \alpha_{i,v,j}^{(k)}\right)$$
   $$\Delta_{i,v}^{(k)} = \sum_{j=1}^{M} \beta_{i,v,j}^{(k)}$$

Dans le cas où $\Gamma_{i,v}^{(k)} \neq \Delta_{i,v}^{(k)}$ alors un coefficient $\beta_{i,v,j}^{(k)}$ est « corrigé » de la façon suivante :

– si $\Delta_{i,v}^{(k)} < \Gamma_{i,v}^{(k)}$ alors

$$\beta_{i,v,J}^{(k)} = \beta_{i,v,J}^{(k)} + 1 \quad \text{où } J \text{ est tel que } \delta_{i,v,J}^{(k)} < 0,\ |\delta_{i,v,J}^{(k)}| = \max_{\delta_{i,v,j}^{(k)}<0} \left(\left| \delta_{i,v,j}^{(k)} \right| \right), \\ j \in \{1, \ldots, M\} \text{ et } T_{i,J}^{(k)} > \beta_{i,v,J}^{(k)}$$

– si $\Delta_{i,v}^{(k)} > \Gamma_{i,v}^{(k)}$ alors

$$\beta_{i,v,J}^{(k)} = \beta_{i,v,J}^{(k)} - 1 \quad \text{où } J \text{ est tel que } \delta_{i,v,J}^{(k)} > 0,\ |\delta_{i,v,J}^{(k)}| = \max_{\delta_{i,v,j}^{(k)}>0} \left(\left| \delta_{i,v,j}^{(k)} \right| \right), \\ j \in \{1, \ldots, M\} \text{ et } T_{i,J}^{(k)} > 0$$

6.2.2 Consommation énergétique d'une carte réseau sans fil de type WiFi

Nous commençons par situer le coût des communications au niveau de l'adaptateur sans fil par rapport à d'autres sources de consommation d'énergie. Les coûts externes à l'interface réseau seront négligés, i.e. le coût d'une tâche de calcul, d'une lecture sur disque, etc., seront abordés ultérieurement. Nous ferons ensuite un bref rappel de notions utiles de physique, suivi par la présentation de différentes manières de calculer le coût énergétique des communications.

Les sources de consommation

De nombreux travaux ont mis en évidence le fait que les communications ne sont pas les opérations qui font le plus consommer d'énergie par l'interface réseau. En effet, Cano et Manzoni [117] ont montrés qu'en définissant un algorithme qui coupe de manière dynamique l'interface radio (le composant RF) lorsque le nœud n'est pas un interlocuteur d'une transmission, on peut économiser entre 25% et 60% de l'énergie totale du nœud. Leur algorithme est basé sur le dialogue RTS/CTS qui implémente la « poignée de main » (mécanisme qui permet de réserver le canal de communication pour une communication point à point) de la norme IEEE 802.11. À noter que dans leur article ils considèrent l'adaptateur réseau 802.11 *WaveLAN Bronze* de Lucent, mais donnent également des indications relatives à l'ORiNOCO/IEEE Turbo 11 Mbps, voir ci-dessous pour leurs caractéristiques.

- *WaveLAN Bronze* de Lucent
  - débit de 2 Mbps ;
  - portée radio de 250 mètres ;
  - fréquence radio de 2,4 GHz ;
  - consomme 9 mA en veille (mode *sleep*) ;
  - 230 mA en réception ;
  - 330 mA en émission ;
  - tension d'alimentation de 5 V.

- ORiNOCO/IEEE Turbo 11 Mbps
  - débit de 11 Mbps ;
  - consomme 15 mA en veille (mode *sleep*) ;
  - 240 mA en réception ;
  - 280 mA en émission ;
  - tension d'alimention de 5 V.

Le scénario retenu pour les expérimentations est le suivant : 25 nœuds mobiles dispersés sur une zone de 500×500 mètres, parmi ces nœuds 20 génèrent des flux de données CBR (*Constant Bit Rate*) à raison de 4 paquets de données par seconde, chaque paquet ayant une taille de 512 octets. La durée d'une simulation est de 900 secondes.

Dans [118], les auteurs utilisent une stratégie similaire : ils mettent l'adaptateur réseau dans l'état / le mode *idle*, intelligement bien entendu. Ils identifient en effet trois sources de « gaspillage » de l'énergie :

1. l'état *idle* / en attente qui correspond à l'absence de transmissions. Par exemple, l'adaptateur *WaveLAN* de Lucent fonctionnant à une fréquence radio de 915 MHz consomme 1,15 W dans cet état, 1,2 W lors d'une réception et 1,6 W pour une émission ;
2. l'*overhearing*, i.e. la sur-écoute de paquets destinés à un autre nœud. Son coût est identique à celui d'une réception.
3. l'*Erroneous Carrier Sensing* (ECS) qui correspond à la réception d'un signal radio dont le rapport signal/interference n'est pas acceptable. En clair, il y a interférence de signaux, traduisant la présence d'une ou plusieurs collision(s).

Les expérimentations montrent que les transmissions ne représentent que de 3,4% à 6,8% de l'énergie consommée (dépend de la charge réseau), tandis que les trois états évoqués précédemment représentent de 81,8% à 88,7% de l'énergie consommée. Leur approche consiste à limiter l'*Erroneous Carrier Sensing*. Pour ce faire, ils définissent un protocole au niveau de la couche MAC. Ce protocole force l'interface réseau à passer dans un état *idle* basse consommation (il y a moins de composants électroniques alimentés en courant que dans l'état *idle* normal) dès détection d'un ECS. La durée de maintient dans l'état *idle* basse consommation est calculée dynamiquement à partir de la taille des paquets de la couche MAC. À noter que cette approche n'est valide que pour les cartes 802.11 de dernière génération, supportant l'état *idle* de basse consommation. Le tableau ci-après montre la puissance consommée par une carte de ce type, mettant en évidence la pertinence de cette approche. Les résultats expérimentaux montrent une réduction combinée du coût énergétique de l'*overhearing* et de l'ECS, et cela jusqu'à 62%, augmentant la durée de vie de réseau jusqu'à 154%. Le scénario de test est assez similaire à celui évoqué précédemment : 60 nœuds mobiles dispersés sur une zone de 550×120 mètres (les déplacements se font à une vitesse de 6,4 m/s), le flux de données est identique si ce n'est l'intervalle entre deux émissions de paquet qui varie de 0,3 secondes à 1,2 secondes. Un flux de données entre deux nœuds est produit pendant 500 secondes.

État adaptateur	Description	Consommation
Idle	en attente, prêt à recevoir ou émettre	66 mW
Receiving	réception de paquets destinés à soit même	594 mW
Overhearing	réception de paquets destinés à d'autres	594 mW
ECS	réception d'un signal de mauvaise qualité	594 mW
Transmitting	émission de paquets	924 mW (max.)

TAB. 6.1 – Consommation de la *Socket Communication Inc's Low Power Wireless LAN Compact Flash Card.*

De même, dans [119] il est clairement mis en évidence que dans le cadre d'un réseau ad hoc saturé, où les nœuds se disputent l'accès au médium de communication, ce sont les modes *passifs* de l'interface réseau qui consomment l'énergie, le mode *actif* n'ayant qu'un impact minime. Par modes passifs, les auteurs entendent les états *idle*, *overhearing*, *sensing* et *receiving*, l'état *transmitting* correspondant au mode actif. Dans cet article est développé et validé un modèle analytique de prédiction de la consommation d'énergie dans un réseau ad hoc saturé présentant énormément de collisions. Margi *et al.* montrent dans ce contexte que, d'une part le coût énergétique pour transmettre des données avec succès augmente linéairement avec la taille du réseau, d'autre part qu'il est plus intéressant d'échanger des paquets de données plus gros.

Notion de physique

En physique, l'énergie est une manière d'exprimer l'intensité des phénomènes, implicitement c'est une quantité mesurable. L'énergie se définit de différentes manières, variant suivant le phénomène considéré (réaction nucléaire, mouvement, etc.). L'unité du système international pour mesurer l'énergie est le joule (J). D'autres unités, plus spécifiques, sont utilisées, en particulier l'électron-volt (1 eV=$1,602 \times 10^{-19}$ J), la calorie (4,18 J), etc. En électricité, l'unité usuelle est le kilowatt-heure (1 kWh=3,6 MJ).

Une grandeur connexe est la puissance. La puissance consommée par un appareil est l'énergie qu'il consomme pendant sa durée de fonctionnement :

$$P = \frac{E}{t} \tag{6.1}$$

avec P en watts (W), E en joules (J) et t en secondes. En électricité la puissance est liée à la tension et à l'intensité du courant, soit :

$$P = U \times I \tag{6.2}$$

où U est en volts (V) et I en ampères (A) (1 mA=10^{-3} A, idem pour les watts).

En ce qui concerne les batteries, leur capacité est souvent exprimée en ampère-heure (Ah). Par exemple, une batterie Li-ion pour ordinateur portable d'une capacité de 3600 mAh et une tension de 10,8 V a une puissance nominale d'approximativement 39 Wh, soit une capacité énergétique de 140400 J.

Modélisations existantes

- **Approche considérée par Cano et Manzoni [120]**

Ce modèle s'inscrit dans le cadre des réseaux ad hoc mobiles (MANETs). Il consiste simplement à utiliser les équations (6.1) et (6.2) pour déterminer l'énergie consommée lors des communications (émissions et réceptions). En effet, à partir de l'équation (6.1) on a :

$$E = P \times t \tag{6.3}$$

on remplace ensuite P par le terme de l'équation (6.2) pour obtenir :

$$E = U \times I \times t \tag{6.4}$$

avec

$$t = \frac{\texttt{<Taille du paquet en octets>} \times 8}{\texttt{<Débit en bits par seconde>}}$$

On constate donc que l'énergie dissipée dépend des caractéristiques de l'adaptateur réseau sans fil, de la taille des données échangées et du débit du canal de communication.

À titre d'illustration, reprenons l'exemple de Cano et Manzoni : l'adaptateur réseau sans fil *WaveLan Bronze* de Lucent dont les caractéristiques ont été données précédemment :

$$E_{\texttt{RX}}(<\ \texttt{Paquet}\ >) = 5 \times 230 \times \frac{\texttt{<Taille du paquet en octets>} \times 8}{2 \times 10^6}$$

$$E_{\texttt{TX}}(<\ \texttt{Paquet}\ >) = 5 \times 330 \times \frac{\texttt{<Taille du paquet en octets>} \times 8}{2 \times 10^6}$$

en considérant que les énergies résultantes sont en milli-Joules et où `RX`, `TX` dénotent respectivement une réception et un envoi.

- **Approche considérée par Feeney et Nilsson [121, 122]**

Feeney *et al.* décrivent une série d'expérimentations menées pour mesurer précisément la consommation énergétique d'un adaptateur sans fil dans un réseau ad hoc. Le résultat est un ensemble d'équations linéaires permettant de calculer la consommation lors de l'émission, la réception ou la non prise en compte de paquets du fait de la sur-écoute. À noter que les auteurs font la distinction entre communication par diffusion et communication point à point. Cette distinction apparaît dans peu de modèles de consommation énergétique. Naturellement, les valeurs numériques des coefficients dans les différentes équations sont spécifiques aux adaptateurs considérés, puisqu'elles résultent de mesures directes.

Les adapteurs choisis sont de type *WaveLAN* de Lucent. Ils fonctionnent à la fréquence radio de 2,4 GHz et se différencient par le débit offert : 2 Mbps pour la version *Bronze* et 11 Mbps pour la version *Silver*. Le tableau 6.2 reporte les caractéristiques de ces deux adaptateurs. Les résultats obtenus permettent de tirer plusieurs enseignements, parmi lesquels on retrouve certains des points mis en exergue dans des travaux évoqués précédemment :

2 Mbps	Conso. mesurée	Conso. théorique
Sleep mode	14 mA	9 mA
Idle mode	178 mA	
Receive mode	204 mA	280 mA
Transmit mode	280 mA	330 mA
11 Mbps	Conso. mesurée	Conso. théorique
Sleep mode	10 mA	10 mA
Idle mode	156 mA	
Receive mode	190 mA	180 mA
Transmit mode	284 mA	280 mA
Tension d'alimentation	4.74 V	5 V

TAB. 6.2 – Caractéristiques des adaptateurs *WaveLAN* de Lucent.

- il n'y a pas de lien direct entre consommation d'énergie et débit ;
- du fait de la forte contention, les paquets de petite taille sont très coûteux ;
- le lien entre vitesse de transmission et coût énergétique est très complexe ;
- l'état *idle* a un coût aussi élevé qu'une réception, surtout que dans le cadre d'un réseau ad hoc les nœuds devraient en principe toujours être prêt à recevoir des données, et donc écouter le canal de communication ;
- enfin, plus la densité de nœuds est élevée, plus les communications seront coûteuses en énergie.

La modélisation de la consommation énergétique par paquet lors de l'envoi, la réception ou la non prise en compte de celui-ci consiste en des équations linéaires du type :

$$E = m \times \texttt{<Taille du paquet en octets>} + b \tag{6.5}$$

Ainsi, il y a une partie fixe modélisant le coût du changement d'état et de l'accès au médium de communication, plus une partie évoluant proportionnellement à la taille du paquet. Les résultats expérimentaux mettent en évidence la pertinence de cette approche, permettant de fixer les valeurs des coefficients m et b pour les différents états / modes. Ce modèle ne prend pas en compte le coût énergétique dans le cas d'un échec de l'accès au médium de communication (contention), de la perte de message en raison de collisions, des erreurs de transmission, de la disparition de la connectivité du réseau. En définitive, des équations sont définies pour calculer le coût associé à une communication par diffusion, point à point, ainsi que pour le coût de la sur-écoute et d'un mode spécifique (auquel on ne s'intéressera pas) dans lequel le nœud écoute tout le trafic réseau qu'il capte.

Pour ce qui est du protocole expérimental, le trafic réseau considéré consiste en des paquets UDP produits à une fréquence de 10 à 20 paquets par seconde, tandis que les interférences éventuelles ont été réduites au minimum. Enfin, au niveau physique l'énergie consommée a été déterminée en mesurant directement la tension d'entrée et l'intensité du courant consommé par l'adaptateur réseau. À partir des mesures effectuées dans les différents modes les coefficients des équations sont obtenus par régression linéaire.

① Communication par diffusion (multicast) - *Broadcast traffic*

Quand un nœud veut procéder à une communication par diffusion, voici comment il procède :

1. il commence par écouter le canal ;
2. s'il ne détecte aucune communication en cours, il envoie son message. Dans le cas contraire le nœud tente à nouveau sa chance à l'issue d'un délai d'attente tiré aléatoirement, utilisant l'algorithme de retour aléatoire (*backoff algorithm*).

Deux équations sont définies pour modéliser le coût d'une communication par diffusion, distinguant le coût associé à une émission de celui d'une réception :

$$E_{\texttt{broadcast\_send}} = m_{\texttt{send}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{send(bcast)}}$$
$$E_{\texttt{broadcast\_recv}} = m_{\texttt{recv}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{recv(bcast)}}$$

À noter que la communication par diffusion dans le cadre d'un réseau WiFi en mode ad hoc (accès au médium grâce à la fonction distribuée de coordination (DCF) basée sur le CSMA/CA) n'inclut pas de paquets de contrôle, contrairement à une communication point à point. Par conséquent, on ne peut pas réserver le canal de communication, il n'y a donc pas de mécanisme permettant de se prémunir du problème du nœud caché et de réduire le temps perdu en cas de collision.

② Communication point à point (unicast) - *Point-to-point traffic*

Pour échanger un paquet de données par une communication point à point, les interlocuteurs (source et destination) commencent par échanger des paquets de contrôle RTS / CTS de manière à réserver le canal. On limite ainsi le problème du terminal caché et les collisions. Dès lors que le canal est réservé, les données sont envoyées par le nœud source et le nœud destinataire en accuse réception. Les paquets RTS et CTS contiennent dans leur en-tête la taille des données à échanger définissant implicitement la durée de l'échange. Dans le cas où un nœud ne reçoit pas de paquet CTS en réponse au paquet RTS qu'il a émis, il peut ré-émettre ce dernier. Tout nœud détectant un échange RTS / CTS ne doit pas communiquer pendant la durée du dialogue (*Network Allocation Vector*). Ce mécanisme de détection virtuelle de la porteuse réduit les collisions, mais ne les élimine pas. La figure 6.2 décrit les paquets échangés entre deux nœuds lors d'une communication point à point.

Remarque : de nombreux travaux critiquent la norme IEEE 802.11 dans le cadre d'un environnement multi-sauts (on parle également de multi-bonds). Ceci pour plusieurs raisons : le problème du terminal caché n'est que partiellement résolu ; le problème du terminal exposé n'est pas pris en compte ; enfin l'accès au canal est inéquitable.

Les équations sont :

$$E_{\texttt{point-to-point\_send}} = m_{\texttt{send}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{send(p2p)}}$$
$$E_{\texttt{point-to-point\_recv}} = m_{\texttt{recv}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{recv(p2p)}}$$

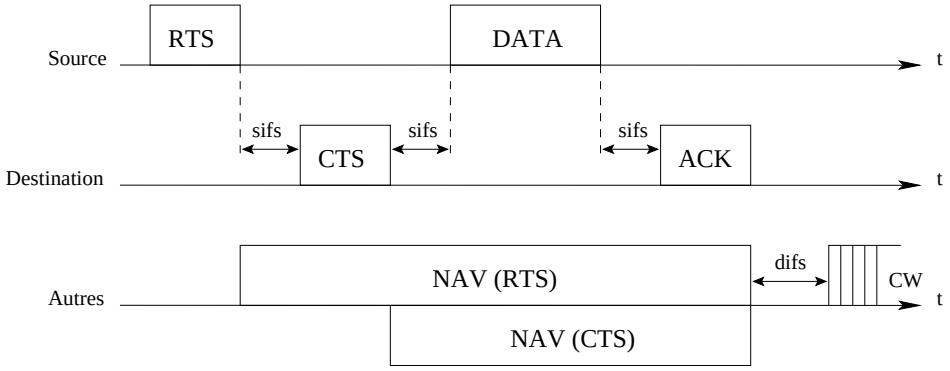


FIG. 6.2 – Paquets échangés dans le cadre d'une communication point à point.

La différence par rapport à une communication par diffusion se situe au niveau des coefficients b. Ce sont ces coefficients qui doivent refléter la différence au niveau du contrôle de l'accès au canal de communication.

Le protocole de contrôle est très coûteux pour de petits paquets. Aussi, pour des paquets plus petits qu'une taille fixée, l'émetteur écoute simplement le canal avant d'envoyer le paquet de données, idem pour le récepteur lors de l'envoi de l'acquittement. Cette approche est bien entendu sensible au problème du terminal caché, la sensibilité augmentant avec la taille du paquet. Les deux équations précédentes sont alors remplacées par les suivantes :

$$E_{\texttt{point-to-point\_send}} = m_{\texttt{send}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{send}(<\texttt{thresh})}$$
$$E_{\texttt{point-to-point\_recv}} = m_{\texttt{recv}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{recv}(<\texttt{thresh})}$$

Les valeurs des coefficients b devraient être comprises entre celles d'une communication par diffusion et celles du cas point à point.

③ **Sur-écoute** - *Discard traffic*

Il s'agit de déterminer le coût énergétique de la réception de paquets lors d'une communication point à point au niveau des nœuds qui ne sont pas le destinataire. En effet, par exemple un nœud à portée radio de la source de l'échange recevra tous les paquets émis par cette dernière. La sur-écoute est un point important car c'est elle qui est à l'origine de la majorité de la consommation énergétique de l'adaptateur réseau. Le coût énergétique d'une communication point à point sur un nœud qui n'est pas un des deux interlocuteurs revient à modéliser un des trois cas possibles :

1. le nœud est à portée radio de la source et de la destination ;
2. le nœud considéré est à portée radio de la source, mais pas de la destination ;
3. le nœud est à portée radio de la destination seule.

Chacun des trois cas se modélise par une équation :

$$E_{\texttt{discard}} = m_{\texttt{disc}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{non-dest}(S,D)}$$
$$E_{\texttt{discard}} = m_{\texttt{disc}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{non-dest}(S,\not{D})}$$
$$E_{\texttt{discard}} = m_{\texttt{none}} \times \texttt{<Taille du paquet en octets>} + b_{\texttt{non-dest}(\not{S},D)}$$

où S et $\not S$ représentent respectivement le fait que le nœud considéré soit à portée radio ou non de la source, de même pour la destination avec D. Plusieurs points sont à noter :

- les coefficients $b_{\texttt{non-dest}}$ reflètent le coût des paquets du protocole MAC ;
- dans le cas où le nœud considéré est à portée radio de la source, le coefficient $m_{\texttt{disc}}$ défini le mode d'économie d'énergie adopté par l'adaptateur réseau :
  - si $m_{\texttt{disc}} > 0$, en particulier si $m_{\texttt{disc}} = m_{\texttt{recv}}$ cela signifie que le nœud reçoit effectivement le paquet de données avant de le jeter ;
  - si $m_{\texttt{disc}} = 0$ l'adaptateur est dans l'état *idle* pendant l'envoi des données ;
  - si $m_{\texttt{disc}} < 0$ il est évident que l'adaptateur économise en quelque sorte de l'énergie. L'efficacité de cette approche dépend du temps nécessaire à la transmission des données et du coût du changement d'état entre l'état basse consommation et l'état *idle* ;
  - le coefficient $m_{\texttt{none}}$ reflète le comportement de l'interface réseau lors de l'envoi des données par la source.

La table 6.3 récapitule pour l'adaptateur *Silver* les équations (le valeurs numériques des différents coefficients sont données) obtenues à l'issue des mesures. Les résultats sont conformes aux spécifications, en revanche pour l'adaptateur *Bronze* les équations exhibent une consommation moindre qu'attendu lors de l'échange des données. On constate également que le fait de passer d'un débit théorique de 2 Mbps à 11 Mbps n'induit pas une réduction de l'énergie d'un facteur 5, ni une augmentation du débit utile du même facteur. Une raison majeure est le fait que pour garantir l'inter-opérabilité de l'adaptateur 11 Mbps avec des adaptateurs plus « lents », de nombreux paquets sont en fait émis de manière à ce que ces derniers les reçoivent correctement (à des débits moindres). Cela est particulièrement vrai pour les paquets de contrôle, mais également dans le cas d'une communication par diffusion. En revanche, une communication point à point entre deux adaptateurs à 11 Mbps permet de tirer parti de l'augmentation du débit.

11 Mbps	$\mu W \cdot sec/byte$[1]		$\mu W \cdot sec$
point-to-point send (a)	0.48	$\times size$	+431
broadcast send (b)	2.1	$\times size$	+272
point-to-point recv (c)	0.12	$\times size$	+316
broadcast recv (d)	0.26	$\times size$	+50
	non-destination $n \in S, D$		
discard (f)	0.11	$\times size$	+66
	non-destination $n \in S, n \notin D$		
discard (h)	0.11	$\times size$	+42
	non-destination $n \notin S, n \in D$		
discard (j)	0	$\times size$	+38
idle (ad hoc mode) (k)	741 mW		

TAB. 6.3 – Modèle de consommation énergétique - *WaveLAN* Lucent *Silver*.

[1] en excluant la taille des en-têtes MAC et PLCP

Pour Feeney et Nilsson leur modélisation linéaire est valide car le coefficient de corrélation est de l'ordre de 0,99. De plus, les résultats ont été obtenus avec la mesure de 50 à 90 paquets et les coefficients calculés ont un écart-type inférieur généralement à 5%.

Notons que c'est ce modèle de consommation énergétique des communications que nous avons retenu pour effectuer les simulations.

• **Approche considérée par Margi et Obraczka** [123]

Cette dernière approche est similaire à celle de Cano et Manzoni [120], en fait elle l'étend de manière à modéliser d'autres états / modes que l'émission et la réception. Le modèle linéaire de Feeney et Nilsson est également évoqué dans cet article, ainsi qu'un autre modèle plus spécifique aux réseaux de capteurs [124]. Pour finir, remarquons que dans cet article on trouve également une description de la modélisation de la consommation énergétique telle qu'elle est implémentée dans différents simulateurs, à savoir GloMoSim / QualNet et NS-2. Pour les auteurs, le modèle utilisé par QualNet est peu réaliste, celui de NS-2 est meilleur, mais présente également plusieurs manques :

– l'état en veille (*sleeping*) n'est pas pris en compte ;
– l'état en attente (*idle*) a par défaut un coût nul.

Margi et Obraczka ont défini ce modèle pour comparer l'impact énergétique de protocoles MAC. Il est utilisé pour comparer les performances énergétiques des protocoles IEEE 802.11 DCF et S-MAC [30]. À noter que ce dernier protocole prend en compter l'aspect économie d'énergie et a été défini plus spécifiquement pour les réseaux de capteurs. Plus précisément, ce modèle tient explicitement compte du mode basse consommation de l'adaptateur réseau. L'objectif est d'avoir un modèle aussi proche que possible de la réalité, i.e. modélisant la consommation énergétique de tous les modes / états possibles de la carte réseau. Pour le reste, le modèle est identique à celui de Cano et Manzoni :

$$E_y = \alpha_y \times t_y \tag{6.6}$$

où y dénote un état de l'adaptateur, α_y la puissance consommée par ce dernier par unité de temps et t_y le temps passé dans l'état considéré. Le temps dépend bien entendu de la taille du paquet et du débit de l'interface réseau.

Ce modèle est implémenté au niveau de la couche physique. Il peut donc être utilisé par n'importe quelle couche de niveau supérieur. Enfin, ce modèle a été validé en comparant des résultats analytiques obtenus pour IEEE 802.11 DCF et S-MAC à des résultats de simulations. Les résultat analytiques ont été obtenus en considérant l'interface radio TR1000, celle-ci est adaptée à des communications courte distance, supportant des débits allant jusqu'à 115,2 Kbps.

6.2.3 Consommation énergétique des nœuds du réseau

On s'intéresse maintenant à la consommation énergétique de plateformes matérielles pouvant jouer le rôle de nœud capteur. Il s'agit d'une part d'un ordinateur

portable [116], d'autre part de la carte Crossbow Stargate utilisée comme nœud capteur multimédia [17]. À partir des mesures présentées on peut en effet définir une simulation réaliste. En l'occurrence, il s'agirait d'un réseau de capteurs surveillant un périmètre géographique. L'intérêt de ces travaux est qu'ils mesurent non seulement le coût énergétique de tâches élémentaires de communication, mais également celui d'autres tâches telles que le traitement de données (calculs) ou un accès au disque dans le cas de l'ordinateur, ou en mémoire pour le Crossbow Stargate.

Consommation d'une plateforme de type ordinateur portable

Les tâches considérées comme élémentaires consistent en :
- du calcul ;
- une opération d'entrée / sortie (disque, affichage, etc.) ;
- une communication sans fil (émission ou réception).

La consommation énergétique de plusieurs combinaisons de tâches pré-citées est également mesurée. Le coût énergétique d'une tâche du jeu de tests est obtenu en mesurant, grâce à l'ACPI, la décharge de la batterie. L'objectif n'est pas seulement d'avoir des éléments d'information sur la consommation, mais également de pouvoir étudier des stratégies permettant de la réduire. Par exemple, dans le cadre de calculs distribués de déterminer s'il est plus intéressant de faire les calculs localement ou de les déporter sur une autre machine, ceci en tenant compte de plusieurs critères :
- l'énergie disponible sur les machines ;
- du coût énergétique des calculs locaux / déportés ;
- du coût du transfert des données dans le cas du choix de l'approche calculs distants.

Notre algorithme s'inscrit évidemment dans ce contexte.

① Jeu de tests de consommation énergétique

Il est constitué de 4 types de tâches élémentaires :

1. une tâche dite de base, qui sert de référence ;
2. une tâche de calcul intensif ;
3. une tâche d'accès intensif aux données sur disque ;
4. une tâche de communication intensive (envoi ou réception).

L'impact des composants graphiques (carte vidéo et écran LCD) est également considéré en étudiant l'effet de l'affichage.

– Tâche de base

Mesure la consommation énergétique induite par l'absence d'activité, seules des tâches systèmes basiques sont exécutées. C'est en quelque sorte l'état *idle*, il sert de référence pour tous les tests. Enfin, il est à noter que dans cet état / test, l'interface réseau est désactivée.

– Calcul intensif

Ce test utilise le benchmark FFT qui fait partie de la suite de tests SPEC CPU2000. FFT, pour *Fast Fourier Transform*, est un algorithme de calcul de la transformée de Fourier discrète et de son inverse.

– **Accès disque intensif**

Utilise `IOzone`, un logiciel évaluant les performances de système de fichiers. Deux types de tests sont réalisés : lecture seule et écriture seule d'un fichier de 3 Go. Le test de lecture consiste à lire et re-lire le fichier, tandis que l'écriture correspond à la création de nouveaux fichiers et à la recopie de fichiers existants.

– **Communication intensive**

L'outil utilisé est `Iperf`, celui-ci permet de mesurer la bande passante TCP disponible. `Iperf` est utilisé en mode client pour la transmission, en mode serveur pour la réception. Dans les deux cas l'outil a été configuré pour générer un trafic réseau UDP à un débit de 10 Mbps.

– **Affichage**

`xset` permet de désactiver à la demande la carte vidéo et l'écran LCD.

② **Étude de cas -** DELL LATITUDE C600

– **Caractéristiques**
  – Processeur Pentium III (Cœur Coppermine - 256 Ko de cache) à 750 MHz ;
  – mémoire vive : 256 Mo de RAM ;
  – disque dur d'une capacité de 20 Go ;
  – carte réseau sans fil Cisco Aironet 350 ;
  – batterie Li-Ion : tension = 14,8 VDC, capacité de 59 Wh ;
  – systèmes d'exploitation : Linux Debian (kernel 2.6.1).

– **Résultats des tests**

  Le tableau ci-dessous donne la consommation moyenne (et son écart-type) associée à chaque tâche du jeu de tests. On constate que le calcul est évidemment la tâche la plus consommatrice en énergie, suivie des accès disques. Pour les auteurs ce résultat est surprenant car ils pensaient que les accès disques seraient les tâches les plus coûteuses. D'autre part, il est à noter que les accès disques ont un coût très proche de celui des communications.

Tâche	Moyenne et écart-type
Base	$10,586 \pm 4,285$
Calcul	$25,111 \pm 1,155$
Accès en écriture	$19,588 \pm 5,218$
Accès en lecture	$16,233 \pm 5,124$
Com. envoi	$18,315 \pm 4,295$
Com. réception	$16,014 \pm 4,236$

TAB. 6.4 – Consommation de chaque tâche du jeu de tests (Watts).

Margi *et al.* valident analytiquement certains des résultats. Par exemple, à partir des spécifications du processeur on peut calculer :
– la puissance consommée à pleine charge, soit $P_{\texttt{core}} = Vcc_{\texttt{core}} \times Icc_{\texttt{core}}$, numériquement on obtient $P_{\texttt{core}} = 1,65 \times 15 = 24,75\ W$;
– la puissance consommée en veille, soit $P_{\texttt{sleep}} = Vcc_{\texttt{sleep}} \times Icc_{\texttt{sleep}} = 12\ W$.

La puissance consommée à pleine charge correspond à peu près à la consommation mesurée pour la tâche de calcul, tandis que la puissance consommée en veille approxime dans une moindre mesure la consommation de la tâche de base. Une approche analogue permet de valider les résultats obtenus pour la consommation induite par les communications. En revanche aucune validation des résultats pour les accès disques n'est présentée. En effet, la difficulté vient des nombreux paramètres à prendre en compte tels que les temps d'accès, la taille du cache, etc. Pour ce qui est de l'affichage, les mesures effectuées en l'activant lors des tâches de base et de calcul montre un sur-coût énergétique non négligeable. Le tableau ci-dessous donne les valeurs mesurées.

Tâche	Moyenne	Écart-type
Base	14,516	6,129
Calcul	28,648	2,147

TAB. 6.5 – Impact de l'affichage sur la consommation énergétique (Watts).

Finalement, le tableau 6.6 donne la consommation énergétique de trois combinaisons de tâches élémentaires :

1. calcul (1,2 sec. - temps réel) + accès disque en écriture (2,8 sec. - 10 Mo de données) et en lecture (0,4 sec. - 10 Mo de données) ;
2. calcul (1,2 sec.) + envoi de données (2,3 sec. - 10 Mo de données) ;
3. calcul (1,2 sec.) + envoi de données (2,3 sec. - 10 Mo de données) + accès disque en lecture (0,4 sec. - 10 Mo de données).

Tâche	Moyenne	Écart-type
Combinaison 1	22,455	4,343
Combinaison 2	19,511	6,754
Combinaison 3	17,486	4,392

TAB. 6.6 – Consommation énergétique de combinaisons de tâches (Watts).

Consommation d'une plateforme de type Crossbow Stargate

Les tâches considérées comme élémentaires consistent en :
- du calcul ;
- une opération d'entrée / sortie (mémoire flash, capture d'image, etc.) ;
- une communication sans fil (envoi ou réception).

Comme précédemment, des combinaisons de tâches élémentaires sont également considérées. D'autre part, le jeu de tests est exécuté en distinguant les différents modes de l'interface réseau (*sleep*, *idle*, *transmission* et *reception*) et de la webcam (*off*, *on*, *acquiring image*). La consommation énergétique des états stables du nœud capteur et des transitions entre états est obtenue par mesure du courant à l'aide d'un ampèremètre. Ces mesures sont comparées à celles fournies par le système de mesure de l'énergie consommée intégré directement dans le nœud.

① États et transitions entre états

Plutôt que de considérer l'état (ou mode) de composants élémentaires, les auteurs considèrent les différentes unités du nœud capteur (cf. sous-section 1.4.1) :

1. l'unité de traitement qui regroupe le processeur, les mémoires (flash et RAM), plus les composants associés ;
2. l'unité de capture qui regroupe la webcam et l'interface USB ;
3. l'unité de communication qui comprend la carte d'accès sans fil et les modules PCMCIA associés.

Ces unités peuvent être dans différents états (*idle*, *sleep*, etc.). Naturellement, les auteurs n'explorent pas de manière exhaustive l'espace d'état, ils se focalisent sur quatre combinaisons d'états qui correspondent à des configurations pertinentes :

1. unité active : traitement - unités inactives : capture et communication ;
2. unités actives : traitement et capture - unité inactive : communication ;
3. unités actives : traitement et communication - unité inactive : capture ;
4. unités actives : traitement, capture et communication - unité inactive : aucune.

En pratique, pour contrôler l'état de composants élémentaires, tel que le processeur ou la carte d'accès réseau, des utilitaires de niveau système sont utilisés. Par exemple, `cardctl suspend` permet de mettre la carte WiFi en mode *sleep*, alors que `cardctl resume` la fait passer en mode *idle*.

Les transitions entre états consistent à ajouter ou à enlever des modules au noyau, ces modules contrôlant les sous-systèmes du nœud à activer ou désactiver (webcam, carte d'accès réseau, etc.). Cela peut parfois se traduire par la coupure de l'alimentation électrique de certains composants électroniques. Un changement d'état a un coût énergétique et requiert un certain délai pour être effectif. Les transitions / changements d'états considérés par Margi *et al.* sont :

1. activation / désactivation de la webcam ;
2. activation / désactivation de la carte WiFi ;
3. mise en veille du nœud ;
4. activation du nœud.

② Jeu de tests de consommation énergétique

Le jeu de tests est constitué cette fois de 5 types de tâches élémentaires. On retrouve les quatre types de tâches considérés dans le cas de l'ordinateur portable, plus une tâche de capture d'image.

– **Tâche de base**

Mesure la consommation énergétique induite par l'absence d'activité, seules des tâches systèmes basiques sont exécutées. Cet état sert de référence pour les autres tests.

– **Calcul intensif**

Comme pour l'ordinateur portable, ce test utilise le benchmark FFT.

– **Accès mémoire intensif**
Lecture / écriture de données aléatoires en mémoire flash.

– **Communication intensive**
Des programmes client/serveur générant du trafic réseau UDP sont utilisés. Les données transmises consistent en des nombres aléatoires.

– **Tâche de capture d'image**
Utilise le programme d'acquisition d'images `videotime` fourni par l'OS.

③ Étude de cas - Crossbow Stargate

– **Caractéristiques**
  – Processeur XScale PXA255 à 400 MHz ;
  – mémoire flash : 32 Mo ;
  – mémoire vive : 64 Mo de SDRAM ;
  – carte d'extension fournissant divers connecteurs : Ethernet, USB et série ;
  – carte réseau sans fil 802.11b ORiNOCO Gold ;
  – webcam : Logitech QuickCam Pro 4000 connexion USB (résolution de 640×480 pixels) ;
  – batterie Li-Ion : tension = 7,4 VDC, capacité de 1000 mAh ;
  – système d'exploitation : Stargate 7.3, linux pour système embarqué - kernel 2.4.19 - occupe moins de 10 Mo en mémoire flash.

– **Résultats des tests et mesure du coût et du délai des transitions**
Le tableau ci-dessous donne le coût énergétique, mesuré en milli-Ampères, lors de l'exécution du jeu de tests pour chacune des quatre combinaisons d'états. Ce sont des valeurs moyennes avec leur écart-type respectif obtenues à partir de 5 mesures réelles.

Tâche	Unité(s) active(s)			
	Traitement	Trait./Capture	Trait./Com.	Toutes
Base	$139 \pm 1,57$	$324 \pm 2,33$	$300 \pm 2,92$	$487 \pm 2,51$
Calcul	$291 \pm 1,21$	$474 \pm 9,70$	$457 \pm 1,23$	$642 \pm 9,20$
Accès en lecture	$248 \pm 2,12$	$434 \pm 5,53$	$414 \pm 2,89$	$604 \pm 1,94$
Accès en écriture	$248 \pm 1,33$	$436 \pm 6,90$	$413 \pm 1,05$	$604 \pm 2,16$
Capture d'image	-	$341 \pm 24,17$	-	$515 \pm 2,04$
Com. envoi	-	-	$383 \pm 0,92$	$559 \pm 8,39$
Com. réception	-	-	$361 \pm 2,72$	$537 \pm 9,10$
Veille	$3 \pm 0,04$	$3 \pm 0,02$	$10 \pm 0,26$	$9 \pm 0,11$

Tab. 6.7 – Consommation énergétique en milli-Ampères.

Plusieurs éléments d'information sont mis en évidence par le tableau 6.7 précédent. Tout d'abord, les communications sont moins coûteuses que le calcul et les accès mémoire, de plus, le sur-coût énergétique d'une émission par rapport à une réception est relativement faible (de l'ordre de 5 %). Toutefois, comme le font remarquer les auteurs, cette différence est minorée du fait que l'on ne

considère pas uniquement la carte d'accès réseau. En fait, si on ne se focalise que sur la consommation des communications une émission est bien plus coûteuse qu'une réception. Plus récemment, les auteurs ont donnés un tableau équivalent avec la consommation directement en Watts (voir le tableau 6.8). Globalement, l'activation de l'unité de capture se traduit par une consommation supplémentaire de 185 mA par rapport à l'unité de traitement seule. L'activation de l'unité de communication en plus de celle de traitement induit elle une consommation supplémentaire de 165 mA. La consommation résultant de l'activation d'une unité est identique pour toutes les tâches (mis à part la veille prolongée).

	Unité			Énergie
	Traitement	Capture	Communication	consommée
État	Veille	Veille	Veille	0,34
	Idle	Veille	Veille	0,67
	Active	Veille	Veille	1,9
	Idle	Veille	Idle	1,51
	Active	Veille	Idle	2,8
	Idle	Veille	Active	2,95
	Idle	Veille	Active	2,73
	Idle	Idle	Idle	2,38
	Active	Idle	Idle	3,68
	Idle	Idle	Réception	3,5
	Idle	Idle	Envoi	3,7
	Idle	Idle	Veille	1,53
	Active	Idle	Veille	2,8

TAB. 6.8 – Consommation énergétique en Watts.

Comme le montre le tableau 6.9, le coût énergétique des changements d'état du nœud capteur n'est pas négligeable. D'autre part, le délai nécessaire pour que la transition soit réalisée ne doit pas être oublié. Cependant, il n'y a pas de corrélation entre consommation énergétique et délai.

	Unité(s) active(s)			
Tâche	Traitement	Trait./Capture	Trait./Com.	Toutes
Active webcam	-	122 ± 4	-	126 ± 5
Désactive webcam	-	74 ± 14	-	69 ± 4
Active WiFi	-	-	92 ± 2	95 ± 3
Désactive WiFi	-	-	47 ± 4	47 ± 2
Nœud en veille	65 ± 2	119 ± 9	127 ± 24	176 ± 19
Active nœud	87 ± 14	269 ± 6	504 ± 91	835 ± 16

TAB. 6.9 – Consommation en milli-Coulombs lors des changements d'état.

6.3 Description des simulations

Afin d'évaluer expérimentalement les performances de l'algorithme 5.1, nous avons développé une simulation générique en utilisant le simulateur à évènements discrets OMNeT++ [125, 106]. La simulation est générique dans le sens où d'une part la taille et la topologie du réseau considéré peuvent être changés facilement, d'autre part le type de chaque nœud est choisi aléatoirement (réseau de capteurs hétérogènes). Dans nos expérimentations nous considérons trois topologies réseau :

- complètement connecté ;
- linéaire ;
- mixte.

Le réseau mixte, le plus réaliste, a été engendré aléatoirement grâce au générateur de topologie réseau BRITE [126]. À noter que ce générateur de réseau n'est pas spécifique à OMNeT++. Les deux premières topologies ont été étudiées pour des tailles de réseau de 5 et 10 nœuds, celle du réseau mixte étant de 20 nœuds.

Une caractéristique fort intéressante de notre algorithme est sa capacité à gérer des réseaux hétérogènes. C'est pourquoi, dans notre scénario de simulation (cf. section 6.1) nous avons considéré deux types de nœuds capteurs. Nous pensons également que les différentes consommations énergétiques reflète assez fidèlement la réalité. En effet, la consommation liée à l'exécution de tâches, inhérente à chaque type de nœud s'appuie sur des mesures réelles [17, 116], et c'est aussi le cas du modèle de consommation énergétique choisie puisqu'il s'agit de celui de Feeney et Nilsson [121, 122]. Chaque nœud d'un réseau simulé peut donc être soit un nœud ayant des capacité limitées, du type Crossbow Stargate, soit un nœud plus puissant en capacité de traitement et disposant d'une grande provision d'énergie, type ordinateur portable. Le type d'un nœud est choisi aléatoirement à l'initialisation du réseau simulé. Enfin, on peut remarquer que notre approche est apte à gérer une topologie dynamique, que l'évolution de la topologie résulte de la mobilité ou de la perte de liens suite, par exemple, à la défaillance d'un nœud.

Les différentes simulations ont été réalisées en paramétrant la consommation énergétique associée à l'exécution de tâches et la quantité d'énergie disponible initialement, pour chaque type de nœud, suivant les valeurs données dans le tableau 6.10.

Nœud		Crossbow Stargate		Portable Dell	
Batterie	Capacité / charge max.	7,4 Wh / 50%		56,25 Wh / 50%	
Classe de tâche		$e_{i,j}$	$N_{i,j}$	$e_{i,j}$	$N_{i,j}$
	1	5 W	1	28,65 W	1
j	2	4,75 W	1	24,75 W	1
	3	4,25 W	1	21 W	1
Tâche de base (nœud idle)		1.53 W		10.59 W	
Éxec. algorithme (coût fixe)		1.74 W		11 W	

TAB. 6.10 – Paramétrage énergétique de chaque type de nœud.

Ces valeurs sont basées sur celles des tableaux 6.4, 6.5, 6.6 et 6.8.

La charge initiale de chaque nœud est définie comme suit : les nœuds de type Crossbow Stargate auront en charge 250 tâches, l'autre type de nœud en aura lui le double, soit 500 tâches. Les tâches sont réparties aléatoirement dans trois classes, chaque classe ayant un coût d'exécution unitaire différent (voir le tableau 6.10). Le coût d'exécution de l'algorithme a lui été fixé arbitrairement. Au départ, tout nœud exécute une tâche de chaque classe durant un pas de temps ($\forall j, N_{i,j} = 1$), mais dès qu'une classe n'a plus de tâches, une tâche supplémentaire d'une autre classe sera exécutée. Puis, lorsque plus aucune tâche n'est à exécuter, c'est la tâche de base qui prend le relais, cela jusqu'à épuisement total de l'énergie. En fait, la charge d'un nœud décroît nécessairement dans le temps, car nous n'avons intégré la génération de tâches dans nos simulations que durant un nombre limité de pas de temps (étape 17 de l'algorithme 5.1). Au niveau de la quantité de données à transmettre lors de la migration d'une tâche, elle est de l'ordre de 460 Ko pour une tâche de la première classe et de 46 Ko pour les deux autres.

La capacité initiale de la batterie d'un nœud est calculée à partir de sa charge initiale. En effet, on commence par tirer aléatoirement, suivant une distribution uniforme, le ratio initial du nœud, ensuite on calcule l'énergie dont il disposera connaissant sa charge à l'instant $k = 0$. Deux distributions sont considérées. La première a pour domaine de définition $[0.25, 1)$, alors que celui de la seconde est $[1, 2)$. L'idée sous-jacente est d'avoir certains nœuds dont le ratio est supérieur à 1, c'est-à-dire qui vont défaillir avant d'avoir exécuter toutes leurs tâches.

6.4 Performances de l'algorithme

Les différents réseaux ont été évalués en considérant 20 configurations initiales aléatoires différentes (choix aléatoire du type, de la répartition de la charge et du ratio, tout deux à l'instant $k = 0$, pour chaque nœud).

6.4.1 Convergence des ratios dans le réseau

La première question qui se pose concerne la convergence des ratios. De fait, la validation expérimentale préliminaire à laquelle nous avons procédé, présentée dans la sous-section 5.3.2, montre que les ratios convergent, mais en l'absence de la consommation d'énergie. La question est donc de savoir quid de la consommation énergétique ? Les figures 6.3, 6.4 et 6.5 sont représentatives de l'évolution des ratios pour chacune des topologies considérées. Comme on peut le voir, les ratios convergent toujours. La vitesse de convergence est évidemment liée à la topologie et à la taille du réseau. On observe que la convergence la plus rapide est obtenue avec le réseau complètement connecté, requérant en moyenne 15 itérations pour converger. Dans le cas de la topologie linéaire elle est de l'ordre de 26 itérations et bien plus pour un réseau « mixte ». On peut aussi remarquer la robustesse de notre algorithme vis à vis de la perte d'un nœud, puisque dans la figure 6.5 il apparaît qu'un nœud

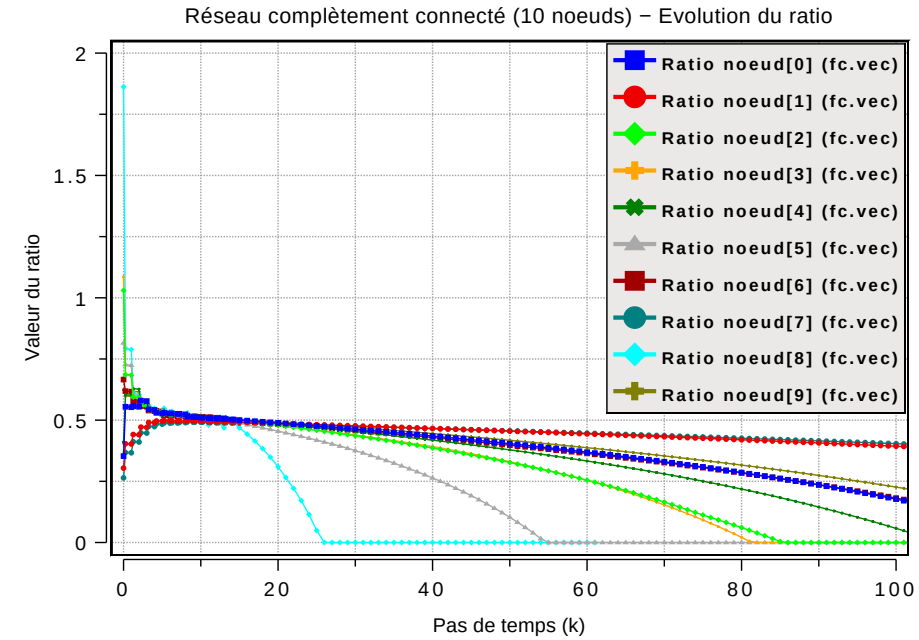


FIG. 6.3 – Courbe d'évolution des ratios dans le cas du réseau complètement connecté (10 nœuds).

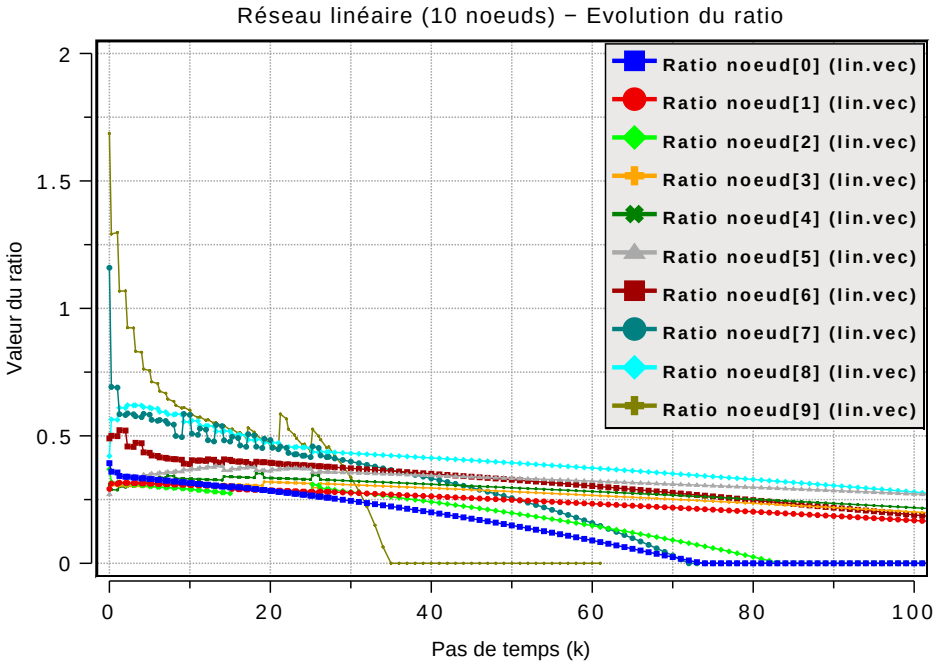


FIG. 6.4 – Courbe d'évolution des ratios dans le cas du réseau linéaire (10 nœuds).

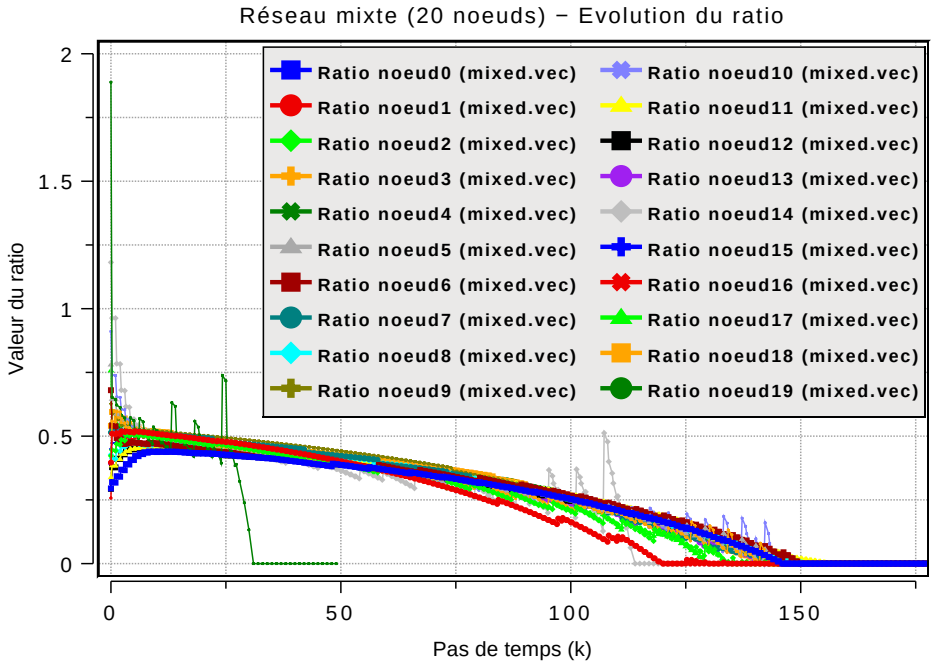


FIG. 6.5 – Courbe d'évolution des ratios dans le cas du réseau mixte (20 nœuds).

« meurt » avant le pas d'itération 50 (arrêt de la courbe), alors que l'algorithme est encore en cours d'exécution.

Un autre élément intéressant est le fait que les courbes d'évolution de ratio se rapprochent de zéro et atteignent systématiquement cette valeur. Cela signifie que grâce à notre algorithme aucun nœud ne « meurt » avant d'avoir exécuter toutes les tâches des trois classes. Bien entendu, il faut prendre en compte l'arrêt de la génération de tâches au bout de 10 itérations. On constate donc que l'algorithme que nous proposons permet de garantir un certain niveau de Qualité de Service, car il garantit que toutes les tâches seront exécutées.

Enfin, on constate des oscillations au niveau de certaines courbes, plus particulièrement dans la figure 6.5. Ces oscillations de la valeur du ratio apparaissent quand le nœud correspondant approche de sa fin de vie. L'explication est que lorsqu'un nœud n'a plus beaucoup d'énergie c'est l'évolution de la charge du nœud qui va guider celle du ratio. En effet, d'un côté l'exécution de tâches va faire fortement baisser son ratio, de l'autre côté si son ratio est inférieur à certains de ses voisins, ceux-ci vont lui envoyer des tâches qui vont faire croître fortement son ratio. Pour limiter ce phénomène qui aboutit à des migrations de tâches inutiles et donc précipitant la défaillance d'un nœud, nous avons introduit un niveau d'énergie en dessous duquel tout nœud arrête d'exécuter l'algorithme, sauf si son ratio est supérieur à 1.

Les figures 6.6(a) et 6.6(b) présentent les quantités de données reçues et envoyées par chaque nœud, respectivement pour les réseaux complètement connecté et linéaire (10 nœuds) des figures 6.3 et 6.4. Ces valeurs nous semblent réalistes pour un réseau de capteurs multimédia. Dans le cas du réseau complètement connecté on voit que le nœud 8 ne fait presque qu'envoyer des données, car c'est lui qui a la valeur de ratio la plus élevée au départ (voir la figure 6.3). Alors que le nœud 7 qui a le ratio initial le plus faible ne fait quasiment que recevoir des données. Le réseau linéaire se caractérise, comme attendu, par une croissance du trafic réseau des nœuds externes vers les nœuds les plus internes.

6.4.2 Gain au niveau de la durée de vie du réseau

Le tableau 6.11 quantifie précisément l'amélioration moyenne de la durée de vie pour les différentes configurations de réseau. Comme on pouvait s'y attendre, les meilleures performances sont obtenues avec un réseau complètement connecté, quelle que soit le nombre de nœuds, avec un gain moyen de plus de 50 %. La vitesse de convergence, très rapide pour cette topologie, y est pour beaucoup. La topologie linéaire est très sensible à la taille du réseau, cela provient de l'augmentation de la distance moyenne entre les nœuds. Le réseau mixte présente les moins bonnes performances, notamment du fait du nombre de nœuds, néanmoins le gain reste tout à fait satisfaisant.

Si on regarde les performances suivant les tailles de réseau pour les topologies linéaires et mixtes, on voit une dégradation quasi linéaire. Cela suggère l'inadéquation de notre algorithme au passage à l'échelle, du moins dans sa forme actuelle. Pour surmonter ce problème, nous envisageons dans le futur d'explorer la piste d'une version hiérarchique de notre algorithme. L'idée serait de s'appuyer sur l'agrégation de nœuds en clusters, i.e. utiliser l'algorithme actuel pour équilibrer les ratios au sein de chaque clusters, puis définir un méta-algorithme qui serait chargé de gérer la migration inter-cluster (cet algorithme serait uniquement exécuté par un chef de cluster). Cette approche hiérarchique nécessiterait l'exécution préalable d'un algorithme de formation de clusters.

La figure 6.7 compare la durée de vie sans l'algorithme à celle avec, pour chaque nœud du réseau linéaire de la figure 6.4. Ce graphique à barres verticales montre que 4 noeuds ont une durée de vie plus longue, en particulier les nœuds 7 et 9 qui sont les deux premiers à épuiser leur énergie. Le prolongement de la vie de certains nœuds fait logiquement décroître celle d'autres nœuds. Finalement, la figure 6.8 montre que sans l'algorithme le nœud 9 serait « mort » avant l'itération 50, sans avoir exécuté toutes ses tâches puisque le ratio tend vers l'infini.

Topologie	Complètement connectée		Linéaire		Mixte
Taille	5	10	5	10	20
$\epsilon = 10^{-2}$	53,50%	57,80%	46,40%	23,50%	12,75%

TAB. 6.11 – Gain moyen de durée de vie pour les différents réseaux considérés.

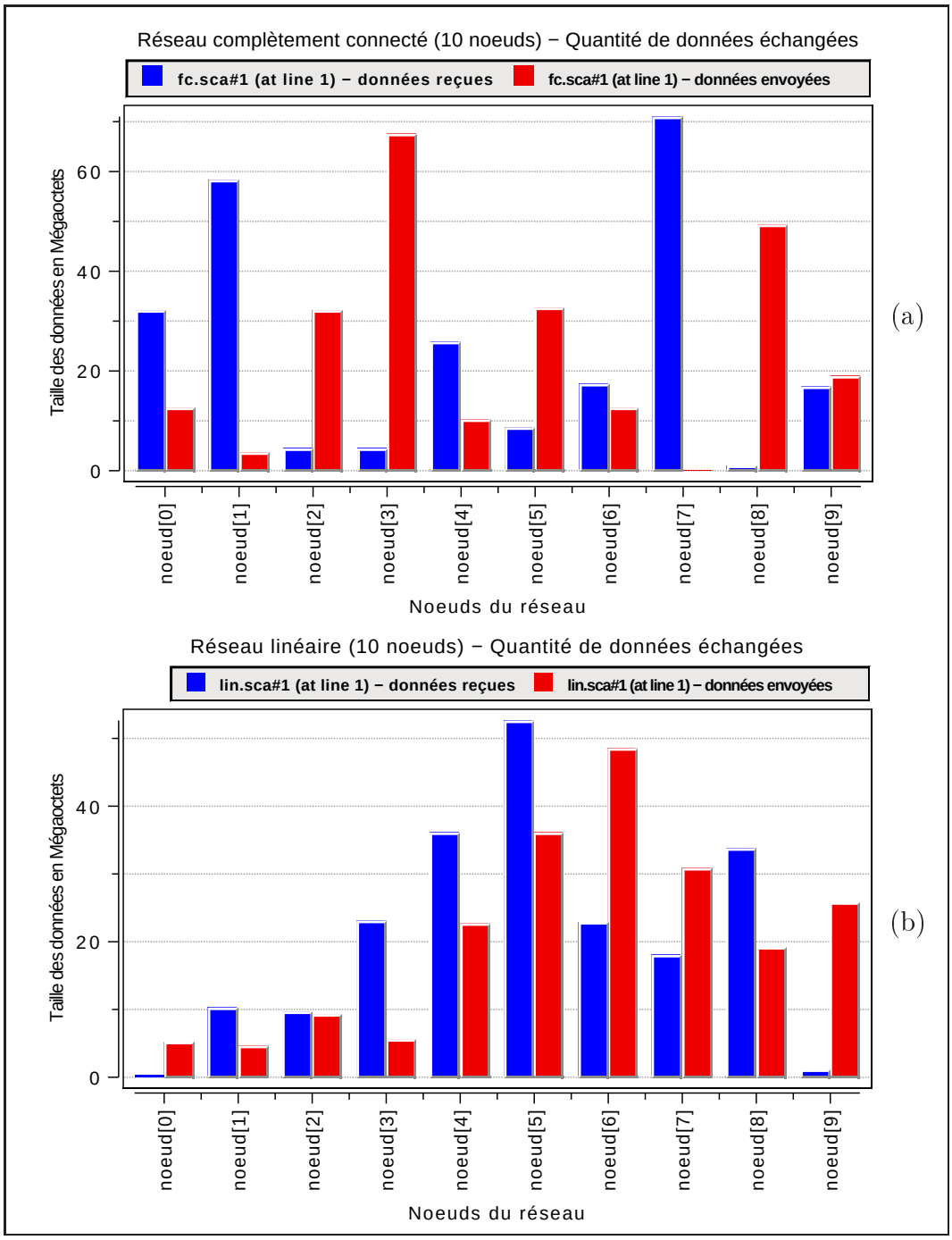


FIG. 6.6 – Quantité de données échangées par chaque nœud capteur, dans le cas du réseau complètement connecté (a), du réseau linéaire (b).

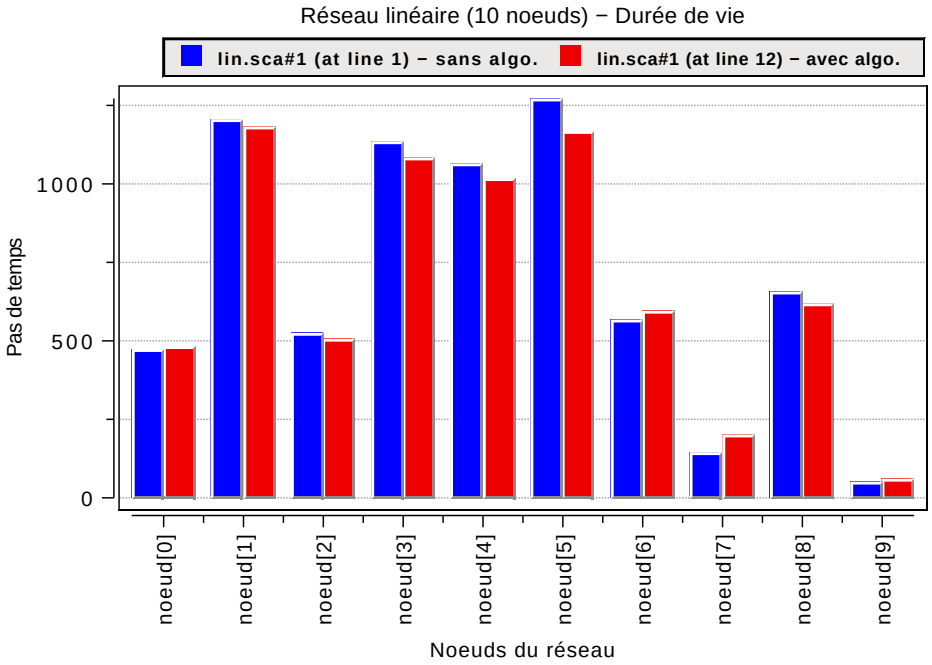


FIG. 6.7 – Comparaison de la durée de vie de chaque nœud du réseau linéaire (10 nœuds), sans et avec l'algorithme 5.1.

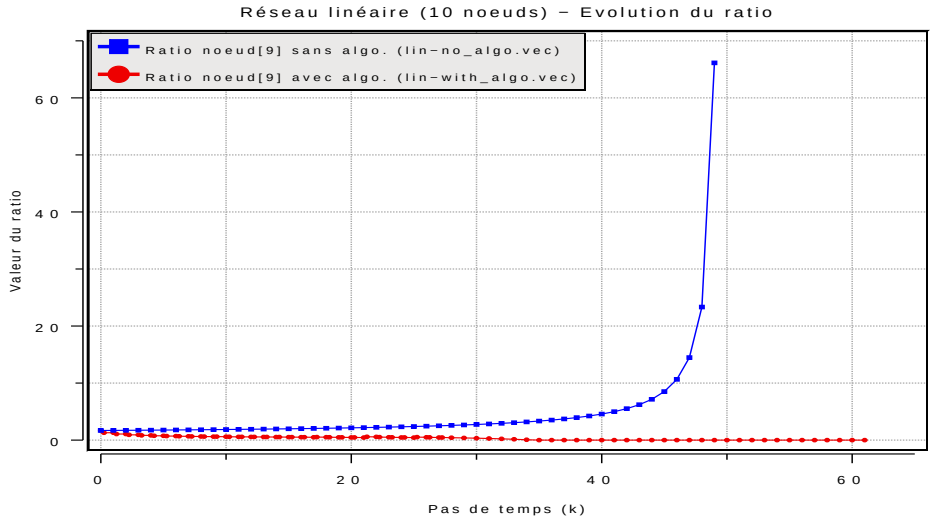


FIG. 6.8 – Courbe d'évolution du ratio du nœud 9 sans optimisation de la durée de vie *versus* avec optimisation.

Chapitre 7

Simulation de réseaux dynamiques

7.1 Introduction

La simulation de réseaux dynamiques est plus complexe, car contrairement au cadre statique où le nombre de nœuds du réseau est fixe et ceux-ci immobiles, avec un réseau dynamique il faut pouvoir gérer :

- la disparition de nœuds, ce qui était déjà toutefois le cas du simulateur développé pour les réseaux statiques ;
- l'apparition de nœuds à l'issue de phases de redéploiement ;
- la mobilité des nœuds.

Cela suppose en particulier de savoir gérer l'évolution dynamique du voisinage d'un nœud. La finalité du simulateur que nous décrivons ci-après, encore en cours de développement, est précisément de répondre à ces besoins. Comme nous le verrons, l'architecture générale du simulateur est en place, mais en l'état nous ne sommes pas encore en mesure de simuler des réseaux dynamiques. Toutefois, nous donnons quelques résultats obtenus dans le cadre statique.

7.2 Description du simulateur en développement

Ce simulateur, dénommé « AdhocNet-Sim » permet de simuler, d'une manière générale, la dynamique de nœuds / hôtes d'un réseau ad hoc. La dynamique est régie par l'exécution de tâches au niveau de chaque nœud et par le protocole de communication du réseau (gestion des émissions / envois et des réceptions). Bien entendu, on considère des nœuds hétérogènes, chacun ayant une dynamique qui lui est propre. Du point de vue programmation, le simulateur AdhocNet-Sim est implanté en C++ et exploite, tout comme le simulateur présenté pour le cadre statique, les routines d'envoi et de réception du simulateur OMNet++ [125, 106]. Comme le montre la figure 7.1, il comporte quatre composants :

1. gestionnaire d'un nœud ;
2. gestionnaire de tâches ;
3. gestionnaire des envois / réceptions ;

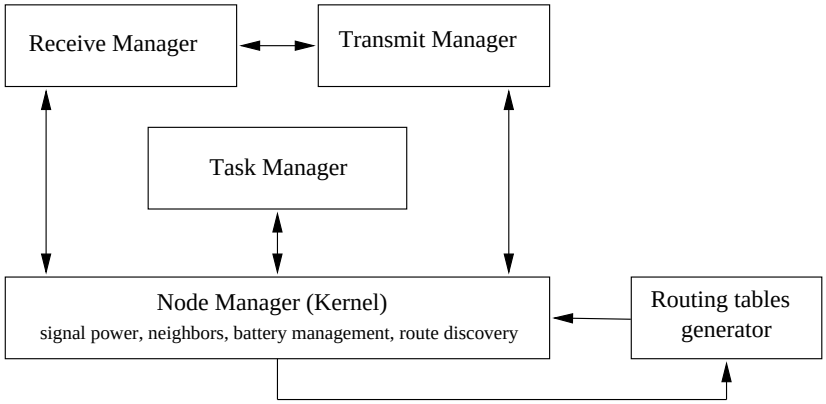


FIG. 7.1 – Les différents composants du simulateur AdhocSim-Net.

4. générateur des tables de routage : Le routage adopté est basé sur le Flooding / Gossiping. Plusieurs stratégies ont été implantés, le choix des noœuds destinations est choisi selon une loi uniforme et/ou Poisson, dont les paramètres sont justifiés dans la suite du chapitre.

La philosophie de conception de cet outil est basée sur un modèle probabiliste. Aussi, les envois de messages et l'exécution de tâches sont paramétrables selon des lois de probabilité. Cette approche nous permet de contrôler au mieux les aspects réseaux notamment le type de trafic, la charge ou encore la fréquence d'exécution des tâches localement au niveau des nœuds. En effet, pour certaines applications réseaux le trafic généré est poissonien, pour d'autres de type CBR (*Constant Bit Rate* - cas des flux audios et vidéos) [127]. Les différents paramètres nous permettent d'avoir une flexibilité sur les types de réseaux que l'on désire simuler, de contrôler les taux d'activités (forte ou faible), les taux d'occupation, etc. Dans la suite, seuls des réseaux modélisés par des graphes connectés seront considérés. Enfin il est important de souligner que les envois, réceptions et calculs sont non-bloquants.

Plus généralement, nous supposerons qu'un réseau sera :

– composé de N nœuds, avec

$$N = |\Omega| \text{ où } \Omega = \{1, 2, \ldots, N\}$$

dénote l'ensemble des nœuds du réseau ;

– ceux-ci étant hétérogènes, ils sont répartis en C classes

$$C = |\Psi|, \text{ tel que } \Psi = \{\Psi_1, \Psi_2, \ldots, \Psi_C\}$$

représente l'ensemble des classes de nœuds.

– Les nœuds seront déployés de façon ad hoc et d'une manière complètement aléatoire dans un espace à deux ou trois dimensions. De plus, il existera M classes de tâches

$$M = |T|, \text{ tel que } T = \{T_1, T_2, \ldots, T_M\}$$

est l'ensemble des classes de tâches à exécuter.

7.2.1 Aspects réseaux

Les réseaux considérés dans cette étude sont quelconques, on ne se focalise pas sur des topologies particulières. Les graphes de connexion sont générés de façon aléatoire, en utilisant une loi de distribution sur les arêtes (uniforme, géométrique, etc.), et restent fixes tout au long de la durée de vie du réseau. Les communications sans fil entre nœuds du réseau sont simulées par des canaux uni ou bidirectionnels (ou inclusif), comme on peut le voir sur la figure 7.2. Ainsi, la matrice d'incidence associée au graphe de connexion du réseau peut être symétrique ou non. Il est à noter que la perte, l'ajout et la mobilité des nœuds ne sont pas pris en compte pour le moment.

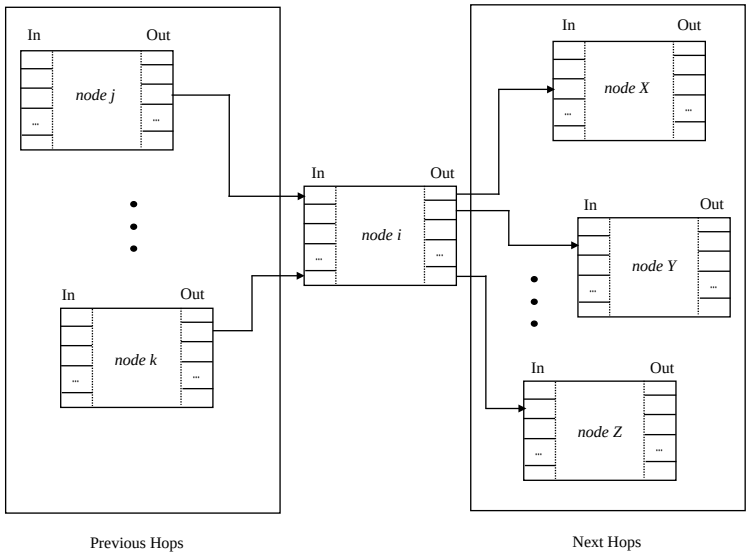


FIG. 7.2 – Connexion entre nœuds d'un réseau.

Justification de la modélisation des connexions choisie

La modélisation des communications entre nœuds adoptée dans cette étude n'est pas un modèle arbitraire. En effet, il est issue de la littérature des modèles de couche MAC du modèle OSI existants dans le contexte des réseaux sans fil. Le modèle présenté dans la figure 7.2 correspond parfaitement au mode CDMA (*Code Division Multiple Access)*. En l'occurrence, dans le mode CDMA chaque nœud i envoie à un nœud j sur la fréquence $f_{i,j}$, telle que $f_{i,j} = f_{j,i}$ (voir figure 7.3). Dans son rayon de couverture r_i, le nœud i dispose des fréquences de réception associées à ses nœuds voisins. Dans la suite de ce document, on ne parlera plus des fréquences d'envoi ou de réception, elles seront modélisées par des ports de communication entrants et sortants.

Voisinage d'un nœud

Chaque nœud dispose d'une interface de communication sans fil limitée en puissance d'émission, celle-ci définit implicitement la zone de couverture associée au

nœud. On rappelle que la zone de couverture d'un nœud i, notée r_i, correspond à la surface dans laquelle tout nœud présent pourra communiquer directement avec le nœud i. À partir du rayon de couverture de chaque nœud d'un réseau on peut définir toutes les topologies réseau possibles, i.e. l'organisation des communications entre les nœuds du réseau.

7.2.2 Définition des différentes classes considérées

Classes de nœuds

Comme les nœuds sont hétérogènes au niveau de leurs caractéristiques (capacité de calcul et de stockage, interface de communication, etc.), ils sont répartis suivant ces dernières en C classes. Chaque classe $\Psi_k, k \in \{1, \ldots, C\}$, est définie par un ensemble de paramètres bien précis :

- Identificateur de classe : $k,\ k \in \{1, \ldots, C\}$;
- Description : *Palm*, *PDA*, *ordinateur portable*, etc. ;
- Capacité énergétique maximale : E^0 ;
- Paramètres de communications : $\beta_k, \mu_{1_k}, \mu_{2_k}$;
- Paramètres de calculs : σ_k.

Classes de tâches

Les tâches exécutées dans le réseau sont divisées en deux classes principales : tâches système et tâches applicatives. Chaque nœud du réseau dispose de tâches hétérogènes à exécuter tout au long de sa durée de vie. Pour être plus réaliste, on adoptera les conventions suivantes :

- un nœud i ne peut exécuter d'applications sans avoir de tâches système ;
- les tâches applicatives traitées par un nœud dépendent des capacités en calcul et en communication de ce nœud.

Une classe de tâches $T_j, j \in \{1, \ldots, M\}$, est définie comme suit :

- Identificateur de classe : j ;
- Description : tâche système ou applicative (traitement d'image, de vidéo, etc.) ;
- T_{exec_j} : temps (en secondes) nécessaire pour exécuter une tâche de cette classe ;

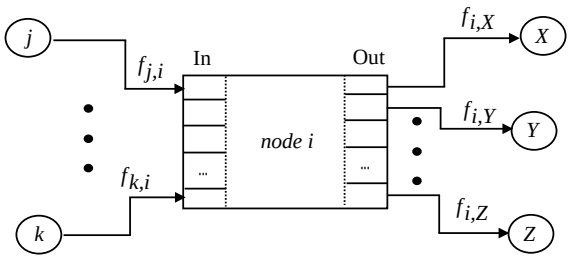


FIG. 7.3 – Modèle de communication en mode CDMA.

- Taille d'une tâche : occupation mémoire en octets de cette tâche durant son cycle de vie ;
- Nombre total de tâches de cette classe disponibles dans le nœud au démarrage de la simulation, soit à l'instant $t = 0$;
- γ_j : paramètre coût énergétique (voir la sous-section consacrée aux consommations énergétiques) ;
- Min_j : nombre minimal de tâches de cette classe à exécuter durant un pas de temps donné ;
- Max_j : nombre maximal de tâches de cette classe à exécuter durant pas de temps donné ;
- θ_j : paramètre pour exécuter N_j^t tâches durant le pas de temps t, N_j^t vérifiant $Min_j \leq N_j^t \leq Max_j$;
- pour simuler la migration de tâches entre nœuds, on fait correspondre à chaque tâche, une taille en mode statique (taille en octets du programme à déporter : calcul du factoriel ou de compression vidéo, etc.).

Classes de messages

Le protocole de communication modélisé fait appel à cinq classes de messages qui sont détaillés dans les tables 7.1 et 7.2. Chaque type de message a un rôle bien défini :

- **messages de type 1** : Auto-Organisation (*Self-organization*) ;
- **messages de type 2** : Requêtes de demande de Routage (*Request to route*) ;
- **messages de type 3** : Paquets de données (*Packets*) ;
- **messages de type 4** : ACK ;
- **messages de type 5** : NACK ($NACK = \overline{ACK}$).

7.2.3 Consommations énergétiques

Énergie consommée par les communications

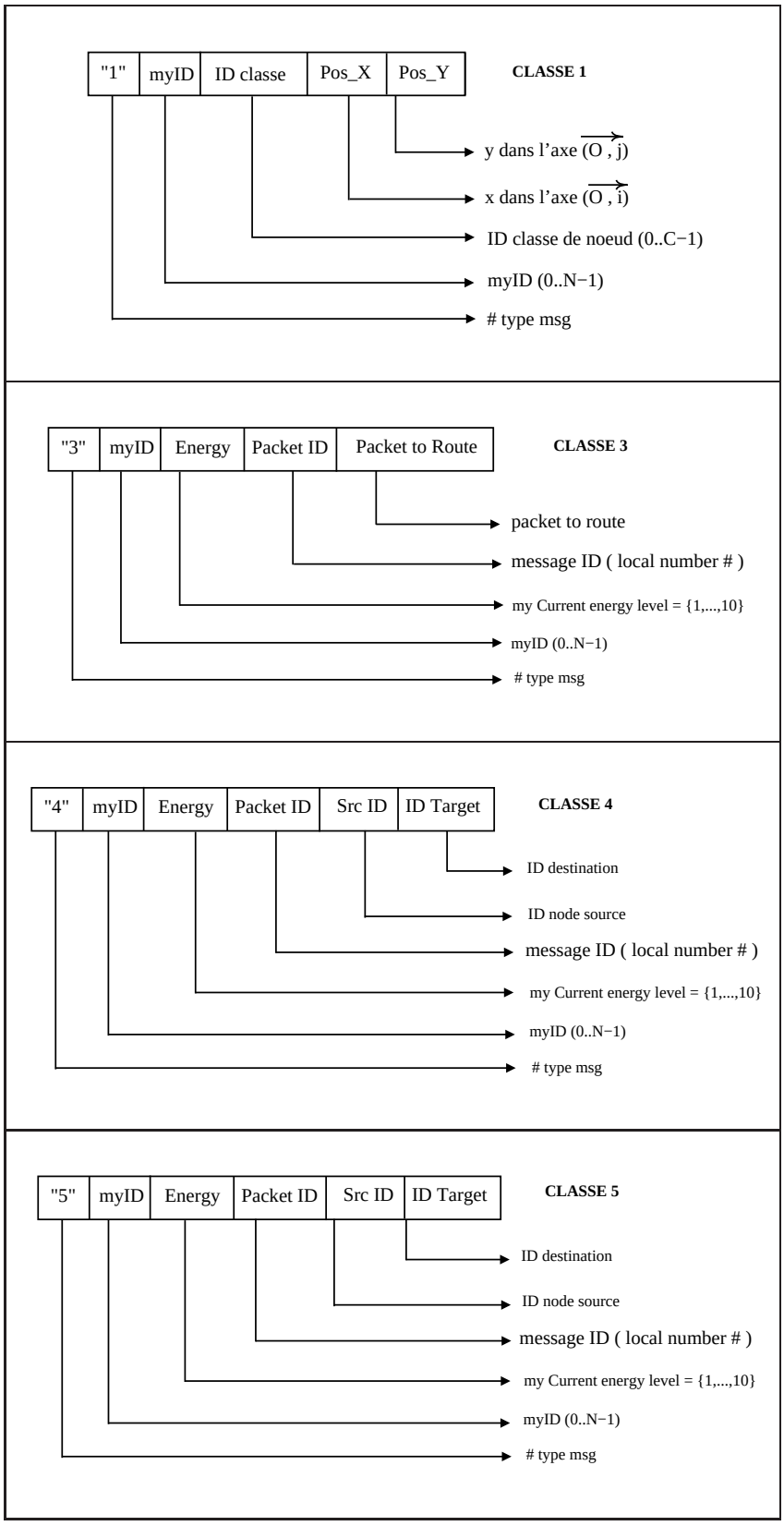
Lors de la présentation du simulateur développé dans le cadre de réseaux statiques, nous avons décrit l'approche considérée par Cano et Manzoni pour formalisée la quantité d'énergie consommée par les interfaces de communication sans fil de type WiFi (IEEE 802.11). L'énergie consommée lors d'une communication induisant l'échange d'un paquet de données est donc définie, suivant qu'il s'agit d'un émission ou d'une réception, par l'une des deux équations ci-dessous.

- **Énergie consommée à l'émission**

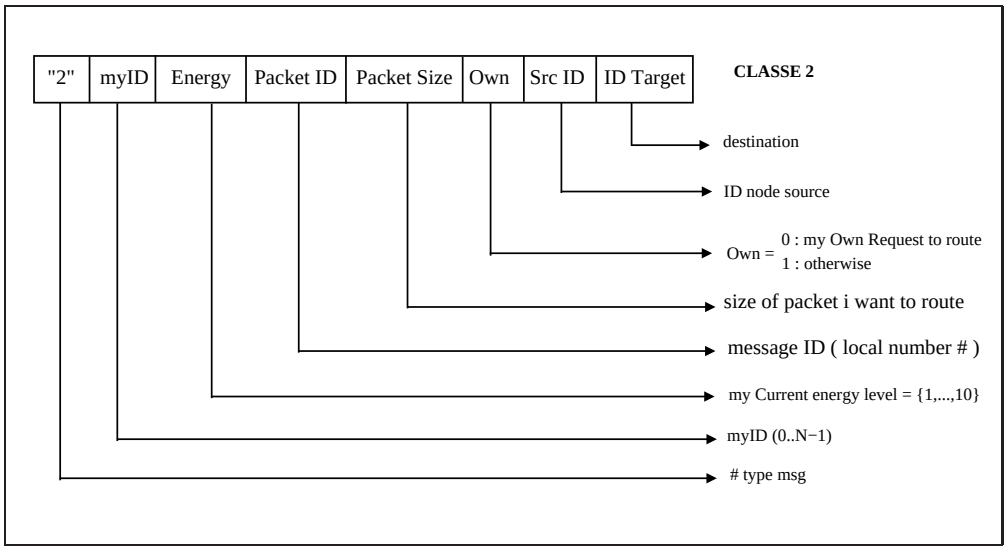
$$E_{\texttt{TX}}(<\texttt{Paquet}>) = 5 \times 330 \times \frac{\texttt{<Taille du paquet en octets>} \times 8}{2 \times 10^6} \quad (7.1)$$

- **Énergie consommée à la réception**

$$E_{\texttt{RX}}(<\texttt{Paquet}>) = 5 \times 230 \times \frac{\texttt{<Taille du paquet en octets>} \times 8}{2 \times 10^6} \quad (7.2)$$



TAB. 7.1 – Messages de types 1, 3, 4 et 5.



TAB. 7.2 – Message de type 2.

Énergie consommée lors de l'exécution d'une tâche

L'énergie consommée par l'exécution d'une tâche T_j de classe j est plus complexe à modéliser. Intuitivement, nous avons choisi de définir E_{T_j} (l'énergie nécessaire à l'exécution d'une tâche T_j) en fonction de $E_{\texttt{TX}}$ et du facteur γ_j :

$$E_{T_j}(\gamma_j) = E_{\texttt{TX}}(< \texttt{Taille } T_j \texttt{ en bits} >) \times \gamma_j \tag{7.3}$$

où γ_j désigne un facteur de coût énergétique associé à la classe de tâches T_j et la taille de T_j correspond au total de son occupation mémoire durant tout le cycle de vie de la tâche.

7.2.4 Niveau d'énergie d'un nœud

Le niveau d'énergie d'un nœud est défini par un entier compris entre 0 et 10, de manière à alléger les paquets de données qui contiennent l'information sur l'énergie. L'énergie dont dispose un nœud i à l'instant t est un entier sur 64 bits, par contre, le niveau d'énergie est seulement sur 4 bits. Formellement, à l'instant t, le niveau d'énergie $L(E_i^t)$ du nœud i est donné par :

$$L\left(E_i^t\right) = \left\lceil \frac{10 \times E_i^t}{E^0} \right\rceil \tag{7.4}$$

7.2.5 Fonctionnement du simulateur

Phase d'initialisation

La phase d'initialisation d'un réseau se compose de quatre parties :

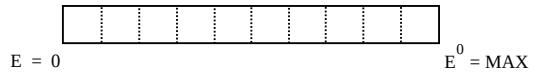


FIG. 7.4 – Jauge de batterie

1. génération de la topologie du réseau ;
2. initialisation des nœuds du réseau ;
3. diffusion d'initialisation ;
4. génération des tables de routage (cette partie sera décrite ultérieurement).

- **Génération de la topologie du réseau**

  La topologie du réseau est générée à partie de l'interface offerte par le simulateur OMNeT++. Cela nous permet de définir les connexions entre nœuds du réseau de façon déterministe ou stochastique comme suit :

```
// Exemple 1 :
  connections :
    n[0].out++ -->  delay 150ms --> n[1].in++;
    n[0].in++  <--  delay 150ms <-- n[1].out++;

    n[0].out++ -->  delay 280ms --> n[2].in++;
    n[0].in++  <--  delay 200ms <-- n[2].out++;

    n[0].out++ -->  delay 100ms --> n[3].in++;
    n[0].in++  <--  delay 100ms <-- n[3].out++;
               ...
    n[23].out++ -->  delay 250ms --> n[28].in++;
    n[23].in++  <--  delay 150ms <-- n[28].out++;

    n[23].out++ -->  delay 350ms --> n[12].in++;
    n[23].in++  <--  delay 150ms <-- n[12].out++;

    ...
// Exemple 2 :
  connections :
    for i=0..size-1 do
      n[i].out++ --> delay 100ms --> n[intuniform(0,size-1)size].in++
       if intuniform(0,size-1) != i;
      n[i].in++ <-- delay 100ms <-- n[intuniform(0,size-1)size].out++
       if uniform(0,1) < 0.3;
      n[i].out++ --> delay 100ms --> n[intuniform(0,size-5)size].in++
       if intuniform(0,size-1) < i;
    endfor;
```

- **Initialisation des nœuds du réseau**

  Cette phase est dédiée à l'affectation des paramètres de « dynamicité »aux différents nœuds qui constituent le réseau : initialisation de la classe d'appartenance (i.e. la classe de nœuds à laquelle il appartient), des classes de

tâches, des paramètres d'exécution, etc. La figure 7.5 décrit plus précisément les différents éléments qui sont définis lors de l'initialisation d'un nœud i.

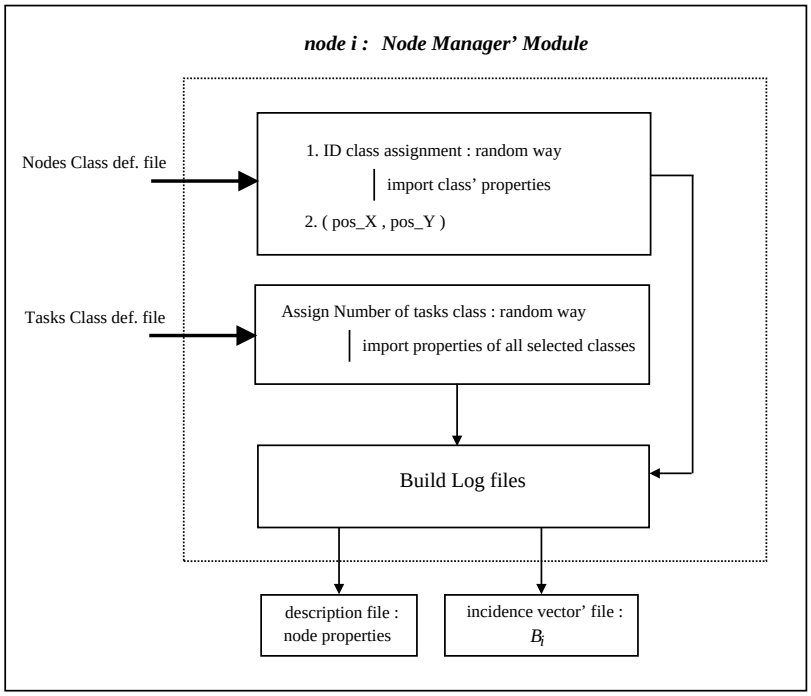


FIG. 7.5 – Initialisation d'un nœud i.

- **Diffusion d'initialisation**

  Chaque nœud i diffuse un message de type 1 dans son voisinage V_i (voir la figure 7.6. Le message contient la position géographique du nœud, ainsi que sa classe d'appartenance ce qui sera utile pour les stratégies de migration de tâches (migrer une tâche selon les capacités du nœud destinataire et donc de sa classe d'appartenance).

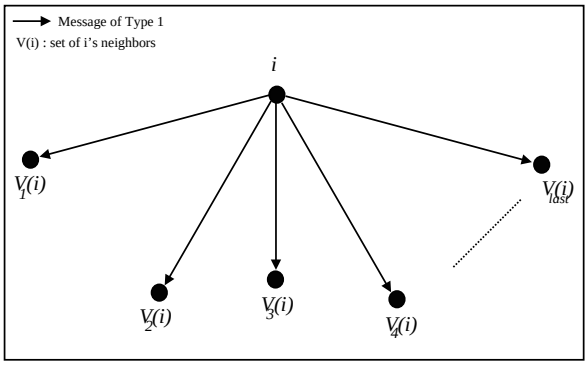


FIG. 7.6 – Diffusion d'initialisation

Gestionnaire d'un nœud

Le gestionnaire d'un nœud offre une multitude de fonctionnalités. Il s'agit principalement de gérer les différents paramètres du nœud :

- activation du gestionnaire des émissions et des réceptions ;
- sélection des classes du nœud et du nombre de tâches à exécuter durant un pas de temps t ;
- l'accès à la table de routage ;
- sélection des nœuds pour les requêtes de demande de routage ;
- acceptation ou refus de routage ;
- contrôle du niveau d'énergie ;
- etc.

Gestionnaire des tâches

- À l'instant t, chaque nœud i dispose de M_i^t classes de tâches telles que :

$$M_i^t = |T_i^t| \quad \text{avec} \quad T_i^t \in \mathcal{J} \quad \text{et} \quad M_i^t \leq M. \tag{7.5}$$

  où

  $\mathcal{J} = \{\{T_1\}, \{T_1, T_2\}, \ldots, \{T_1, T_2, \ldots, T_M\}\}$ est l'ensemble fini des partitions de T ;

  $T = \{T_1, \ldots, T_M\}$ est l'ensemble des classes de tâches ;

  T_1 désigne la classe des tâches système (imposées) ;

  et T_i^t désigne l'ensemble des classes de tâches dont dispose le nœud i à l'instant t.

- Le nombre total de tâches dont dispose le nœud i à l'instant t est donc donné par :

$$\sum_{k=1}^{M_i^t} |u_k^t|, \tag{7.6}$$

  où

  $u_k^t \in T_i^t$ est l'ensemble des tâches des classes d'indices k dans T_i^t ;

  et $|u_k^t|$ représente le nombre de tâches de cette classe disponible à l'instant t.

- Pour un nœud i quelconque, le nombre de tâches N_j^t de la classe j à exécuter durant le pas de temps t est :

$$N_j^t = \boldsymbol{U}^{\mathbb{N}^+}(Min_j, Max_j) \tag{7.7}$$

  où

  $\boldsymbol{U}^{\mathbb{N}^+}$ désigne la loi uniforme discrète dans l'espace $\Omega = \{Min_j, \ldots, Max_j\}$, inclus dans $\mathbb{N}^+$

avec la probabilité :

$$\mathcal{P}(N_j^t) = \begin{cases} \frac{1}{(Max_j - Min_j)+1} & \text{si} \quad Min_j \leq N_j^t \leq Max_j \\ 0 & \text{sinon} \end{cases}$$

- **Activation du gestionnaire des tâches**

  La figure 7.7 représente le schéma fonctionnel du gestionnaire des tâches d'un nœud i. σ_i désigne le paramètre de la loi de distribution aléatoire qui définie l'activité des tâches au sein du nœud i :

$$X_{\sigma_i} \rightsquigarrow \boldsymbol{U}^{\mathbb{R}^+}([0, \sigma_i]) \tag{7.8}$$

  où $\boldsymbol{U}^{\mathbb{R}^+}([0, \sigma_i])$ désigne la loi de distribution uniforme continue positive sur $[0, \sigma_i] \subset \mathbb{R}^+$.
  À l'instant t, si $(X_{\sigma_i} > \sigma_i)$, le gestionnaire des tâches est inhibé et aucune tâche n'est exécutée durant le pas de temps $t+1$. Le gestionnaire des tâches est uniquement actif si $(X_{\sigma_i} < \sigma_i)$.

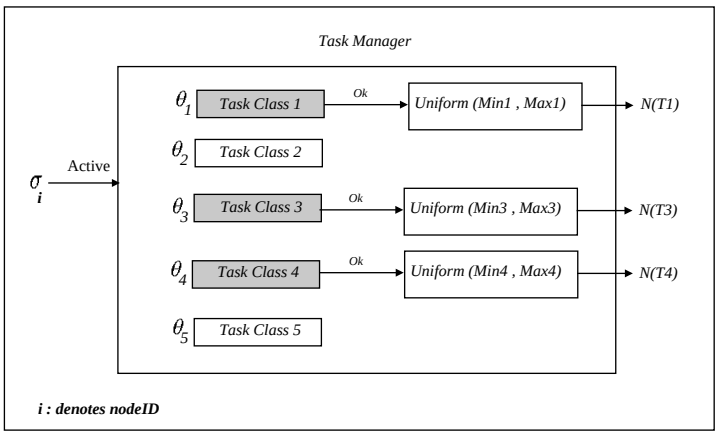


FIG. 7.7 – Gestionnaire de tâches.

- **Sélection des classes de tâches**

  Cette étape suppose que le gestionnaire de tâches soit actif, auquel cas il est donc possible de lancer l'exécution de tâches durant le pas de temps $t+1$. Les classes de tâches à activer, c'est-à-dire les classes candidates à l'exécution pendant le pas de temps $t+1$, suivant les paramètres $\theta_1, \theta_2, \ldots$ respectivement associés aux classes de tâches $T_1, T_2, \ldots$.

- **Sélection du nombre de tâches à exécuter**

  Pour chacune des classes de tâches candidates, on choisi aléatoirement et en fonction des propriétés physiques du nœud i, un nombre aléatoire de tâches à exécuter durant le pas de temps $t+1$.

- **Simulation de l'exécution d'une tâche**

  La figure 7.8 décrit la méthode adoptée pour simuler l'exécution d'une tâche quelconque.

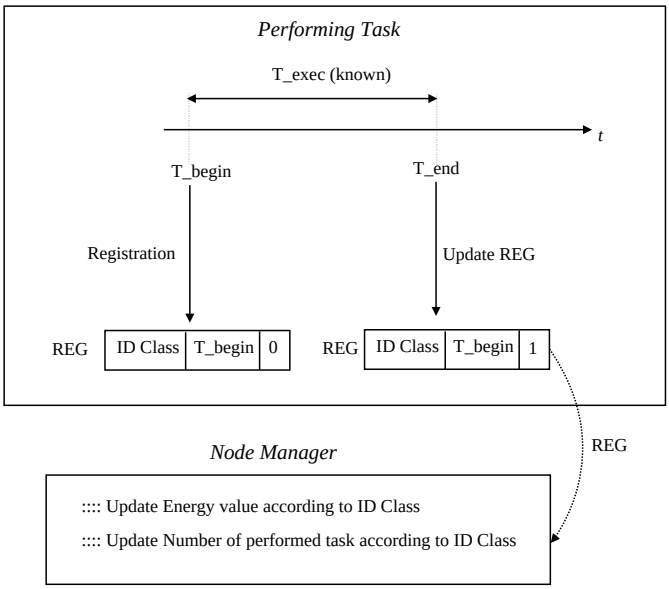


FIG. 7.8 – Simulation de l'exécution d'une tâche.

Gestionnaire des émissions / réceptions

Pour tout nœud i, l'envoi de requêtes de routage de données (message de type 2) est régi par le paramètre β_i. Le nœud sélectionne un nœud destinataire selon une distribution uniforme et lui transmet la requête de routage (cf. figure 7.9). Évidemment, les lois d'arrivées de messages dépendent de façon asynchrone des lois régissants les sorties des nœuds prédécesseurs desquels le nœud i reçoit des messages.

Les figure 7.10 et 7.11 complètent la description du gestionnaire des émissions et des réceptions.

Générateur des tables de routage

Le module « Génération des tables de routage » (voir la figure 7.12) est en cours d'implantation. Pour chacune des nœuds i du réseau, le module génère tous les chemins d'accès optimaux à tout nœud v. À chaque chemin générique ρ_i, on associe une fonction de coût $\mathcal{C}(\rho_i)$ qui modélise la notion de plus court chemin, celui-ci pouvant être sujet à plusieurs contraintes :

- énergie d'acheminement minimale ;
- distance minimale ;
- etc.

Dans la littérature de la théorie des graphes il existe de nombreux algorithmes de recherche du plus court chemin entre deux nœuds. Nous utilisons soit l'algorithme de *Dijkstra*, soit l'algorithme par *construction d'arbres* [128] pour générer les tables de routage optimales. Ainsi, le générateur des tables de routage permet à tout nœud i d'avoir une table R_i des chemins d'accès optimaux vers tous les autres nœuds du réseau.

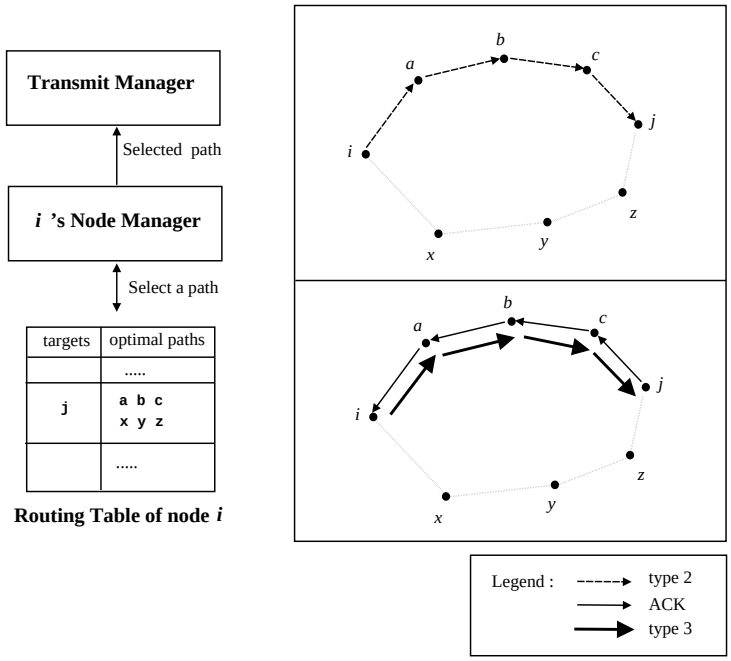


FIG. 7.9 – Routage de données.

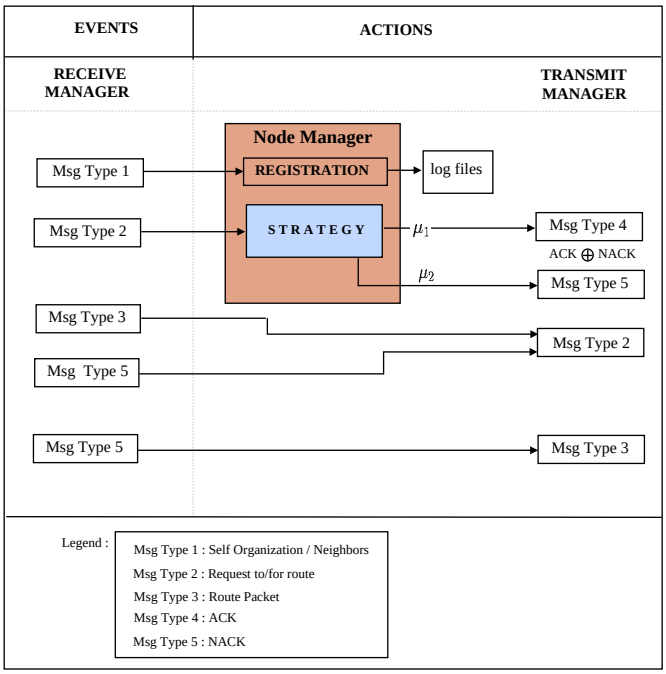


FIG. 7.10 – Gestionnaire de messages.

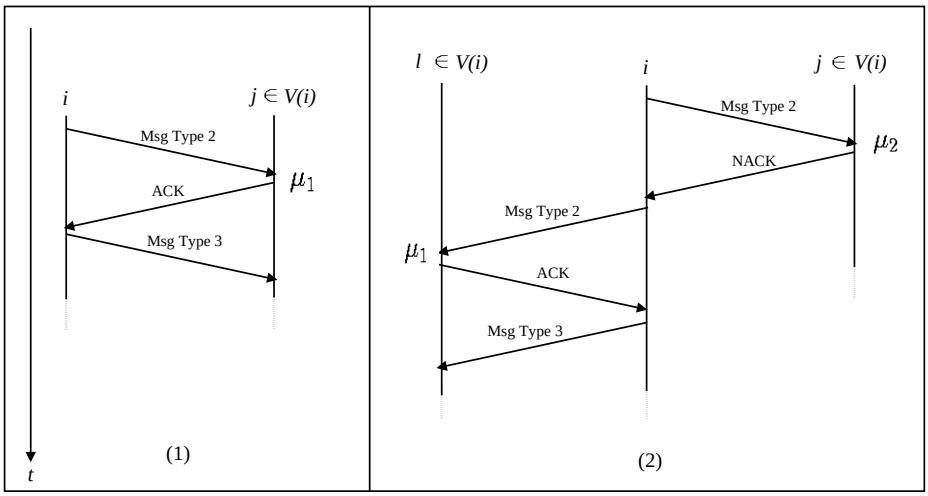


FIG. 7.11 – (1) Routage avec ACK (2) Refus de routage

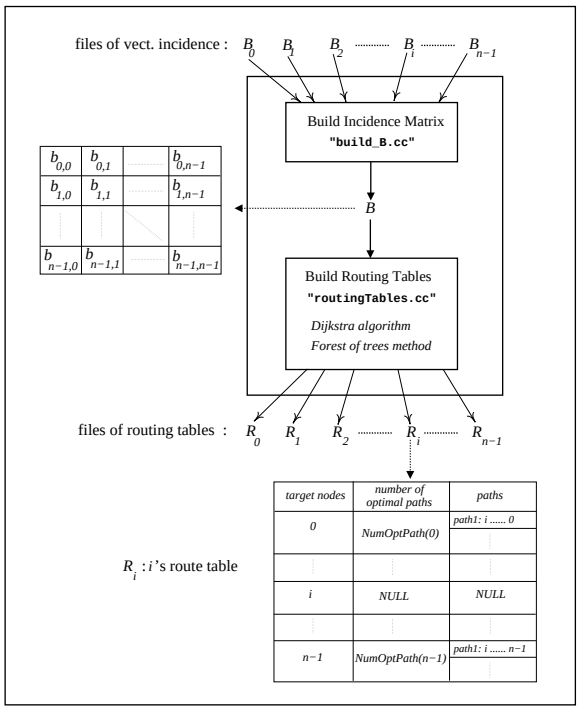


FIG. 7.12 – Génération des tables de routage.

Gestion de l'ajout / suppression de nœuds

La maintenance des réseaux de capteurs consiste en un processus qui assure le maintien des fonctionnalités globales à des coûts moindres. Dans cette partie nous discutons deux points essentiels :

- Gestion suite à d'éventuelles pannes au niveau nœuds.
- Gestion après déploiement de nœuds additionnels

Il arrive que des nœuds du réseau tombent en panne d'énergie ou qu'une défaillance matérielle les mettent hors fonctionnement, ces nœuds ne font donc plus parti du réseau, et risquent de pénaliser le fonctionnement global de celui-ci. En effet, la perte d'un nœud dans le réseau entraine systématiquement sa mise en quarentaine et donc devient non fonctionnel, ce qui implique manifestement la coupure des liaisons avec ses voisins physiques. Les données en provenance des nœuds voisins qui devraient être acheminées vers le(s) collecteur(s) devraient être acheminés via d'autre routes, d'où la necéssité de génération de nouvelles tables de routages en tenant en compte les nœuds mis hors-service.

Pour ce faire, plusieurs stratégies peuvent être adoptées :

⋆ Dès qu'un nœud atteind un niveau d'énergie relativement bas, il diffuse le message de retrait $msg - type_6$. Dès réception du message de retrait, le nœud sera retiré des tables de routage ;

⋆ Néanmoins, la suppression de nœuds dans le réseau ne pose pas problème avec les mécanismes de relai « Flooding / Gossiping » étant donné que le routage se fait par diffusion, le nœud émetteur ne doit pas systématiquement avoir connaissance sur les nœuds voisins.

Dans les réseaux à forte activité, des nœuds additionnels peuvent être déployés afin d'étendre le réseau oubien palier à certain problèmes ; des problèmes liés à la connectivité globale dûe à la mise hors-service de quelques nœuds au sein du réseau. Les mécanismes d'intégration des nouveaux noeuds dans le réseaux peuvent être coûteux en énergie, notement lors de la phase d'auto-organisation avec construction des tables de routage.

7.2.6 Expérimentations

Le réseau considéré, présenté par la figure 7.13 pour les premières expérimentations se compose de 30 nœuds hétérogènes, numérotés de 0 à 29, répartis uniformément dans 4 classes (voir la table 7.3). Les nœuds sont déployés aléatoirement dans un espace à deux dimensions de taille 15 $m \times 20$ m. À l'instant t_0, chaque nœud i dispose d'ensemble $T_i^{t_0}$ de classes de tâches à exécuter tout au long de sa durée de vie (voir la table 7.4).

Évolution du niveau d'énergie au cours du temps

La table 7.5 donne quelques paramètres générés lors de la phase d'initialisation du réseau (seules certaines valeurs sont présentées).

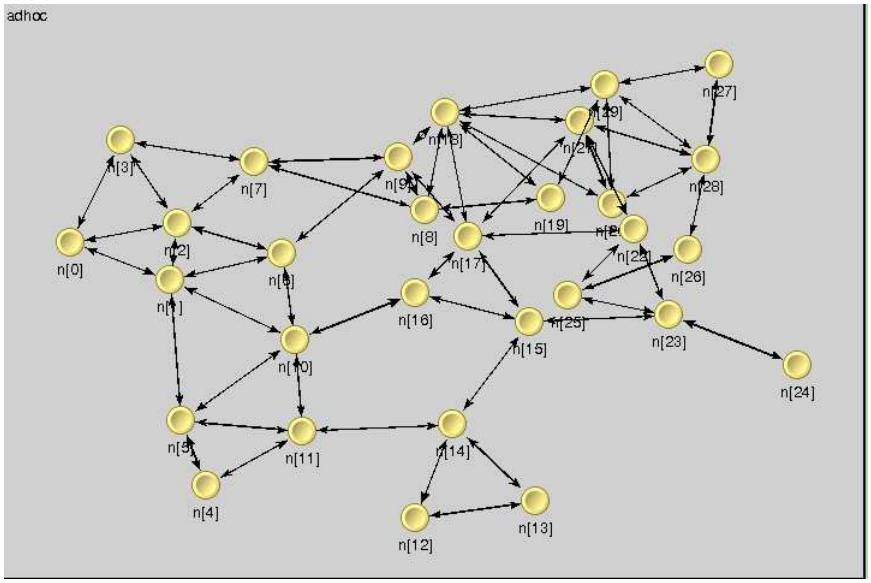


FIG. 7.13 – Réseau de 30 nœuds.

ID « classe nœud »	Description	E_0^t	Interval des β param. de $\boldsymbol{U}^{\mathbb{R}^+}$	Interval des σ pram. de $\boldsymbol{U}^{\mathbb{R}^+}$
0	PALM	150	$[0.05, 0.45]$	$[0.05, 0.45]$
1	X-Palm	300	$[0.05, 0.45]$	$[0.05, 0.45]$
2	Y-Device	450	$[0.20, 0.60]$	$[0.20, 0.60]$
3	Laptop	900	$[0.30, 0.70]$	$[0.30, 0.70]$

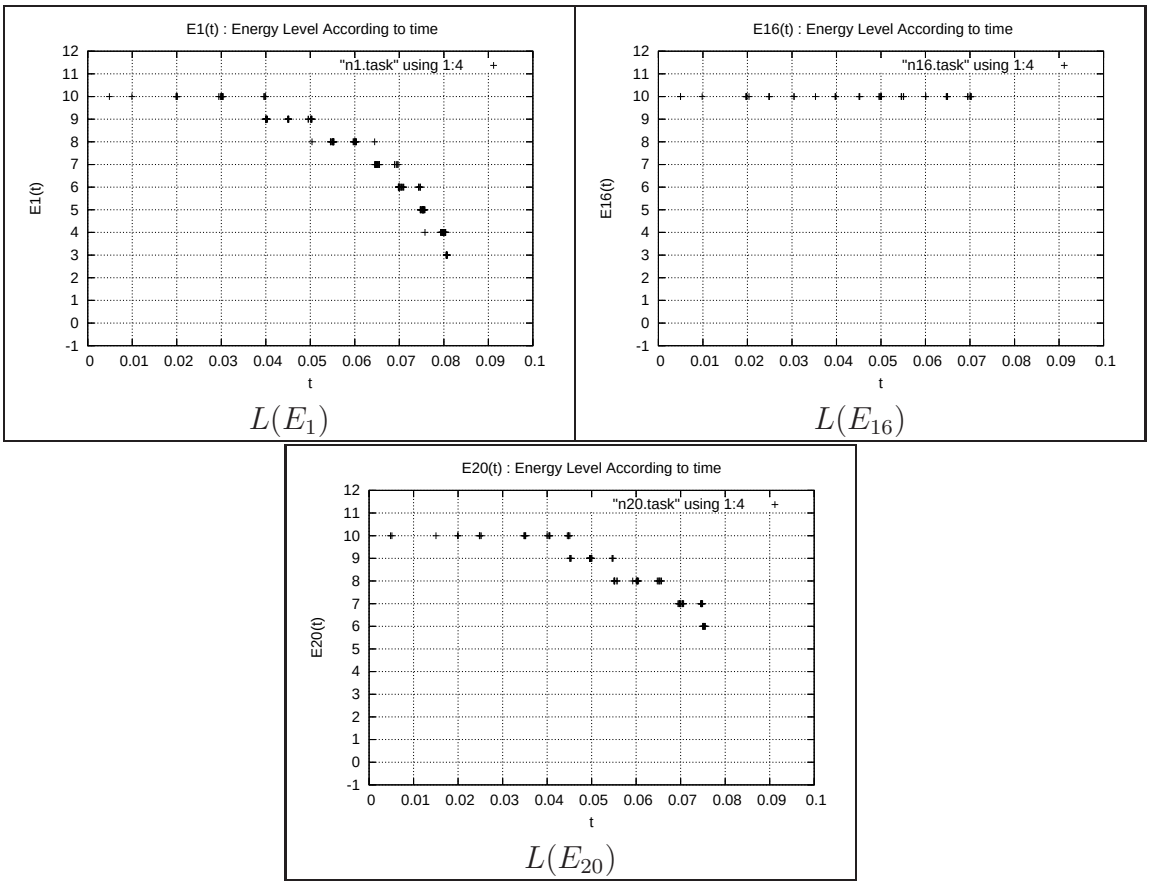
TAB. 7.3 – Description des classes de nœuds.

ID classe	Description	tps d'exec	taille Stat.	$[Min, Max] \subset \mathbb{N}^+$	γ	θ	$N_{i,j}^t \in [min, max] \subset \mathbb{N}^+$
0	systemTask	180	700	$[10^4, 5 \times 10^3]$	0.01	0.8	$[10, 50]$
1	imageProcess	400	1200	$[5 \times 10^2, 10^3]$	0.2	0.4	$[1, 20]$
2	soundProcess	180	1700	$[500, 1000]$	0.3	0.4	$[1, 20]$
3	videoProcess	500	2900	$[100, 500]$	0.6	0.4	$[1, 10]$
4	bioProblem	1500	4900	$[50, 100]$	1.5	0.2	$[1, 5]$

TAB. 7.4 – Description des classes de tâches.

Nœud i	class ID	PosX	PosY	# Tâches de classe0	# Tâches de classe1	# Tâches de classe2	# Tâches de classe3	# Tâches de classe4
0	3	14	14	14420	-	-	362	-
7	3	11	9	32459	529	937	274	70
9	2	6	20	32302	789	646	129	99
..	..	..	..	..	..	..	..	..
29	3	11	4	37822	610	-	463	91

TAB. 7.5 – Paramètres de simulation.



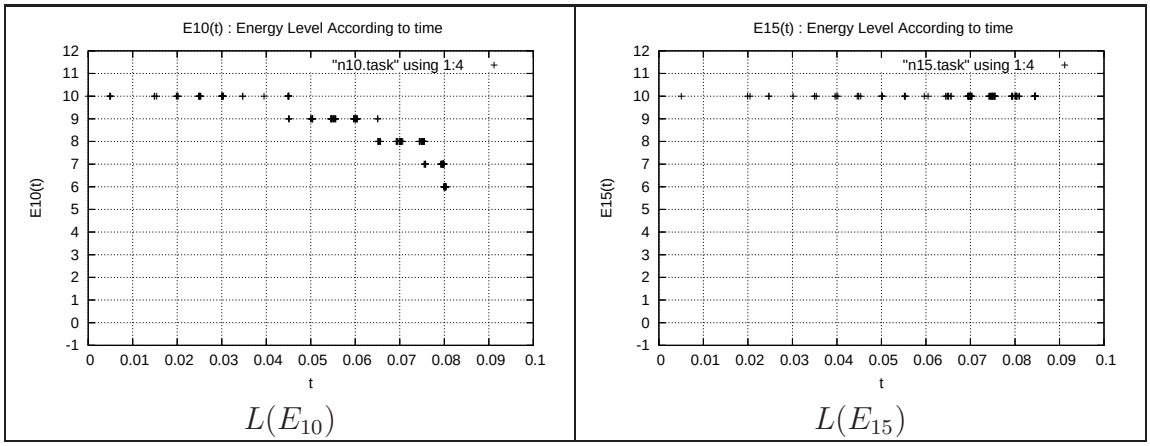
TAB. 7.6 – Variations du niveau d'énergie de quelques nœuds de la classe Ψ_0.

Les variations dans le temps des niveaux d'énergie des nœuds du réseau sont graphiquement représentées dans les courbes des tables 7.6, 7.7, 7.8 et 7.9. Les activités au sein d'une même classe de nœuds diffèrent d'une nœud à un autre. En effet, on peut par exemple remarquer que le nœud 1 est un nœud ayant une forte activité, à l'inverse du nœud 16. Cette différence de niveau d'activité se répercute naturellement sur le plan énergétique, puisqu'on peut voir qu'à l'instant $t_{0.07}$ on a $Level_1(t_{0.07}) = 3$ et $Level_{16}(t_{0.07}) = 10$.

Bien entendu, le niveau d'énergie d'un nœud définit implicitement sa durée de vie et par voie de conséquence, en considérant tous les nœuds, celle du réseau. La perte d'un nœud peut engendrer un dysfonctionnement global du réseau, plus généralement la perturbation du réseau est fortement liée au taux de pertes de nœuds. Ainsi, dans l'exemple de la figure 7.13, la perte du nœud 14 se traduit par l'isolement des nœuds 12 et 13 du reste du réseau.

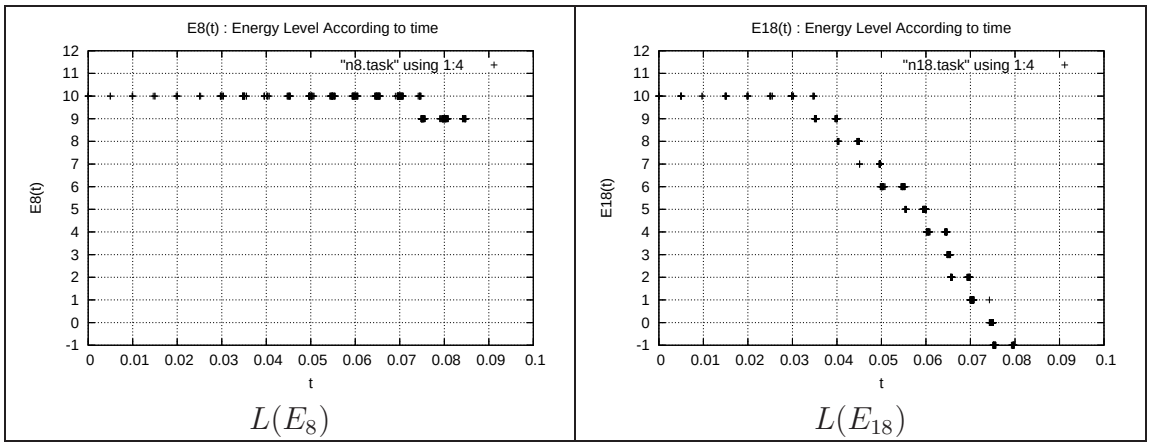
Le niveau d'énergie nous renseigne également sur le taux d'activité de chaque nœud du réseau. Pour un nœud i, la cadence d'activité Γ_i est complexe à modéliser, elle est fonction de :

$$\Gamma_i = \Phi_i \left(\mathcal{F}^i_{Tasks}, \mathcal{F}^i_{Send}, \mathcal{F}^i_{Routing}, \mathcal{F}^i_{V_i} \right) \tag{7.9}$$



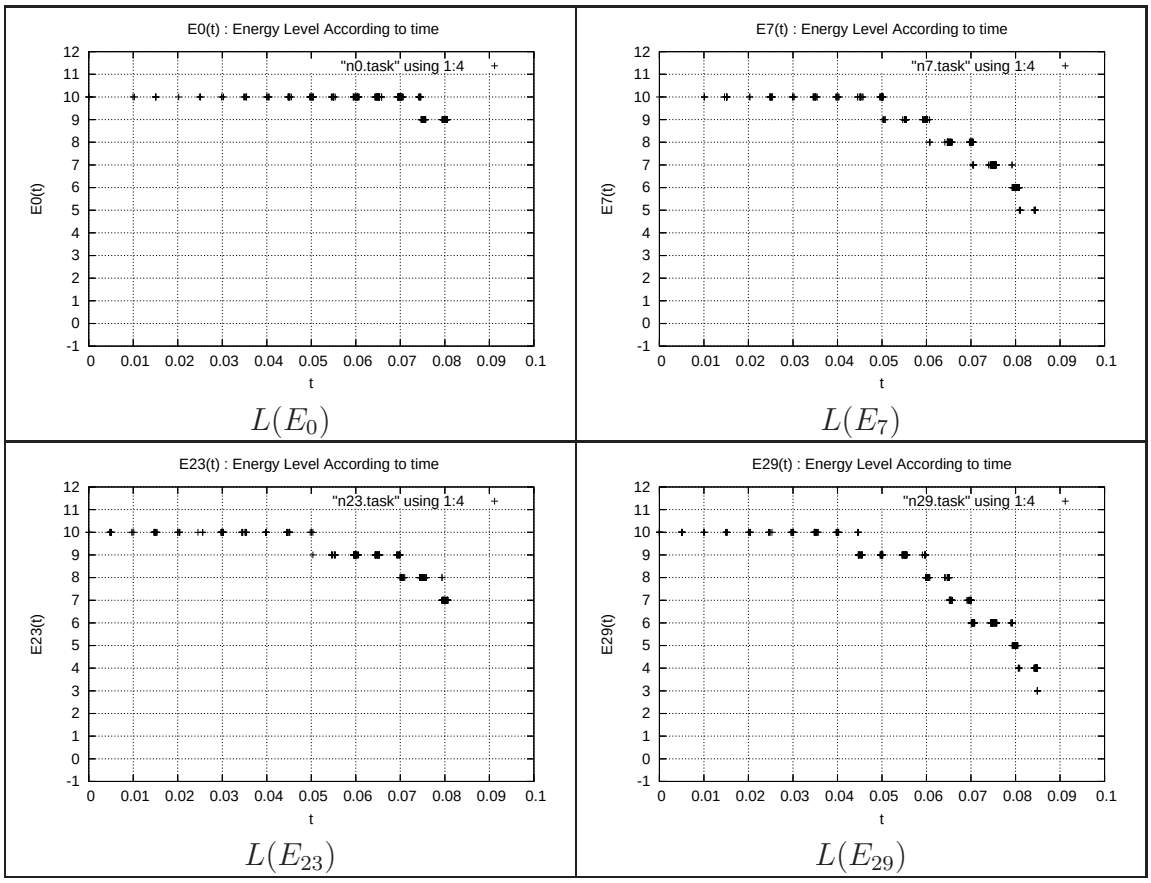
$L(E_{10})$ $L(E_{15})$

TAB. 7.7 – Variations du niveau d'énergie de quelques nœuds de la classe Ψ_1.



$L(E_8)$ $L(E_{18})$

TAB. 7.8 – Variations du niveau d'énergie de quelques nœuds de la classe Ψ_2.



TAB. 7.9 – Variations du niveau d'énergie de quelques nœuds de la classe Ψ_3

avec

$\mathcal{F}^i_{Tasks}$: fonction modélisant l'exécution de tâches d'un nœud i ;

$\mathcal{F}^i_{Send}$: fonction des activités d'envoi ;

$\mathcal{F}^i_{Routing}$: fonction des activités de routage ;

$\mathcal{F}^i_{V_i}$: activités dans le voisinage V_i

Les variations du niveau d'énergie au cours du temps de certains nœuds, observables sur les courbes présentées dans les tables 7.6 à 7.9, résultent des différentes activités d'un nœud. Ces activités sont :

- exécution de tâches (applications) ;
- envoi et réception de messages :
  - messages d'initialisation ;
  - requêtes de demande de services ;
  - messages de contrôle ;
  - routage des données à échanger.

Analyse des activités

Les courbes présentées dans les tables 7.10, 7.11, 7.12, 7.13 montrent les activités liées à l'exécution de tâches (dans les nœuds du réseau) en fonction du temps. Cette activité varie d'une classe de nœuds à une autre, mais aussi au sein des nœuds d'une même classe. Plus précisément, les courbes représentent graphiquement la variation dans le temps du nombre total de tâches exécutées, toutes classes de tâches confondues. En d'autres mots, il s'agit de la quantité :

$$\sum_{\pi \geq 0} \sum_{k \geq 0} N_k^\pi \tag{7.10}$$

où

N_k^π désigne le nombre de tâches de la classe d'indice k dans u^π, exécutées durant $t = \pi.$;

u^π désigne l'ensemble des classes de tâches disponibles dans le nœud i à l'instant π.

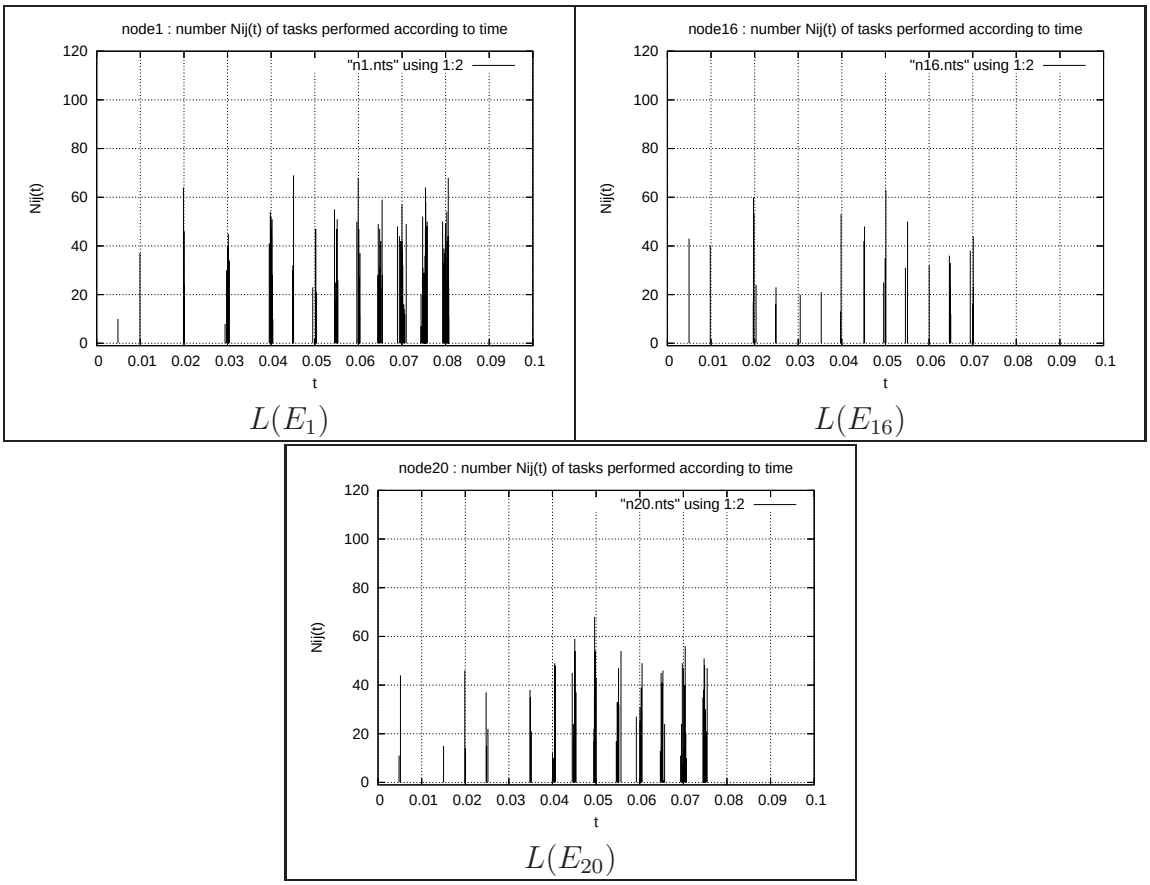
La migration de tâches engendre pour un nœud i une dynamique au sein même des classes de tâches, aussi les classes de tâches ne sont pas fixes. Ainsi, un nœud i peut disposer à t_0 d'un nombre $|u^0|$ de classes de tâches et d'un nombre $|u^t|$ de classes à t.

Exemple :

à t_0, un nœud i dispose des classes de tâches suivantes : $u^{t_0} = \{T_0, T_1, T_3\}$

à t_0 : il execute N_0^0 tâches de la classe T_0 et N_3^0 tâches de la classe T_3

à t_1, le nœud i, reçoit des tâches T_2 à exécuter (tâches déportées), on a donc : $u^{t_1} = \{T_0, T_1, T_3, T_2\}$



TAB. 7.10 – Activités liées à l'exécution de tâches pour quelques nœuds de Ψ_0.

à t_1 : il execute N_2^1 tâches de cette classe et N_0^1 de la classe 0

à t_2 : il n'execute aucun tâche,

à t_3 : il execute N_0^3 tâches de la classe T_0, N_1^3 tâches T_1, N_3^3 tâches T_3 et N_2^3 tâches T_2

dans l'intervalle $[t_0 \ \ldots \ t_3]$ le nœud i a exécuté :

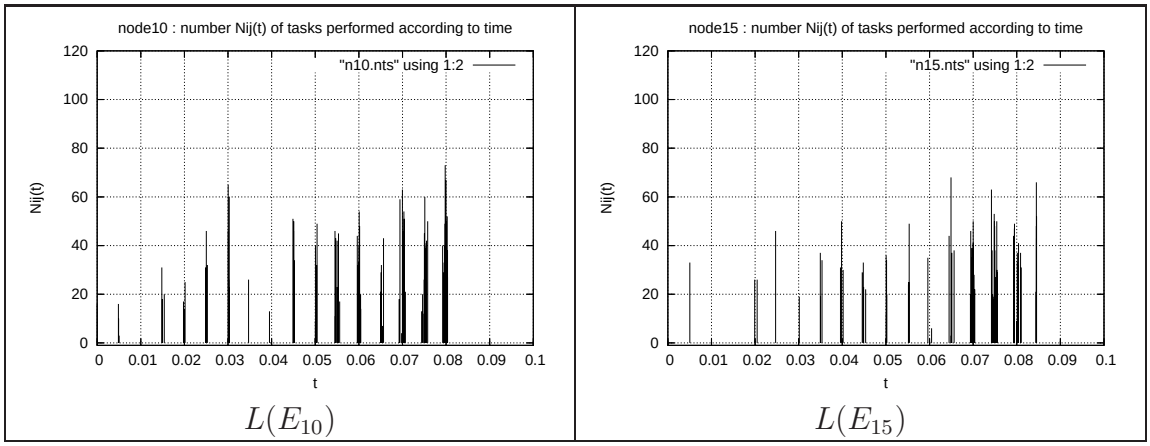
$$N^{t_0} = N_0^0 + N_3^0$$

$$N^{t_1} = N_2^1 + N_0^1$$

$$N^{t_2} = 0$$

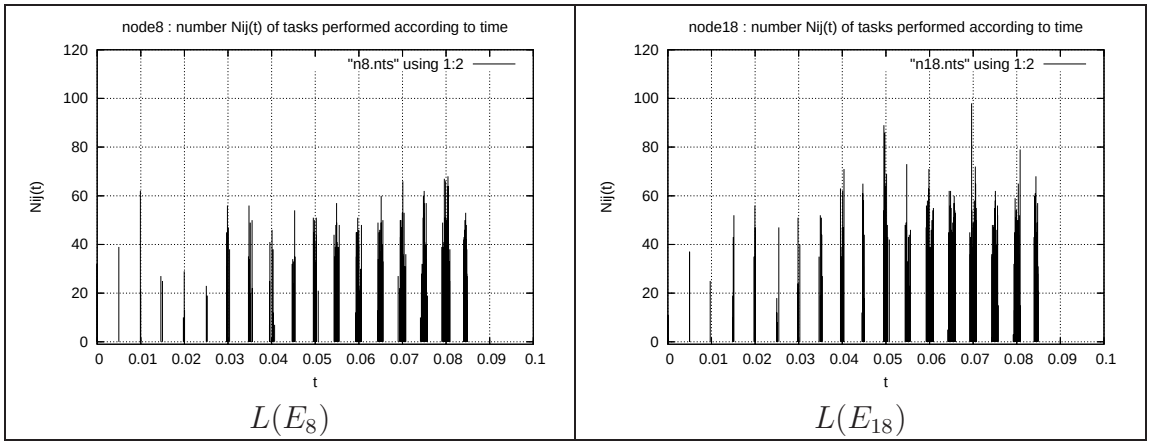
$$N^{t_3} = N_0^3 + N_1^3 + N_3^3 + N_2^3$$

Le faible variation du niveau d'énergie du nœud 16 (table 7.6), peut se justifier par le faible taux d'activité d'exécution de tâches au sein de ce nœud. Inversement, la forte baisse du niveau d'énergie du nœud 18, se justifie notamment par la forte activité au sens des tâches executées (voir table 7.12).



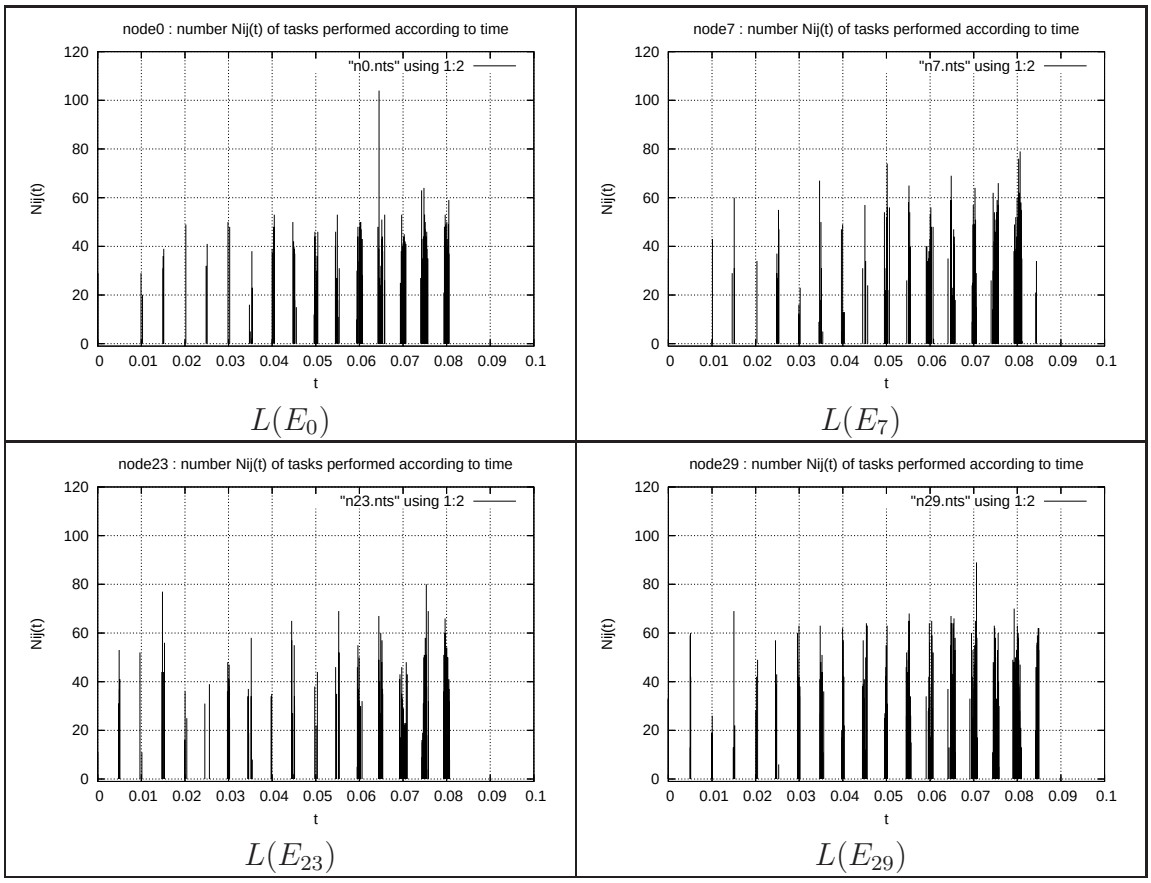
$L(E_{10})$ $L(E_{15})$

TAB. 7.11 – Activités liées à l'exécution de tâches pour quelques nœuds de Ψ_1.



$L(E_8)$ $L(E_{18})$

TAB. 7.12 – Activités liées à l'exécution de tâches pour quelques nœuds de Ψ_2.



TAB. 7.13 – Activités liées à l'exécution de tâches pour quelques nœuds de Ψ_3.

7.2.7 Conclusion

Le simulateur présenté pour le cadre dynamique n'est pas encore pleinement opérationnel. Cependant, les résultats présentés sont intéressants, puisque les courbes obtenues convergent vers les besoins recherchés (en terme de simulation) pour les expérimentations.

Conclusion et perspectives

Le domaine des réseaux de capteurs sans fil est un domaine en pleine expansion, apparu à la suite du développement et de la convergence de nombreux domaines technologiques. Ces réseaux sont amenés à se développer en raison du vaste champ d'applications potentielles. Dans ce contexte, un nœud capteur est doté d'une source d'énergie non renouvelable et a des capacités de traitement limitées. Aussi, une problématique majeure dans le domaine des réseaux de capteurs est la gestion de l'énergie, et plus particulièrement la prolongation de la durée de vie des nœuds capteurs. Il est donc indispensable d'établir de nouveaux algorithmes / protocoles qui tiennent compte de la contrainte énergétique. Dans cette thèse, nous avons proposé et développé une nouvelle approche permettant d'augmenter la durée de vie d'un réseau de capteurs hétérogènes.

La répartition aléatoire des ressources énergétiques et l'indéterminisme des activités entre les nœuds sont des contraintes fortes. D'une part, certains nœuds au sein d'un même réseau se trouvant dans des zones à densité faible sont plus enclins aux défaillances énergétiques, étant donné qu'ils seront plus sollicités que ceux des zones denses. D'autre part, des nœuds peuvent être dans l'impossibilité d'exécuter la totalité de leur charge, à cause de ressources énergétiques insuffisantes. Ces contraintes risquent donc d'entraîner l'éclatement du réseau en une multitude de groupes de nœuds capteurs non coopérants et disjoints au sens de la connectivité globale.

Afin d'augmenter la durée de vie du réseau, nous avons proposé une approche distribuée qui consiste à transférer de la charge entre nœuds communiquants directement (nœuds voisins), dans le but de décharger les nœuds ayant une forte activité. L'idée sous-jacente est d'équilibrer les activités entre les nœuds en tenant compte de l'énergie dont ils disposent. Pour cela, nous définissons un ratio d'énergies qui permet de quantifier la participation d'un nœud à l'activité du réseau par rapport à l'énergie dont il dispose. Ce ratio, qui évolue avec le temps, correspond plus précisément au rapport entre l'énergie nécessaire au nœud considéré pour exécuter la totalité de sa charge et la provision d'énergie dont il dispose effectivement. Ainsi, faire en sorte que tout nœud participe proportionnellement à l'énergie dont il dispose, revient à équilibrer ce ratio à travers le réseau.

Pour atteindre cet objectif, nous nous sommes inspirés des mécanismes d'équilibrage de charge proposés dans le contexte des systèmes distribués. Notre algorithme d'équilibrage reprend le principe de l'algorithme de diffusion du premier ordre. Il équilibre les ratios en migrant des tâches entre nœuds voisins, il tient compte de l'hétérogénéité des nœuds capteurs et peut gérer des réseaux de topologies dyna-

miques. Comme nous l'avons prouvé, la convergence des valeurs de ratio est assurée. Naturellement, la vitesse de convergence sera fortement liée à la topologie et à la taille du réseau considéré.

Le gain en durée de vie, que permet d'obtenir l'algorithme, a été évalué via des expérimentations. Pour ce faire, nous avons développé des simulations avec le simulateur à évènements discrets OMNeT++. Le scénario consiste en un réseau de capteurs multimédias hétérogènes utilisés à des fins de détection d'intrusion. Diverses topologies réseaux ont été étudiées : réseau complètement connecté, réseau linéaire et réseau mixte. Les deux premières topologies correspondent en quelque sorte à des cas extrêmes, néanmoins elles permettent d'étudier le comportement de l'algorithme. Les résultats obtenus mettent en évidence une augmentation de la durée de vie du premier nœud défaillant et donc du réseau de capteurs. Par exemple, dans le cas d'un réseau complètement connecté de 10 nœuds nous avons observé un gain moyen de plus de 55%, contre un peu moins de 25% pour un réseau linéaire de même taille. Cela s'explique surtout par un plus grand nombre d'itérations nécessaires à l'algorithme pour converger dans le cas linéaire. Enfin, les simulations montrent également que grâce à notre algorithme tout nœud aura une charge qu'il pourra exécuter avant de tomber en panne. Ce point est très important du point de vue de la qualité de service (QoS).

Perspectives de recherche

À la suite de nos travaux, plusieurs perpectives peuvent être envisagées.

- Tout d'abord, il serait fort intéressant d'étudier la prise en compte d'une charge dynamique au sein du réseau. En effet, dans le travail présenté nous avons considéré que chaque nœud avait une charge statique qui était fixée à l'initialisation et aucune nouvelle tâche ne pouvait être produite.
- L'intégration lors du calcul du nombre de tâches à envoyer par un nœud à un de ses voisins de critères tels que l'énergie consommée lors de l'envoi d'un type tâche permettra sans doute d'obtenir de meilleures performances.
- Il serait également intéressant d'étudier les performances de l'algorithme en considérant d'autres définitions de la notion de durée de vie d'un réseau de capteurs. Par exemple, le réseau pourrait être supposé défaillant à la suite de la perte d'un sous-ensemble de nœuds et non plus d'un seul nœud.
- Finalement, dans certaines applications la mobilité des nœuds pourrait être envisagée comme solution au déploiement de nœuds additionnels pour remplacer les nœuds défaillants. Aussi, nous envisageons d'étudier l'impact de la mobilité sur l'algorithme développé et de proposer une approche qui tienne compte des densités de nœuds dans le réseau.

Bibliographie

[1] Ian F. Akyildiz, Weillan Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless Sensor Networks : A Survey. *Computer Networks Journal*, 38(4) :393–422, March 2002.

[2] Ian F. Akyildiz, Weillan Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A Survey on Sensor Networks. *IEEE Communication Magazine*, 40(8) :102–116, August 2002.

[3] Kay Römer and Friedemann Mattern. The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications*, 11(6) :54–61, December 2004.

[4] Alan M. Mainwaring, David E. Culler, Joseph Polastre, Robert Szewczyk, and John Anderson. Wireless Sensor Networks for Habitat Monitoring. In *WSNA'02 : Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 88–97. ACM Press, September 2002.

[5] David Culler, Deborah Estrin, and Mani Srivastava. Overview of Sensor Networks. *IEEE Computer Magazine*, 37(8) :41–49, August 2004.

[6] Aleksandar Milenkovic, Chris Otto, and Emil Jovanov. Wireless Sensor Networks for Personal Health Monitoring : Issues and an Implementation. *Computer Communications*, 29(13-14) :2521–2533, 2006.

[7] Loren Schwiebert, Sandeep K.S. Gupta, and Jennifer Weinmann. Research Challenges in Wireless Networks of Biomedical Sensors. In *MobiCom'01 : Proceedings of the 7th annual International Conference on Mobile Computing and Networking*, pages 151–165. ACM Press, 2001.

[8] Giuseppe Amato, Stefano Chessa, Fabrizio Conforti, Alberto Macerata, and Carlo Marchesi. *Networking and Data Management for Health Care Monitoring of Mobile Patients*, volume 117 of *Studies in Health Technology and Informatics*. IOS Press, October 2005.

[9] Cliodhna Ní Scanaill, Sheila Carew, Pierre Barralon, Norbert Noury, Declan Lyons, and Gerard M. Lyons. A Review of Approaches to Mobility Telemonitoring of the Elderly in Their Living Environment. *Annals of Biomedical Engineering*, 34(4) :547–563, 2006.

[10] Richard Beckwith, Dan Teibel, and Pat Bowen. Report from the Field : Results From an Agricultural Wireless Sensor Network. In *LCN'2004 : Proceedings of the 29th Annual IEEE Conference on Local Computer Networks*, pages 471–478. IEEE Computer Society, November 2004.

[11] Jenna Burrell, Tim Brooke, and Richard Beckwith. Vineyard Computing : Sensor Networks in Agricultural Production. *IEEE Pervasive Computing*, 03(1) :38–45, 2004.

[12] Online. Intel Research : Proactive Agriculture, Sensor Networks for understanding Site Variability. http ://www.intel.com/labs.

[13] Geoffrey Werner-Allen, Konrad Lorincz, Matt Welsh, Omar Marcillo, Jeff Johnson, Mario Ruiz, and Jonathan Lees. Deploying a Wireless Sensor Network on an Active Volcano. *IEEE Internet Computing*, 10(2) :18–25, 2006.

[14] Miklos Maroti, Gyula Simon, Akos Ledeczi, and Janos Sztipanovits. Shooter Localization in Urban Terrain. *IEEE Computer Magazine*, 37(8) :60–61, 2004.

[15] Anish Arora, Prabal Dutta, Sandip Bapat, Vinod Kulathumani, Hongwei Zhang, Vinayak Naik, Vineet Mittal, Hui Cao, Murat Demirbas, Mohamed G. Gouda, Young ri Choi, Ted Herman, Sandeep S. Kulkarni, Umamaheswaran Arumugam, Mikhail Nesterenko, Adnan Vora, and Mark Miyashita. A Line in the Sand : A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks*, 46(5) :605–634, 2004.

[16] Guy Pujolle. *Les Réseaux*. Editions Eyrolles, Juillet 2004.

[17] Cintia B. Margi, Katia Obraczka, and Roberto Manduchi. Characterizing Energy Consumption in a Visual Sensor Network Testbed. In *TridentCom'2006 : Proceedings of the 2nd International IEEE/Create-Net Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*. IEEE Computer Society, March 2006.

[18] Online. Crossbow Wireless Sensor Networks. http ://www.xbow.com.

[19] Ian F. Akyildiz and Ismail H. Kasimoglu. Wireless Sensor and Actor Networks : Research Challenges. *Ad Hoc Networks*, 2(4) :351–367, 2004.

[20] C. Evans-Pughe. Bzzz zzz [ZigBee Wireless Standard]. *IEEE Review*, 49 :28–31, 2003.

[21] H.C. Huang, J.W. Din, and Y.M. Huang. An Implementation of Battery-Aware Wireless Sensor Network Using ZigBee for Multimedia Service. In *Proceedings of IEEE International Conference on Consumer Electronics*, pages 369–370. IEEE Computer Society, January 2006.

[22] Online. ZigBee Alliance - Wireless Control That Simply Works. http ://www.zigbee.org.

[23] Geoff V. Merrett, Alex S. Weddell, Nick R. Harris, Neil M. White, and Bashir M. Al-Hashimi. The Unified Framework for Sensor Networks : A Systems Approach. Technical Report UF1, School of Electronics and Computer Science, University of Southampton, September 2006.

[24] Özgür B. Akan and Ian F. Akyildiz. Event-to-Sink Reliable Transport in Wireless Sensor Networks. *IEEE/ACM Trans. Netw.*, 13(5) :1003–1016, 2005.

[25] Vinod Pathari, Manu Jose, G. R. Ragul, and P. Muhammed Irshad. K-RTP : A Reliable Transport Layer Protocol for Wireless Sensor Networks. In *Proceedings of the Sixth IASTED International Multi-Conference on Wireless and Optical Communications*. IASTED/ACTA Press, July 2006.

[26] *802.15.4-2006 IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 15.4 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs).*

[27] Marina Petrova, Janne Riihijärvi, Petri Mähönen, and Saveria Labella. Performance Study of IEEE 802.15.4 Using Measurements and Simulations. In *Proceedings of IEEE Wireless Communication and Networking Conference*, pages 487–492. IEEE Computer Society, April 2006.

[28] Wei Ye and John Heidemann. *Medium Access Control in Wireless Sensor Networks*, chapter 4, pages 73–92. Kluwer Academic Publishers, 2004.

[29] Ilker Demirkol, Cem Ersoy, and Fatih Alagöz. MAC Protocols for Wireless Sensor Networks : A Survey. *IEEE Communications Magazine*, 44(4) :115–121, April 2006.

[30] Wei Ye, John S. Heidemann, and Deborah Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *INFOCOM'2002 : Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE Computer Society, June 2002.

[31] Bao Hua Liu, Nirupama Bulusu, Huan Pham, and Sanjay Jha. CSMAC : A Novel DS-CDMA Based MAC Protocol for Wireless Sensor Networks. In *Proceedings of IEEE GlobeCom Workshops, Wireless Ad hoc and Sensor Networks*, pages 33–38. IEEE Computer Society, November 2004.

[32] Haowen Chan and Adrian Perrig. ACE : An Emergent Algorithm for Highly Uniform Cluster Formation. In *ESWN'2004 : Proceedings of the First European Workshop on Wireless Sensor Networks*, volume 2920. Springer, January 2004.

[33] Malka N. Halgamuge, Siddeswara M. Guru, and Andrew Jennings. Energy Efficient Cluster Formation in Wireless Sensor Networks. In *ICT'2003 : Proceedings of the 10th International Conference on Telecommunications*, volume 2, pages 1571–1576, 2003.

[34] Wendi B. Heinzelman, Anantha Chandraksan, and Hari Balakrishnan. An Application-Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Transactions on Wireless Communications*, 1 :660–670, 2002.

[35] Kilian Weniger and Martina Zitterbart. Address Autoconfiguration in Mobile Ad Hoc Networks : Current Approaches and Future Directions. *IEEE Network Magazine Special issue on 'Ad hoc networking : data communications and topology control'*, July 2004.

[36] Xukai Zou, Byrav Ramamurthy, and Spyros S. Magliveras. Routing Techniques for Wireless Ad Hoc Networks - Classification and Comparison. In *SCI'2002 : Proceedings of the Sixth World Multiconference on Systemics, Cybernetics, and Informatics*, July 2002.

[37] Jamal N. Al-Karaki and Ahmed E. Kamal. Routing Techniques in Wireless Sensor Networks : A Survey. *IEEE Wireless Communications*, 11(6) :6–28, December 2004.

[38] Kemal Akkaya and Mohamed F. Younis. A Survey on Routing Protocols for Wireless Sensor Networks. *Ad Hoc Networks*, 3(3) :325–349, 2005.

[39] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *MobiCom'99 : Proceedings of the 5th annual ACM/IEEE International Conference on Mobile computing and networking*, pages 174–185. ACM Press, 1999.

[40] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed Diffusion : A Scalable and Robust Communication Paradigm for Sensor Networks. In *MobiCom'00 : Proceedings of the 6th annual International Conference on Mobile computing and networking*, pages 56–67. ACM Press, 2000.

[41] Arati Manjeshwar and Dharma P. Agrawal. TEEN : A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks. In *IPDPS'01 : Proceedings of the 15th International Parallel and Distributed Processing Symposium Workshops*, volume 3, page 189. IEEE Computer Society, April 2001.

[42] Arati Manjeshwar and Dharma P. Agrawal. APTEEN : A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks. In *IPDPS'02 : Proceedings of the 16th International Parallel and Distributed Processing Symposium*. IEEE Computer Society, April 2002.

[43] Ya Xu, John S. Heidemann, and Deborah Estrin. Geography-Informed Energy Conservation for Ad Hoc Routing. In *MobiCom'01 : Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 70–84. ACM Press, 2001.

[44] Arvind Giridhar and P. R. Kumar. Maximizing the Functional Lifetime of Sensor Networks. In *IPSN'2005 : Proceedings of the Fourth IEEE International Symposium on Information Processing in Sensor Networks*, pages 5–12. IEEE Computer Society, April 2005.

[45] Jae-Hwan Chang and Leandros Tassiulas. Energy Conserving Routing in Wireless Ad-Hoc Networks. In *INFOCOM'2000 : Proceedings of the 19st International Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 22–31, 2000.

[46] Vivek Rai and Rabi N. Mahapatra. Lifetime Modeling of a Sensor Network. In *DATE'2005 : Proceedings of Design, Automation and Test in Europe*, pages 202–203. IEEE Computer Society, 2005.

[47] Petar Djukic and Shahrokh Valaee. Maximum Network Lifetime in Fault Tolerant Sensor Networks. In *GLOBECOM'05 : Proceedings of the IEEE Global Telecommunications Conference*, volume 5, November 2005.

[48] Yunxia Chen and Qing Zhao. On the Lifetime of Wireless Sensor Networks. *IEEE Communications Letters*, pages 976–978, November 2005.

[49] Krishna K. Ramachandran and Biplab Sikdar. A population based approach to model network lifetime in wireless sensor networks. *SIGMETRICS Performance Evaluation Review*, 33(2) :21–23, 2005.

[50] Sunil R. Shenoy and Akhilesh Daniel. Intel Architecture and Silicon Cadence - The Catalyst for Industry Innovation. *Technology Intel Magazine*, Internal report white paper, March 2006.

[51] Ravi Nagaraj. Energy-Efficient System Architecture Aims to Improve System Energy Efficiencies. *Technology Intel Magazine*, Internal report white paper, March 2006.

[52] Ofri Wechsler. Inside Intel (r) core^{TM}Microarchitecture : Setting New Standards for Energy-Efficient Performance. *Technology Intel Magazine*, Internal report white paper, March 2006.

[53] A. Keith Bates, Mordechai Rothschild, Theodore M. Bloomstein, Theodore H. Fedynyshyn, Roderick R. Kunz, Vladimir Liberman, and Michael Switkes. Review of Technology for 157-nm Lithography. *IBM Journal of Research and Development*, 45(5) :605–614, 2001.

[54] Rajinder S. Dhaliwal, William A. Enichen, Steven D. Golladay, Michael S. Gordon, Rodney A. Kendall, Jon E. Lieberman, Hans C. Pfeiffer, David J. Pinckney, Christopher F. Robinson, James D. Rockrohr, Werner Stickel, and Eileen V. Tressler. PREVAIL-Electron Projection Technology Approach for Next-Generation Lithography. *IBM Journal of Research and Development*, 45(5) :615–638, 2001.

[55] Rick Bailey, Glen Fox, Jarrod Eliason, Marty Depner, Daesig Kim, Edwin Jabillo, John Groat, John Walbert, Scott Summerfelt, K. R. Udayakumar, John Rodriquez, Keith Remack, K. Boku, and John Gertas. FRAM Memory Technology - Advantages for Low Power, Fast Write, High Endurance Applications. In *ICCD'05 : Proceedings of the 2005 International Conference on Computer Design*, page 485. IEEE Computer Society, 2005.

[56] Kinam Kim and Yoon J. Song. Current and Future High Density FRAM Technology. In *Journal of Integrated Ferroelectrics*, volume 61, pages 3–15, 2004.

[57] G. Danese, F. Leporati, R. Lombardi, M. Nucita, G. Pedrazzini, and G. Ricotti. An Instrument for the Characterization of Voltage and Temperature Profile in NiCd and NiMH Batteries. *Euromicro*, page 178, 1997.

[58] Peng Rong and Massoud Pedram. An Analytical Model for Predicting the Remaining Battery Capacity of Lithium-Ion Batteries. In *DATE'2003 : Pro-*

ceedings of Design, Automation and Test in Europe, volume 1, page 11148. IEEE Computer Society, 2005.

[59] C. Arbizzani, A. M. Marinangeli, M. Mastragostino, L. Meneghello, T. Hamaid, and A. Guyot. Lithium/Polymer/Polymer Solid-State Rechargeable Batteries. *Journal of Power Sources*, 44 :453–460, April 1993.

[60] Linda Dailey Paulson. Will Fuel Cells replace Batteries in Mobile Devices ? *IEEE Computer*, 36(11) :10–12, 2003.

[61] Chi Ma, Yuanyuan Yang, and Zhenghao Zhang. Constructing Battery-Aware Virtual Backbones in Sensor Networks. In *ICPP'05 : Proceedings of the 2005 International Conference on Parallel Processing*, pages 203–210. IEEE Computer Society, 2005.

[62] Henrik Lipskoch, Karsten Albers, and Frank Slomka. Battery Discharge Aware Energy Feasibility Analysis. In *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis*, pages 22–27. ACM Press, October 2006.

[63] Gloria Kissin. Upper and Lower Bounds on Switching Energy in VLSI. *Journal of the ACM*, 38(1) :222–254, January 1991.

[64] Carla-Fabiana Chiasserini and Michele Garetto. Modeling the Performance of Wireless Sensor Networks. In *INFOCOM'2004 : Proceedings of the 23rd International Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE Computer Society, March 2004.

[65] Robert N. Mayo and Parthasarathy Ranganathan. Energy consumption in mobile devices : Why future systems need requirements-aware energy scaledown. Technical Report HPL-2003-167, Hewlett Packard Laboratories, August 2003.

[66] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani B. Srivastava. Optimizing Sensor Networks in the Energy-Latency-Density Design Space. *IEEE Transactions on Mobile Computing*, 1(1) :70–80, 2002.

[67] Vijay Raghunatha, Curt Schurgers, Sung Park, and Mani B. Srivastava. Energy-Aware Wireless Sensor Network. *IEEE Signal Processing Magazine*, 9(2) :40–50, March 2002.

[68] J. Proakis. *Digital Communications*. 1995. third edition.

[69] Jochen Schiller, Achim Liers, Hartmut Ritter, Rolf Winter, and Thiemo Voigt. ScatterWeb - Low Power Sensor Nodes and Energy Aware Routing. In *HICSS'05 : Proceedings of the 38th Annual Hawaii International Conference on System Sciences - Track 9*. IEEE Computer Society, 2005.

[70] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory. J. Pottie. Protocols for Self-Organization for a Wireless Sensor Networks. *IEEE Personal Communications*, 7(5) :16–27, October 2000.

[71] Umamaheswaran Arumugam. Infuse : a TDMA based reprogramming service for sensor networks. In *SenSys'2004 : Proceedings of the 2nd International*

Conference on Embedded Networked Sensor Systems, pages 281–282. ACM Press, November 2004.

[72] Biao Ren, Junfeng Xiao, Jian Ma, and Shiduan Cheng. An Energy-Conserving and Collision-Free MAC Protocol Based on TDMA for Wireless Sensor Networks. In *MSN'2005 : Proceedings of the 1st International Conference on Mobile Ad-hoc and Sensor Networks*, volume 3794 of *Lecture Notes in Computer Science*, pages 603–612. Springer, December 2005.

[73] Hai gang Gong, Ming Liu, Xiaomin Wang, and Li Xie. An Energy Efficient TDMA Protocol for Event Driven Applications in Wireless Sensor Networks. In *MSN'2006 : Proceedings of the 2nd International Conference on Mobile Ad-hoc and Sensor Networks*, volume 4325 of *Lecture Notes in Computer Science*, pages 638–649. Springer, December 2006.

[74] Sungrae Cho, Kalyani Kanuri, Jin-Woong Cho, Jang-Yeon Lee, and Sun-Do June. Dynamic Energy Efficient TDMA-Based MAC Protocol for Wireless Sensor Networks. In *ICAS/ICNS'2005 : JProceedings of the Joint International Conference on Autonomic and Autonomous Systems / International Conference on Networking and Services*, page 48. IEEE Computer Society, 2005.

[75] Tijs Van Dam and Koen Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *SenSys'03 : Proceedings of the 1st International Conference on Embedded Networked Sensor Dystems*, pages 171–180. ACM Press, November 2003.

[76] Tao Zheng, Sridhar Radhakrishnan, and Venkatesh Sarangan. PMAC : An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *IPDPS'05 : Proceedings of the 19th International Parallel and Distributed Processing Symposium Workshops*, volume 13, page 237a. IEEE Computer Society, April 2005.

[77] Stanislava Soro and Wendi Rabiner Heinzelman. Prolonging the Lifetime of Wireless Sensor Networks Via Unequal Clustering. In *IPDPS'05 : Proceedings of the 19th International Parallel and Distributed Processing Symposium Workshops*. IEEE Computer Society, April 2005.

[78] Jonas Neander, Ewa Hansen, Jukka Mäki-Turja, Mikael Nolin, and Mats Björkman. A TDMA Scheduler for the AROS Architecture. Technical report, March 2006.

[79] Injong Rhee and Jangwon Lee. Energy-efficient route-aware MAC protocols for diffusion-based sensor networks. Technical Report TR-2004-13, Department of Computer Science, North Carolina State University, April 2004.

[80] Tomás Novosad. A New Family of Quadriphase Sequences for CDMA. *IEEE Transactions on Information Theory*, 39(3) :1083, 1993.

[81] Kenneth G. Paterson. On Codes with Low Peak-to-Average Power Ratio for Multi-Code CDMA. Technical Report HPL-2001-115, Hewlett Packard Laboratories, May 2001.

[82] Tao Shu and Marwan Krunz. Joint Power/Rate Optimization for CDMA-Based Wireless Sensor Networks. In *Proceedings of SenMetrics 2005, Third International Workshop on Measurement, Modelling, and Performance Analysis of Wireless Sensor Networks*, July 2005.

[83] Alaa Muqattash, Marwan Krunz, and William E. Ryan. Solving The Near-Far Problem in CDMA-Based Ad-Hoc Networks. *Ad Hoc Networks*, 1(4) :435–453, 2003.

[84] Min Chen, Changyoon Oh, and Aylin Yener. Efficient Scheduling for Delay Constrained Multi-Rate CDMA Systems. In *ISSSTA'06 : Proceedings of IEEE 9th International Symposium on Spread Spectrum Techniques and Applications*, pages 371–375, August 2006.

[85] Woonsik Lee, Daewon Lee, and Hwang Soo Lee. Lifetime Extension of Border Nodes in SMAC-Based Wireless Sensor Networks by Unifying Multiple Sleep Schedules Among Aadjacent Virtual Clusters. In *PE-WASUN'2005 : Proceedings of the 2nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, pages 267–268. ACM Press, 2005.

[86] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *HICSS'00 : Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 8, page 8020. IEEE Computer Society, 2000.

[87] David Braginsky and Deborah Estrin. Rumor Routing Algorthim for Sensor Networks. In *WSNA'02 : Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, pages 22–31. ACM Press, September 2002.

[88] Curt Schurgers and Mani B. Srivastava. Energy Efficient Routing In Wireless Sensor Networks. In *MILCOM'01*, pages 357–361, October 2001.

[89] Maurice Chu, Horst Haussecker, and Feng Zhao. Scalable Information-Driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks. *The International Journal of High Performance Computing Applications*, 16(3) :293–313, 2002.

[90] Yong Yao and Johannes Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 31(3) :9–18, September 2002.

[91] Narayanan Sadagopan, Bhaskar Krishnamachari, and Ahmed Helmy. The ACQUIRE Mechanism for Efficient Querying in Sensor Networks. In *SNPA'03 : Proceedings of the IEEE International Workshop on Sensor Network Protocols and Applications, held in conjunction with ICC 2003*, May 2003.

[92] Chunkai Yin and Ali Orooji. Routing Protocols for Sensor Networks. In *ICWN'2006 : Proceedings of the 2006 International Conference on Wireless Networks*, pages 15–21. CSREA Press, June 2006.

[93] Stephen T. Hedetniemi, Sandra M. Hedetniemi, and Arthur L. Liestman. Asurvey of broadcasting and gossiping in communication networks. *Networks*, 18 :319–349, 1988.

[94] Rajkumar Vijayaraman. *Directed Riffusion Routing Protocol : Attacks and Counter Measures*. Oklahoma State University, 2005.

[95] Min Chen, Taekyoung Kwon, and Yanghee Choi. Energy-Efficient Differentiated Directed Diffusion (EDDD) in Wireless Sensor Networks. *Computer Communications*, 29(2) :231–245, 2001.

[96] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. In *MobiHoc'01 : Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing*, pages 251–254. ACM Press, 2001.

[97] Anna L. Buczak and Vikram R. Jamalabad. Self-Organization of a Heterogeneous Sensor Network by Genetic Algorithms. In C. H. Dagli, M. Akay, A. L. Buczak, O. Ersoy, and B. R. Fernández, editors, *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 8, pages 259–264, New York, 1998. ASME Press.

[98] Chunhung Richard Lin and Mario Gerla. Adaptive Clustering for Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 15(7) :1265–1275, 1997.

[99] Hasnaa Moustafa and Houda Labiod. Multicast Routing in Mobile Ad Hoc Networks. *Telecommunication Systems*, 25(1-2) :65–88, 2004.

[100] Hasnaa Moustafa, Gilles Bourdon, and Yvon Gourhant. AAA in Vehicular Communication on Highways with Ad Hoc Networking Support : A Proposed Architecture. In *VANET'05 : Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*, pages 79–80. ACM Press, 2005.

[101] Hasnaa Moustafa, Gilles Bourdon, and Yvon Gourhant. Providing Authentication and Access Control in Vehicular Network Environment. In *SEC'2006 : Proceedings of the IFIP TC-11 21st International Information Security Conference*, pages 62–73, May 2006.

[102] Hassnaa MOUSTAFA. *Routage Unicast et Multicast dans les réseaux mobiles Ad hoc*. PhD thesis, ENST, Décembre 2004.

[103] Online. Réseaux de capteurs et réseaux auto-organisés (RECAP) - Plateforme Nationale CNRS. http ://www2.lifl.fr/sensor/Main/HomePage.

[104] Online. NS-2 : Network Simulator 2. http ://www.isi.edu/nsnam/ns/.

[105] Online. GloMoSim : Global Mobile Information Systems Simulation Library. http ://pcl.cs.ucla.edu/projects/glomosim/.

[106] András Varga. OMNeT++ Discrete Event Simulation System, Version 3.3. 2006. http ://www.omnetpp.org.

[107] Sung Park, Andreas Savvides, and Mani B. Srivastava. SensorSim : A Simulation Framework for Sensor Networks. In *MSWIM'00 : Proceedings of the*

3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pages 104–111. ACM Press, 2000.

[108] Rimon Barr, Zygmunt J. Haas, and Robbert van Renesse. JiST : an efficient approach to simulation using virtual machines. *Software, Practice and Experience*, 35(6) :539–576, 2005.

[109] Witold Drytkiewicz, Steffen Sroka, Vlado Handziski, Andreas Koepke, and Holger Karl. A Mobility Framework for OMNeT++. In *3rd International OMNeT++ Workshop, at Budapest University of Technology and Economics*, January 2003.

[110] C. Mallanda, S. Else, A. Suri, V. Kunchkarra, S.S. Iyengar, R. Kannan, and A. Durresi. Simulating Wireless Sensor Networks with OMNeT++. *IEEE Computer*, 2005.

[111] Alexander Kröller, Dennis Pfisterer, Carsten Buschmann, Sandor P. Fekete, and Stephan Fischer. Shawn : A New Approach to Simulating Wireless Sensor Networks. In *Design, Analysis, and Simulation of Distributed Systems 2005, Part of SpringSim 2005*, April 2005.

[112] Raphaël Couturier. *HDR : Algorithmes Itératifs Asynchrones sur Grappes Distantes et Équilibrage de Charge sur Topologies Dynamiques.* Université de Franche-Comté, 2005.

[113] George Cybenko. Dynamic Load Balancing for Distributed Memory Multiprocessors. *Journal of Parallel Distrib. Comput.*, 7(2) :279–301, 1989.

[114] Jacques E. Boillat. Load Balancing and Poisson Equation in a Graph. *Concurrency - Practice and Experience*, 2(4) :289–314, 1990.

[115] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Academic Press, New York, 1979, Reprinted by SIAM, 1994.

[116] Cintia B. Margi, Katia Obraczka, and Roberto Manduchi. Characterizing System Level Energy Consumption in Mobile Computing Platforms. In *Proceedings of IEEE WirelessCom'05.* IEEE Computer Society, June 2005.

[117] Juan-Carlos Cano and Pietro Manzoni. Evaluating the Energy-Consumption Reduction in a MANET by Dynamically Switching-Off Network Interfaces. In *ISCC'01 : Proceedings of the Sixth IEEE Symposium on Computers and Communications*, pages 186–192. IEEE Computer Society, July 2001.

[118] Samir Datta and Subir Biswas. Energy Savings by Intelligent Interface Idling in 802.11 Based Wireless Networks. In *Electro/Information Technology Conference*, 2005.

[119] Marcelo M. Carvalho, Cintia B. Margi, Katia Obraczka, and J. J. Garcia-Luna-Aceves. Modeling Energy Consumption in Single-Hop IEEE 802.11 Ad Hoc Networks. In *ICCCN'2004 : Proceedings of the International Conference On Computer Communications and Networks*, pages 367–372. IEEE Computer Society, October 2004.

[120] Juan-Carlos Cano and Pietro Manzoni. A Performance Comparison of Energy Consumption for Mobile Ad Hoc Network Routing Protocols. In *MASCOTS'00 : Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 57–64. IEEE Computer Society, 2000.

[121] Laura Marie Feeney. An Energy Consumption Model for Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks. *MONET'2001, MObile NETwork Applications*, 6(3) :239–249, 2001.

[122] Laura Marie Feeney and Martin Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *INFOCOM'2001 : Proceedings of the 20st International Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1548–1557, April 2001.

[123] Cintia B. Margi and Katia Obraczka. An Energy Model for Evaluation of Protocols for Power-Constrained Networks. Technical report, University of California Santa Cruz, 2004.

[124] Manish Bhardwaj and Anantha Chandrakasan. Bounding the Lifetime of Sensor Networks Via Optimal Role Assignments. In *INFOCOM'2002 : Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1587–1596. IEEE Computer Society, June 2002.

[125] András Varga. The OMNeT++ Discrete Event Simulation System. In *ESM'2001 : Proceedings of the European Simulation Multiconference*, volume 7, pages 309–324, June 2001.

[126] Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John W. Byers. BRITE : An Approach to Universal Topology Generation. In *MASCOTS'01 : Proceedings of the Ninth IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*. IEEE Computer Society, August 2001.

[127] Pierpaolo Bergamo, Alessandra Giovanardi, Andrea Travasoni, Daniela Maniezzo, Gianluca Mazzini, and Michele Zorzi. Distributed Power Control for Energy Efficient Routing in Ad Hoc Networks. *Wireless Networks*, 10(1) :29–42, 2004.

[128] C. Berge. *Graphes*. Gauthier-Villars, Paris, 1983.

Résumé

Au cours de ces dernières années a émergé un nouveau type de réseaux sans fil, à savoir les réseaux de capteurs. Une problématique majeure dans ce type de réseaux est la maîtrise de l'énergie consommée par chaque nœud capteur, puisque chaque nœud est alimenté par une batterie. Des mécanismes doivent donc être développés afin d'utiliser à bon escient l'énergie et maintenir le réseau en activité aussi longtemps que possible. Dans cette thèse, nous proposons un algorithme itératif décentralisé pour augmenter la durée de vie d'un réseau de capteurs. L'approche proposée consiste à faire participer chaque nœud proportionnellement à son énergie. Pour cela, à chaque nœud est associé un ratio d'énergies, ratio qu'il s'agira d'équilibrer à travers le réseau en faisant migrer des tâches entre nœuds voisins. L'algorithme a été validé par des simulations en OMNeT++, qui mettent en évidence la pertinence de l'approche proposée.

Mots clefs : réseaux de capteurs sans fil, durée de vie, équilibrage, algorithme itératif décentralisé.

Abstract

Recent advances in embedded computing systems have led to the emergence of wireless sensor networks, consisting of small, battery-powered nodes. In these networks, controlling the energy consumptions is an important challenge, since energy is the scarcest resource of wireless sensor networks, and it determines their lifetimes. Algorithms and protocols must be developed in order to use energy efficiently and to maintain the network in activity as a long as possible. In this thesis, we propose a decentralized iterative approach to improve the lifetime of a wireless sensor network. The proposed approach guarantees a fair workload distribution across the network. This approach ensures that each node will contribute proportionally to its available energy. We study the relevance of our algorithm to prolong the lifetime in heterogeneous wireless sensor networks through simulations using OMNeT++.

Key words : wireless sensor networks, lifetime, diffusion algorithm, decentralized iterative algorithm.