UNIVERSITÉ DES ANTILLES ET DE LA GUYANE

U.F.R. des SCIENCES

École doctorale pluridisciplinaire :
Santé, Environnement et Société dans les Amérique

THÈSE

présentée et soutenue publiquement par

Cédric RAMASSAMY

le 23 Novembre 2012
au Département Scientifique Interfacultaire
Campus de Schoelcher

en vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DES ANTILLES ET DE LA GUYANE

Spécialité : Informatique

Analyse des protocoles des Réseaux de capteurs sans-fil

JURY

M. Hacène FOUCHAL	Professeur à l'Université de Reims	Directeur de thèse
Mme Martine COLLARD	Professeur à l'UAG	Co-Directrice
M. Philippe HUNEL	Maître de Conférence à l'UAG	Encadrant
M. Azzedine BOUKERCHE	Professeur à l'Université d'Ottawa	Rapporteur
M. Jalel BEN-OTHMAN	Professeur à l'Université de Paris 13	Rapporteur

Remerciements

En préambule de cette thèse, je remercie tout d'abord tous ceux qui m'ont aidé tout au long de mon cursus universitaire et même avant, c'est-à-dire ma famille. Je dis donc un grand merci pour leur soutien, leur encouragement, d'avoir toujours cru en moi et surtout ma mère que j'aime et j'adore pour avoir été le pilier de ma réussite.

Bien entendu, sans mes encadrants rien de tout cela n'aurait été possible. Tous les doctorants n'ont pas un directeur de thèse disponible et motivé par leurs travaux. J'adresse donc mes très sincères remerciements à Hacène FOUCHAL mon directeur de thèse pour sa gentillesse, sa sympathie, sa disponibilité, ses idées, ses conseils et son caractère qui par sa force m'a encouragé et m'a permis de mener à bien cette thèse. Je remercie également ma codirectrice Martine COLLARD qui a fait l'honneur de s'intéresser à mon travail de thèse par ses idées. Je remercie aussi mon encadrant, Philippe HUNEL de m'avoir encadré durant toute la durée de la thèse, tes conseils, ton enthousiasme et ton recul m'ont été très bénéfiques pour mes travaux.

Ce travail a été réalisé au sein du laboratoire de Mathématiques Informatique et Applications (LAMIA). Je tiens donc à remercier les responsables du laboratoire de m'avoir accueilli et donné l'opportunité de réaliser ce travail de thèse.

Merci à Nicolas VIDOT pour tes conseils, tes discussions diverses et variées et tes analyses justes, qui nous a toujours proposé des perspectives de recherches. Je n'oublie pas mes collègues de bureau, Erick STATTNER et Nathalie DESSART pour la bonne ambiance qui y régnait, les cafés et repas partagés, pour ces années que nous avons surmontées ensemble, pour les discussions et les encouragements dans la vie personnelle et professionnelle, pour votre sympathie durant le temps où on a travaillé ensemble et pour les discussions de recherche dans le domaine des réseaux de capteurs, merci à vous.

La réalisation de ce travail n'aurait pas été possible sans des moments de détente (et réciproquement), donc merci aux amis Gaël, Manuel, Michaël... pour votre soutien indéfectible dans les moments difficiles ainsi que dans les moments mémorables qu'on a passé ensemble.

A tous même ceux qui ne sont pas tous, qui ont participé de près ou de loin à l'élaboration de cette thèse qui n'aurait peut-être pas vu le jour, un grand merci.

Enfin et surtout, merci à ma femme Mickaelle pour tout et le reste.

Table des matières

Chapitre 1

Introduction

Sommaire

1.1 Le développement des réseaux de capteurs

Durant ces dernières années, nous avons été marqués par un développement très rapide des techniques et technologies dans les domaines de l'électronique, la mécanique et les technologies de communication sans fil. Ces innovations ont permis de créer de petits objets communicants équipés de capteurs à un coût raisonnable. De nos jours, l'essor des réseaux de capteurs sans fil (RdC), représente une bonne thématique de recherche. Ces réseaux de capteurs sont composés d'un grand nombre de noeuds ou capteurs communiquant entre eux et déployés sur une zone donnée afin de mesurer une donnée tels que la température, l'humidité, la luminosité, etc. Ceci dans la plupart des cas, dans le but de surveiller un évènement [11]. Dans ce réseau un noeud est équipé d'une unité de mesure, une unité de calcul, une mémoire, une radio et une alimentation. La Figure 1.1 montre l'architecture générale d'un noeud et la Figure 1.2 montre un exemple de capteur.

Grâce à cette technologie, le déploiement des réseaux de capteurs sans fil est possible dans de nombreux domaines applicatifs tels que : militaires, domotique, surveillance industrielle ou de phénomènes naturels, habitat... D'ici un avenir proche selon certains scientifiques les réseaux de capteurs domineront le marché des logiciels [11]. Dans tous les domaines, le rôle d'un réseau de capteurs est quasiment toujours le même comme le montre la Figure 1.3.

Le principe est que les noeuds doivent surveiller un évènement en récupérant des données grâce à leurs capteurs, puis ils envoient les informations à une station de base (puits). Cette station de base (ou sink en anglais) est un noeud particulier doté d'une puissance de calcul supérieure et d'une alimentation quasi illimitée. Il

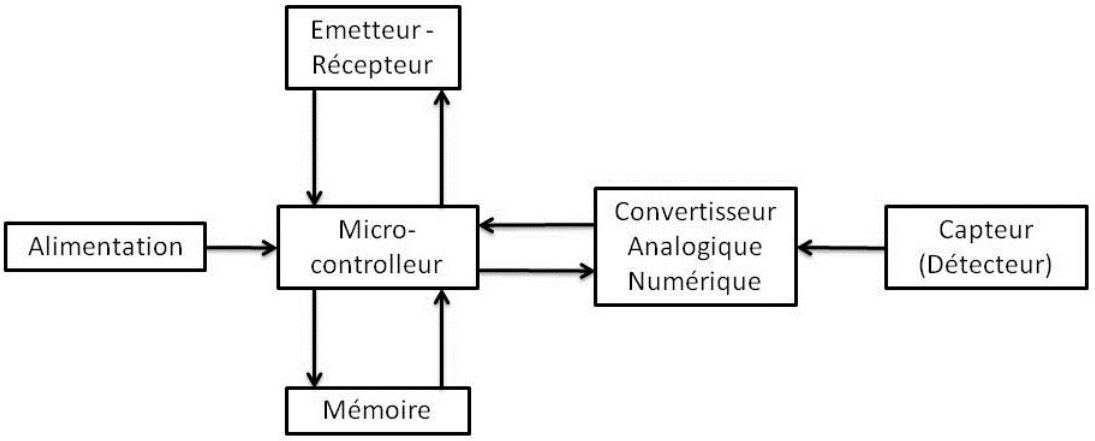


FIGURE 1.1 – Schéma de fonctionnement d'un capteur

FIGURE 1.2 – Exemple d'un capteur Micaz

est en général soit connecté à Internet ou par un lien radio de type GSM ou GPRS qui lui permet d'envoyer les informations à un serveur pour l'utilisateur final. Il est néanmoins possible d'avoir plusieurs stations de base fixes ou mobiles dans le réseau qui permettent le relai de l'information, mais pour des raisons de coût, le réseau comporte généralement beaucoup plus de noeuds que de stations de base.

Cependant, suivant le type d'application et le type de déploiement, il existe plusieurs catégories de capteurs différents qui varient en fonction des différentes caractéristiques tels que la taille du capteur, la capacité de calcul, la taille de la mémoire et la capacité de transmission. Suivant ces caractéristiques, les protocoles de communications, la taille du réseau, le réseau de capteurs est structuré selon quatre types de plateformes [52] :

- ➢ Une plateforme de capteurs miniaturisés qui est dédiée aux plus petits capteurs (quelques mm^3) ayant une faible bande passante ($< 50Kbps$). L'un des exemples les plus connus est de cette plateforme est Spec [5], conçu par l'université de Berkeley. Possédant une taille très petite de l'ordre de $2mm \times 2.5mm$, Spec est un des plus petits capteurs au monde.
- ➢ Une plateforme de capteurs généraux qui est développée pour capter et router des informations du monde ambiant. On retrouve dans cette famille très

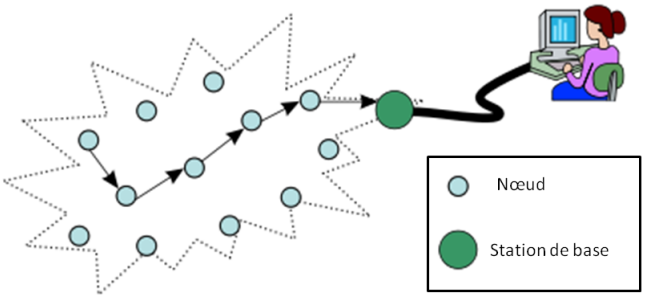


FIGURE 1.3 – Principe de fonctionnement

développée les MicaZ [4] ou encore les Sun Spot [7]. Ces capteurs possèdent une taille $10cm^3$ avec les protocoles de communications IEEE 802.15.4 [15]. De nos jours, MicaZ est toujours une référence dans les travaux de recherche dans le domaine des réseaux de capteurs.

- ➢ Une plateforme de capteurs à haute bande passante qui ont pour but de transporter de gros volumes de données captées tels que de la vidéo surveillance, du son, etc. On retrouve dans cette catégorie les Imote [2] dont la communication se base sur la norme Bluetooth 1.1 [6].
- ➢ Une plateforme de passerelles qui permet à ces dispositifs de transporter les informations envoyées par un réseau de capteurs vers un réseau traditionnel (Ethernet ou 802.11). On retrouve par exemple Stargate [4].

Parmi ces quatre plateformes, nous nous focaliserons plus particulièrement sur la plateforme des capteurs généraux. En effet ceux-ci reflètent de la tendance et popularité actuelle ainsi que la possibilité de programmation sur ce genre de capteurs. Ainsi, nous prendrons comme base de capteurs les MicaZ.

1.2 Difficultés et contraintes

Les réseaux de capteurs sans fil font partie des thèmes de recherches très actifs actuellement, car cette technique peut être appliquée dans de nombreux domaines : surveillance d'environnement [25], observation de la vie des espèces rares ou de personnes [47] et [95], surveillance de la structure des infrastructures [108], optimisation de diagnostique pour les patients [35], etc.

On trouve plusieurs méthodes pour classer les réseaux de capteurs [112]. Pour chaque domaine, les réseaux de capteurs ont des caractéristiques différentes. Ils se distinguent par le modèle de communication, le modèle de transmission de données, le modèle de mobilité dans le réseau, etc. Les acteurs industriels impliqués dans le domaine des réseaux de capteurs doivent être capables de développer rapidement des solutions fiables. Les entreprises qui conçoivent et déploient des réseaux de capteurs doivent le faire le plus rapidement possible pour faire face à la concurrence. L'enjeu économique lié à la conception des réseaux de capteurs est très important.

En dehors de l'enjeu économique lié à la conception, comme vu précédemment les capteurs sont de petits composants avec une faible capacité de stockage, de calcul et sont alimentés avec une batterie ou avec des piles. Pour qu'un réseau de capteurs reste autonome pendant une certaine durée de l'ordre de quelques mois à quelques années sans intervention humaine, la consommation d'énergie devient le problème fondamental. Pour les réseaux sans fil traditionnels, ceci n'est point un problème, car on peut toujours recharger les batteries des dispositifs sans fil comme les téléphones portables ou les ordinateurs portables. Mais dans un réseau de capteurs, il est très difficile de changer la batterie [25]. De même pour un réseau de capteurs déployé dans l'océan, on ne peut pas envoyer quelqu'un toutes les semaines juste pour changer la batterie.

Cependant, l'énergie n'est pas le seul problème parce que ce sont des systèmes complexes qui combinent des caractéristiques propres aux systèmes distribués et aux systèmes embarqués. Les systèmes distribués sont difficiles à concevoir pour plusieurs raisons. Les communications entre les noeuds ne sont pas fiables, surtout dans le cas de noeuds communiquant via une radio. On peut difficilement définir l'état global du système, et enfin l'exécution des processus (c.-à-d. les noeuds) est asynchrone. Quant aux systèmes embarqués, ils ont des ressources (calcul, mémoire) très contraignantes parce qu'ils doivent respecter des contraintes de coût. Et pour concevoir les systèmes embarqués, il faut prendre en compte les interactions fortes entre le logiciel et le matériel.

Ces caractéristiques particulières du réseau de capteurs modifient le critère de performance par rapport aux réseaux sans fil traditionnels. D'où les concepteurs de réseaux de capteurs sont confrontés à plusieurs choix. D'abord, il faut choisir les différents composants matériels qui constituent un noeud. Par exemple, le choix du microcontrôleur qui doit être suffisamment puissant pour subvenir aux besoins de l'application et consommer le moins possible. Le choix de la radio qui dépendra de la fréquence d'émission et de la portée souhaitée. Il faut également choisir ou concevoir les logiciels. Les protocoles de communications, le protocole d'accès au médium (MAC et PHY) ou celui de routage, qui influencent beaucoup la consommation. Le domaine de recherche lié à l'optimisation des réseaux de capteurs est très actif. Pour tous les protocoles, il y a également souvent des paramètres à définir et ces paramètres peuvent interagir. Tous ces choix dépendent bien sûr de l'application et naturellement de l'environnement physique dans lequel il est déployé.

1.3 Notre approche

Dans ce contexte, trouver une méthode pour atteindre la solution optimale en énergie est probablement hors de portée. Trouver des méthodes et outils d'aide à la conception des réseaux de capteurs semble plus accessible. La fiabilité des données et la consommation d'énergie correspondent aux points les plus importants que doivent prendre en compte ces méthodes.

1.4 Organisation du document

Le chapitre 2 présente les réseaux de capteurs de manière plus précise que cette introduction. Et contiens principalement l'état de l'art de la modélisation des réseaux de capteurs. Nous retrouvons une description des réseaux de capteurs ainsi que les outils (simulateurs) utilisés pour la modélisation de ces réseaux.

Le chapitre 3 décrit un état de l'art des différents protocoles usuels utilisés dans l'univers des réseaux de capteurs. On y retrouve les informations concernant la couche MAC et physique d'un capteur du standard 802.15.4 et les protocoles de routage dans les réseaux ad hoc.

Le chapitre 4 décrit une étude de l'impact de la couverture radio, de la topologie et de la taille du réseau sur les performances du réseau de capteurs. Nous étudierons dans ce chapitre l'impact de ces paramètres sur les performances du réseau de capteurs en étudiant les paquets perdus et l'énergie consommée.

Le chapitre 5 décrit l'approche pour l'implémentation d'un modèle de choix des bons paramètres de configuration pour une situation de fonctionnement d'un réseau de capteurs. Nous montrerons nos choix concernant les différents protocoles pour réaliser notre classification et notre outil permettant d'afficher les résultats de nos simulations.

Le chapitre 6 propose une approche pragmatique permettant de tester la conformité d'un réseau de capteurs dans un environnement réel. Pour cela nous décrierons notre architecture de test et notre prototype de cas d'étude.

Le chapitre 7 conclut cette thèse et présente les perspectives de recherche à ce travail.

Première partie

Contexte, Etat de l'art

Chapitre 2

Les Réseaux de capteurs

Sommaire

Nous présentons ici un état de l'art sur les enjeux liés à la conception des réseaux de capteurs. Pour bien comprendre les difficultés de modélisation auxquelles doivent faire face les concepteurs de tels réseaux, la section 2.1 donne quelques exemples de réseaux de capteurs. Elle commence par l'introduction de quelques applications puis explique les solutions protocolaires et matérielles que l'on trouve dans la littérature. Ensuite nous présenterons dans la section 2.2, différentes solutions existantes pour la modélisation des réseaux de capteurs. Etant donné la grande variété de simulateurs existants sur le marché, les références bibliographiques ne seront pas exhaustives.

2.1 Contexte des réseaux de capteurs

2.1.1 Applications

La miniaturisation, l'adaptabilité, le faible coût et la communication sans fil permettent aux réseaux de capteurs de s'installer dans plusieurs domaines d'application et permettent aussi d'étendre le domaine des applications existantes. Parmi ces domaines où ces réseaux se révèlent très utiles et peuvent offrir de meilleures contributions, on peut noter les domaines militaires, de la santé, de l'environnement,...

Applications militaires

Le faible coût, le déploiement rapide, l'auto-organisation et la tolérance aux pannes sont des caractéristiques qui ont rendu les réseaux de capteurs efficaces pour les applications militaires. Plusieurs projets ont été lancés pour aider les unités militaires dans un champ de bataille et protéger les villes contre des attaques, telles que les menaces terroristes. Le projet DSN (Distributed Sensor Network) [27] au DARPA (Defense Advanced Research Projects Agency) était l'un des premiers projets dans les années 80 ayant utilisé les réseaux de capteurs pour rassembler des données distribuées. Les chercheurs du laboratoire national Lawrence Livermore ont mis en place le réseau WATS (Wide Area Tracking System) [46]. Ce réseau est composé de détecteurs des rayons gamma et des neutrons pour détecter et dépister les dispositifs nucléaires. Il est capable d'effectuer la surveillance constante d'une zone d'intérêt. Il utilise des techniques d'agrégation de données pour les rapporter à un centre intelligent. Un réseau de capteurs peut être déployé dans un endroit stratégique ou hostile, afin de surveiller les mouvements des forces ennemies, ou analyser le terrain avant d'y envoyer des troupes (détection des armes chimiques, biologiques ou radiations). L'armée américaine a d'ailleurs réalisé des tests dans le désert de Californie.

Applications médicales

Dans le domaine de la médecine, les réseaux de capteurs peuvent être utilisés pour assurer une surveillance permanente des organes vitaux de l'être humain grâce à des micro-capteurs qui pourront être avalés ou implantés sur le patient (surveillance de la glycémie, détection de cancers, ...) [36]. Ils peuvent aussi faciliter le diagnostic de quelques maladies en effectuant des mesures physiologiques telles que : la tension artérielle, battements du coeur, température, ... à l'aide des capteurs ayant chacun une tâche bien particulière. Dans [34] on retrouve la collecte de données physiologiques pour la détection de maladie chez un patient. D'autre part, ces réseaux peuvent détecter des comportements anormaux (chute d'un lit, choc, cri, ...) chez les personnes dépendantes (handicapés ou âgées).

Applications environnementales

Les réseaux de capteurs se prêtent particulièrement bien à des applications de remontées d'alarmes où la zone à surveiller est difficile d'accès. Les réseaux de capteurs sont notamment pressentis pour aider à la surveillance et à la prévention des feux de forêt. Dans ce cas, le déploiement d'un réseau de capteurs permet d'alerter les secours en cas d'incendie, et d'évaluer le risque de départ de feu grâce à des relevés périodiques (température, humidité...). Le déploiement des capteurs chimiques dans les milieux urbains peut aider à détecter la pollution et analyser la qualité de l'air. Dans ce cas, leur déploiement dans les sites industriels permettrait une détection des risques industriels plus rapidement et réduirait très considérablement la fuite de produits toxiques (gaz, produits chimiques, éléments radioactifs, pétrole, etc.).

On retrouve aussi l'observation de la vie d'espèces rares [47], par exemple, en Martinique le moqueur gorge blanche représente une espèce en voie de disparition. L'utilisation d'un réseau de capteurs permet donc le comptage d'oiseaux de cette

espèce sur la péninsule de la Caravelle et en assurer la protection de l'habitat. Ce réseau peut aussi servir à l'étude de l'espèce d'un point de vue social (relation entre plusieurs individus de l'espèce, façon de fonctionner, etc.).

Domotique

Un autre type d'applications dans lequel les réseaux de capteurs émergent est la domotique. Dans ces applications, le réseau de capteurs est déployé dans l'habitation. Le principe est que le réseau de capteurs forme un environnement, dit environnement pervasif. Son but est de fournir toutes les informations nécessaires aux applications de confort, de sécurité et de maintenance dans l'habitat. Les capteurs sont des capteurs de présence, de son, ils peuvent même être équipés de caméras. Un tel réseau déployé doit permettre de créer une maison intelligente capable de comprendre des situations suivant le comportement des occupants et d'en déduire des actions. Dans ces types d'applications, les réseaux sont donc très hétérogènes, des éléments d'électroménager peuvent faire partie du réseau aussi bien que les ordinateurs personnels ou le réseau. Ces réseaux doivent être hautement reconfigurables : d'une part la topologie du réseau peut changer d'un jour à l'autre avec l'aménagement, d'autre part on peut avoir besoin de changer le type d'applications pendant la vie du réseau. La ZigBee Alliance, s'appuyant sur les couches 1 et 2 standardisées IEEE 802.15.4, répond assez bien aux contraintes de ces réseaux.

2.1.2 Couche Réseau

La couche réseau permet la communication de bout à bout entre deux noeuds, mais la réalisation de cette communication passe par différentes fonctions primordiales. Il y a le routage qui permet de déterminer un chemin entre deux noeuds, le relayage pour circuler l'information de voisin-en-voisin, le contrôle de flux pour éviter les problèmes de congestion. Ce qui fait de cette couche la seule à être directement concernés par la topologie du réseau. Pour cela, dans les réseaux de capteurs, la notion d'auto-organisation est une notion très importante.

L'auto-organisation est indispensable dans le cas de très grand réseaux où la gestion centralisée par un seul noeud n'est pas possible. Elle serait rendue trop complexe et pas assez dynamique à cause d'une surcharge trop importante de l'administrateur. De plus avec cette méthode, toute configuration manuelle ou toute administration devient inutile ce qui dans certaines applications est très appréciable (exemple de la surveillance des fonds marins par un réseau de capteurs largué par avion). Individuellement, chaque noeud possède une vue locale, peut communiquer avec ses voisins et est capable d'actions dites de bas niveau. Globalement, l'ensemble doit pouvoir résoudre une tâche complexe, en agissant de manière coordonnée tout en étant adaptable, robuste et de dimension ajustable. Un banc de poissons est un exemple de système auto-organisé. Dans [88], l'auteur décrit le système auto-organisé en utilisant un banc de poissons :

Adaptable : Il s'adapte à tout changement de situation, de façon à toujours être le plus efficace possible. Il doit donc être dynamique configurable.

Robuste : Il continue à fonctionner malgré la perte fréquente d'un de ses éléments.

Evolutif : Il est capable d'adapter sa taille et sa forme aux circonstances pour pouvoir continuer à fonctionner avec un nombre d'entités très élevé.

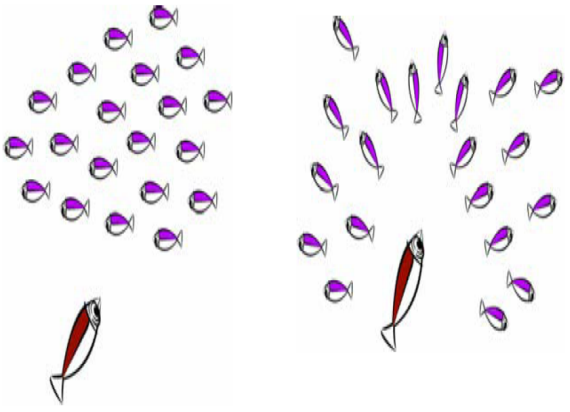
La figure 2.1 décrit le système autoorganisé du banc de poissons.

2.1.3 Couche MAC

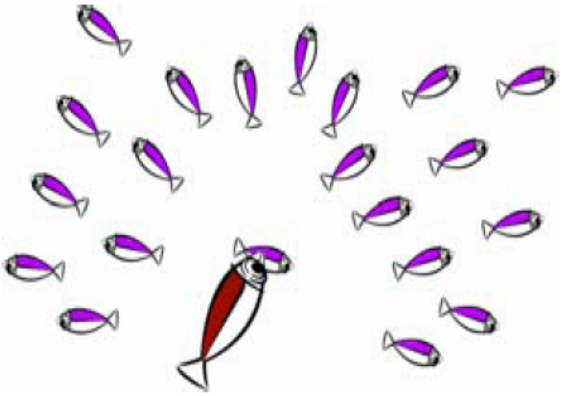
La couche MAC permet de définir les règles d'accès au canal [15]. Dans les réseaux de capteurs, la transmission radio est la source principale de consommation d'énergie et l'activité radio est en grande partie commandée par la couche MAC. Le protocole MAC permet, en ce qui concerne l'énergie, d'optimiser au mieux les pertes, qui sont caractérisées par les collisions, l'écoute pour la réception de données, les délais d'accès au réseau, l'overhearing c.-à-d. la réception de messages destinés à d'autres noeuds, le contrôle de l'overhead ou encore over-emitting c.-à-d. la transmission de messages quand le destinataire n'est pas prêt. Donc, pour économiser au mieux l'énergie de la batterie d'un capteur, il faudrait que les transmetteurs radio soient éteints le plus longtemps possible. Cependant, ceci pourrait poser le problème de la synchronisation des capteurs et la répartition des périodes de réveil. Ainsi, il faudrait que la couche MAC permette aux capteurs d'avoir des périodes de sommeil assez longues, mais sans perturber les communications. Le protocole MAC dédié aux réseaux de capteurs devrait être efficace en terme d'énergie, stable lorsque la taille du réseau augmente, et adaptatif aux changements de la topologie et de la connectivité du réseau lorsque les capteurs cessent de fonctionner, ou de se déplacer. Nous allons distinguer trois catégories de protocoles au niveau de la couche MAC, mais tout d'abord, nous expliquerons rapidement le mécanisme d'acquittement qui est utilisé dans de nombreux protocoles MAC.

Mécanisme d'acquittement

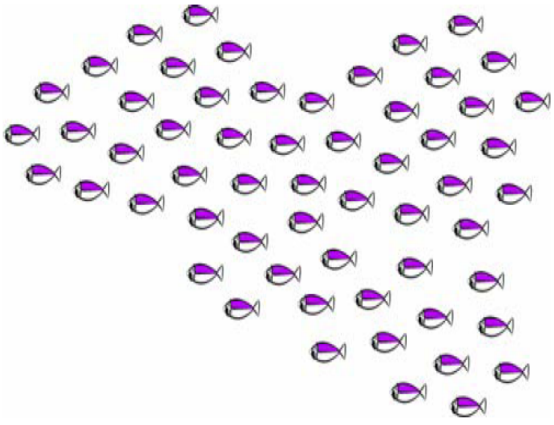
Le canal radio n'est pas fiable, des erreurs dues à des collisions ou à du bruit peuvent engendrer la perte d'un paquet émis. La figure 2.2 montre le problème du noeud caché introduit dans [106] où B ne peut pas entendre A, il peut donc émettre en même temps et ainsi provoquer une collision sur C. Pour éviter que trop de paquets soient perdus, les protocoles MAC implémentent souvent un mécanisme d'acquittement. Après avoir envoyé un paquet de données, le noeud récepteur attend un paquet d'acquittements (noté ACK pour *acknowledgment* que son destinataire doit envoyer à la réception d'un paquet de données. Le paquet ACK est un paquet de petite taille qui est émis juste après la réception du paquet de données. Comme le paquet de données a été reçu et que le paquet ACK est court, il a très peu de chances d'entrer en collision. Si l'émetteur ne reçoit pas l'acquittement, il considère donc que son paquet n'a pas été reçu. Il peut le cas échéant réémettre le paquet, après un certain délai par exemple. Ce mécanisme ne fonctionne plus lorsqu'un paquet est destiné à plusieurs récepteurs. En effet, dans ce cas, les multiples acquittements envoyés simultanément entreraient en collisions et le noeud émetteur du paquet ne pourrait en recevoir correctement aucun. Plusieurs mécanismes pour la réduction des collisions ont été proposés et on en retrouve quelques uns dans [66, 60, 29].



(a) Adaptable



(b) Robuste



(c) Evolutif

FIGURE 2.1 – Exemple de système autoorganisé : le banc de poissons

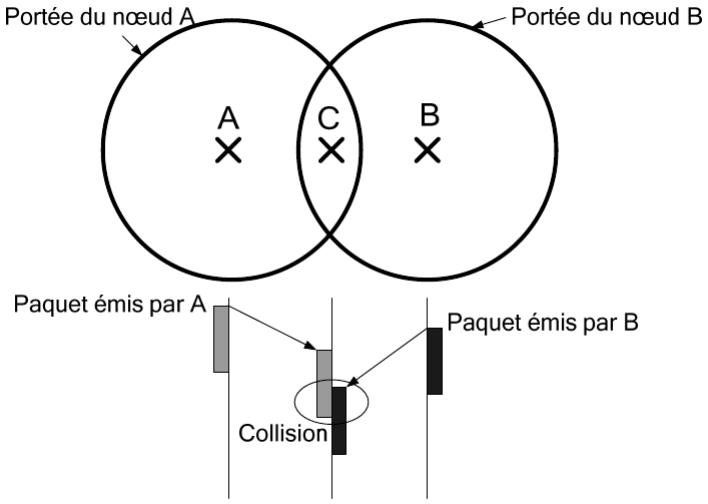


FIGURE 2.2 – Problème du noeud caché

Les protocoles ordonnancés

Ces protocoles sont efficaces en consommation d'énergie parce qu'ils évitent les collisions et l'overhearing. Ils ne permettent pas les communications pair-à-pair et exigent généralement aux noeuds de former des clusters. La communication inter-cluster est réalisée par les approches : TDMA (*Time Division Multiple Access*), FDMA (*Frequency Division Multiple Access*) et CDMA (*Code Division Multiple Access*) [61]. L'utilisation de FDMA pour les communications intra-cluster évite les collisions, mais elle augmente la consommation d'énergie des capteurs à cause du besoin de circuits additionnels pour communiquer dynamiquement avec les différents canaux radio. De même, CDMA évite les collisions, mais les calculs consomment suffisamment d'énergie [33], alors que TDMA essaie d'optimiser le procédé d'attribution des slots dans le but de minimiser la consommation de l'énergie. Cependant, dans ces approches il manque la scalabilité et l'adaptabilité aux changements de la topologie ainsi que l'ordonnancement d'activité des noeuds relais. En outre, ils dépendent d'une synchronisation distribuée et leur débit est limité à cause des slots d'écoute non utilisés. Pour remédier aux limitations présentées dans ces approches, d'autres techniques ont été proposées dans cette direction. On retrouve notamment le protocole SMACS [102] (Self-organizing Medium Access Control for Sensor Networks) qui permet d'organiser le réseau. SMACS implique une méthode d'accès au médium combinant TDMA et FDMA, dans laquelle les noeuds voisins choisissent aléatoirement un slot et une fréquence qui définit un lien. Ce protocole est très utile quand la transmission périodique de l'information exige la maintenance continue du réseau. Cependant, il exige un overhead important et la bande passante n'est pas bien exploitée. En outre, dans [14], Arisha et al. ont présenté un protocole MAC basé sur TDMA, qui se sert des passerelles comme cluster-heads pour assigner des slots aux capteurs dans un cluster. Néanmoins, le passage de l'état "ON/OFF" est très coûteux en consommation d'énergie.

Protocoles basés sur la contention

Dans ce type de protocoles MAC, les noeuds accèdent au médium durant le même intervalle de temps en utilisant un algorithme de la famille CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), chacune de ses variantes essayant d'éviter les collisions. Le point fort de ce type de protocoles est sans doute l'extensibilité et le passage à l'échelle. En revanche, ces protocoles n'offrent pas de délai borné du fait qu'ils ne garantissent pas l'accès au médium dès que la charge du réseau augmente. Tout d'abord nous allons décrire le modèle CSMA/CA de la norme 802.11 puis présenter quelques protocoles basés sur la contention.

CSMA/CA de la norme IEEE 802.11 CSMA/CA [9] est une méthode d'accès similaire à CSMA/CD [1] (*Carrier Sense Multiple Access with Collision Detection*), adaptée aux réseaux sans-fil, puisqu'elle impose à un émetteur de s'assurer que le canal est libre avant d'émettre. Dans CSMA/CA, les collisions ne peuvent pas être détectées comme dans le CSMA/CD, un noeud essaie d'éviter les collisions (sans vraiment les éviter à 100%). Ceci à cause de l'effet d'aveuglement du médium sans fil (*Near Far Effect*) qui empêche une entité de recevoir quand elle est en train d'émettre.

S-MAC S-MAC (Sensor MAC) [111] est un protocole similaire au protocole 802.11. Il utilise la méthode d'accès au médium CSMA/CA RTS/CTS (Request-To-Send, Clear-To-Send) qui permet d'éviter les collisions et le problème du noeud caché. Ce protocole est conçu pour assurer une méthode d'accès économe en énergie pour les réseaux de capteurs sans fil. Pour ce faire, les noeuds se mettent en mode sommeil pendant une certaine durée et se réveillent pour écouter le médium pendant une autre durée. Les noeuds échangent leur calendrier de périodes d'écoute en le diffusant à leurs voisins à un saut. Ainsi, chaque noeud connaît le calendrier de ses voisins et sait quand il faut se réveiller pour communiquer avec un noeud à sa portée. Plusieurs noeuds peuvent avoir le même intervalle de temps comme période d'écoute. Pour maintenir une synchronisation des horloges, les noeuds émetteurs envoient des messages de synchronisation SYNC au début de la période d'écoute de leurs voisins. S-MAC est particulièrement conçu pour les systèmes d'alerte dans lesquels les applications ont de longues périodes d'inactivité et peuvent tolérer la latence. Donc, son but est de maximiser la durée de vie du réseau en dépit des autres critères de performances.

T-MAC T-MAC [32] correspond à une amélioration des performances de S-MAC avec une charge de trafic variable. Il propose de mettre un noeud en mode sommeil après un temps TA durant lequel le noeud n'a reçu aucun message. Ainsi, T-MAC réduit l'*idle listening* par rapport à S-MAC. Avec cette approche, T-MAC fait dormir un noeud sans s'être assuré que ses voisins n'ont plus de données à lui envoyer. En effet, les données à envoyer du voisin ont pu être retardées à cause d'un échec d'accès au canal. Néanmoins dans T-MAC les performances se dégradent quand le trafic est unidirectionnel, par exemple quand les capteurs remontent seulement leurs données collectées à la station de base.

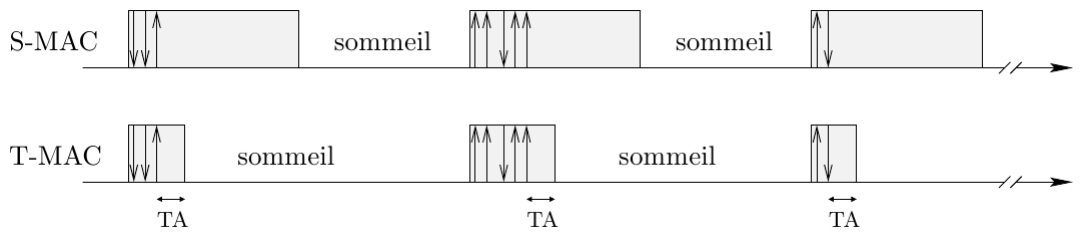


FIGURE 2.3 – Différence entre S-MAC et T-MAC

D-MAC D-MAC [68] (*Data gathering MAC*) propose un séquencement des périodes d'activité qui favorise la collecte d'informations dans une topologie arborescente. Les noeuds de même niveau se réveillent en même temps.

LPL : B-MAC, WiseMAC et X-MAC Le LPL [51] (*Low Power Listening*) est l'une des premières approches pour réduire l'*idle listening* en introduisant une période d'inactivité au niveau de la couche physique. L'idée est de pénaliser les émetteurs en envoyant un préambule long pour économiser l'énergie des récepteurs. Les récepteurs activent leur module radio périodiquement pour détecter la présence d'un préambule. La durée de transmission du préambule doit être de la même longueur que l'intervalle entre deux réveils d'un récepteur. L'un des premiers protocoles MAC pour les réseaux de capteurs sans fil basés sur cette technique est B-MAC [85] (*Berkeley-MAC*). WiseMAC [39] a été proposé pour réduire la longueur du préambule en fonction des périodes de réveil des récepteurs voisins. X-MAC [23] décompose lui aussi le long préambule en plusieurs préambules de petite taille incluant l'identifiant du destinataire du message. Par ailleurs, il a suivi plusieurs modifications apportées dans [104] où l'on obtient une réduction de la consommation d'énergie pour les noeuds routeurs. On observe une baisse importante de 90% de la consommation d'énergie.

Le standard LRWPAN (*Low Rate Wireless Personal Area Networks*) utilise deux méthodes d'accès au médium CSMA/CA et slotted CSMA/CA. CSMA/CA est similaire à celle utilisée par les protocoles 802.11, mais avec un backoff réduit, alors que slotted CSMA/CA (accès multiple durant un intervalle de temps) utilise des supertrames qui sont définies par le coordinateur, et qui ont pour objectif de permettre la synchronisation de l'ensemble des noeuds entre eux et économise la consommation d'énergie de chaque noeud. La synchronisation entre les noeuds est réalisée grâce à l'envoi d'un beacon (balises). Cet envoi est suivi par une zone appelée *Contention Access Period* (CAP), qui permet à chaque noeud d'envoyer ou recevoir des trames de commandes et/ou de données suivant le mécanisme CSMA/CA. Cependant, CSMA/CA induit une consommation d'énergie importante surtout durant les périodes de fort trafic, ce qui a conduit à réduire la valeur du backoff et par conséquent réduire la période de contention pendant laquelle le noeud émetteur devrait écouter le canal [15].

2.2 Modélisations pour les réseaux de capteurs

La méthode de modélisation des réseaux de capteurs consiste à créer un modèle du système que l'on veut développer. Une fois ce modèle créé, on obtient un prototype du système. Ce prototype virtuel est ensuite analysé et modifié jusqu'à ce qu'il respecte les spécifications du problème. On peut alors implémenter le prototype. Le prototypage virtuel permet de développer rapidement des systèmes complexes. Dans les méthodes de prototypage, on retrouve les simulateurs de réseaux. Nous proposons dans cette section un état de l'art des simulateurs de réseaux dans un premier temps, puis les simulateurs spécifiques des réseaux de capteurs puis les simulateurs prenant en compte une modélisation de l'environnement du réseau afin d'obtenir des simulations plus réalistes.

2.2.1 Simulateurs de réseaux généraux

Parmi les simulateurs de réseaux, on trouve les simulateurs de réseaux classiques, ces simulateurs ont été conçus pour modéliser et simuler des réseaux classiques tels que les réseaux filaires, les réseaux sans fil voire les réseaux ad hoc. Dans ces réseaux, le facteur d'énergie n'était pas encore le plus important. On retrouve un des simulateurs de réseaux les plus utilisés : NS-2 (*The Network Simulator*) [76] codé en C++. Le projet VINT correspond à l'origine de NS-2. Breslau et al. [21] estiment qu'il faut un seul simulateur de réseaux pour la communauté scientifique. Ce souhait donne naissance au projet VINT. L'intérêt d'un simulateur unique correspond surtout à la facilité à comparer différentes solutions. NS-2 est un simulateur à événement discret. NS-2 propose quatre niveaux d'abstraction ce qui permet d'adapter le simulateur aux différents intérêts. En effet, certains souhaiteront des informations bas-niveau, pour étudier par exemple l'effet de la propagation de l'onde radio dans un environnement alors que d'autres étudient les protocoles de routages et ne souhaitent que des informations au niveau réseau. NS-2 était d'abord destiné aux réseaux filaires ce qui explique certainement la simplicité des modèles de propagations radio de ce simulateur. Malgré les différents niveaux d'abstraction, NS-2 est essentiellement utilisé par les chercheurs qui s'intéressent aux protocoles de routage et/ ou aux protocoles d'accès au médium. Le code NS-2 étant ouvert, ceci permet à chacun d'ajouter sa pierre à l'édifice. Il existe donc une large bibliothèque décrivant toutes les informations dont on a besoin de la configuration physique d'un noeud jusqu'à l'application déployée sur le réseau. Ceci permet donc de mieux comparer les dernières avancées avec l'état de l'art. Dans le monde de la recherche NS-2 reste un simulateur très utilisé. L'outil Nam [40] permet de visualiser l'exécution d'une simulation. NS-2 génère des traces qui détaillent tous les événements de la simulation. Grâce à l'une de ces traces, Nam génère le "film" de la simulation. Cet outil permet surtout d'avoir une vue de l'exécution de la simulation beaucoup plus simple que la vue du fichier trace qui dans certains cas fait largement plus de 100 000 lignes.

Cependant, NS-2 n'est pas le simulateur parfait. Tout d'abord, l'utilisation n'est pas optimisée, il est nécessaire d'effectuer toutes ces tâches pour chaque simulation : modifier le fichier TCL, lancer la simulation, analyser les résultats, produire les graphiques. Ce qui est un point faible pour l'utilisateur ayant beaucoup de simulations

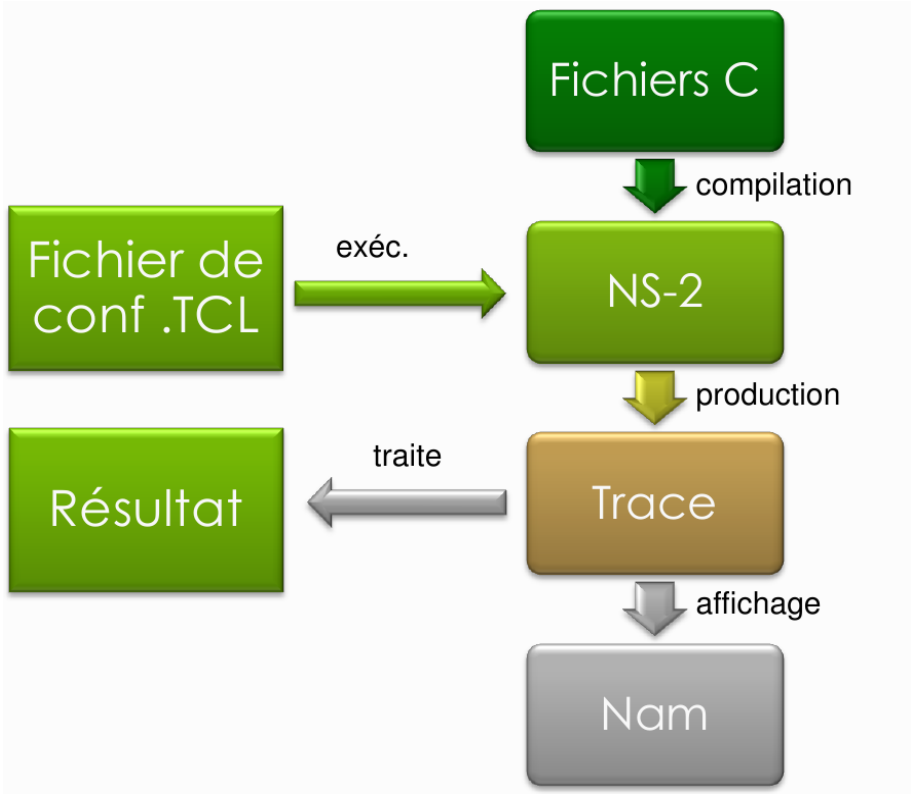


FIGURE 2.4 – Présentation de NS-2

à exécuter. D'autre part à cause des multiples contributions qui enrichissent NS-2, ce simulateur est devenu très complexe. En effet, les composants qui constituent un réseau, ou les noeuds d'un réseau ne sont pas toujours bien séparés dans NS-2. De ce fait pour l'implémentation de protocoles qui contiennent des optimisations inter-couches il est nécessaire de modifier des parties du code à différents niveaux, et dans NS-2 les couches n'étant pas clairement séparées, l'implémentation devient vite illisible. Enfin NS-2 ne propose pas de modèle d'environnement, mais plutôt des outils pour essayer de reproduire un environnement. Concernant la communication on peut choisir soit l'utilisation d'un flux de type TCP/IP, UDP, etc. ou bien considérer une loi de Poisson qui modélise bien l'émission de paquets au niveau d'un noeud. Pour les réseaux de capteurs, ces deux approches sont tout à fait discutables.

OMNET++ [79] est un simulateur à événements discrets implémenté en C++ comme NS-2. La mise en route avec ce simulateur est assez simple grâce à une conception claire du simulateur. Il fournit également une puissante bibliothèque d'interfaces graphiques pour l'animation et la gestion du débogage. Néanmoins il y a un manque cruel de protocoles disponibles dans la bibliothèque comparé à d'autres simulateurs tels que NS-2. Néanmoins, OMNET++ est devenu un outil populaire et son manque de protocoles est réduit grâce aux récentes contributions. En effet un framework pour la mobilité y a été ajouté [78], et il peut être utilisé dans le cadre de la modélisation de réseaux de capteurs.

Pour la simulation de réseaux de grandes tailles, l'utilisation de simulateurs distribués n'est pas négligeable. On retrouve dans ce cadre le simulateur GTNetS [94]

(*Georgia Tech Network Simulator*) qui permet de distribuer les calculs du simulateur sur plusieurs machines. GTNetS n'est pas non plus un simulateur dédié aux réseaux de capteurs.

2.2.2 Simulateurs dédiés aux réseaux de capteurs

Avrora [105] est un simulateur pour réseaux de capteurs. Pour obtenir une simulation fiable, ce simulateur est *cycle accurate* ce qui signifie qu'il simule pour chaque noeud toutes les instructions qui s'exécutent sur ce noeud. Certes, cette approche offre une modélisation particulièrement précise, mais on ne peut pas espérer qu'elle passe à l'échelle. En effet, Avrora est écrit en Java et chaque noeud est implémenté par un *thread* Java ce qui impose une difficulté supplémentaire : il faut synchroniser les threads entre eux pour s'assurer qu'un noeud ne reçoive pas un paquet avant que son voisin ne l'ait envoyé. Ceci complexifie le réseau à mesure que le nombre de noeuds augmente.

Atemu [86] est aussi un simulateur pour réseaux de capteurs très précis. Dans ce simulateur, les noeuds sont émulés, c'est-à-dire que l'on exécute le code binaire de chaque noeud. Par contre les transmissions sans fil sont simulées. Du fait de la précision de ce simulateur, la simulation avec une centaine de noeuds devient très compliquée à réaliser par la quantité d'opérations à faire quand le nombre de noeuds augmente.

TOSSIM [65] est le simulateur dédié à TinyOS. TinyOS [64, 8] est un système d'exploitation dédié aux réseaux de capteurs. TOSSIM essaye de tirer parti du mode d'exécution de TinyOS pour proposer un simulateur efficace et fiable. Le mode d'exécution de TinyOS est dirigé par les événements, ce mode d'exécution se calque bien sur un simulateur à événements discrets. TOSSIM contient un modèle abstrait de chaque composant du matériel d'un noeud. Pour une simulation TOSSIM, on utilise le même code que celui destiné au noeud cible, mais cette fois-ci TOSSIM émule le comportement du matériel en utilisant les modèles des composants. Simuler exactement le code qui tournera sur les noeuds permet de tester l'implémentation finale des algorithmes. Cette approche privilégie le code et fait abstraction du matériel, cependant dans sa première version TOSSIM ne permettait pas d'estimer la consommation énergétique.

PowerTOSSIM [100] est l'extension de TOSSIM qui contient un modèle de consommation d'énergie. Pour les valeurs de consommation, les auteurs se sont basés sur les noeuds de type Mica2 (cousin du MicaZ [4]. Les auteurs connaissent les consommations des différents composants de ce noeud suivant leurs états. Il faut donc connaître l'état de chaque composant d'un noeud pendant la simulation. Grâce au modèle de simulation basé sur TinyOS, on connaît immédiatement l'état des composants autres que le microcontrôleur puisque les changements d'état correspondent à des événements dans TinyOS et donc dans TOSSIM. Néanmoins plusieurs composants du noeud sont parfois abstraits dans TOSSIM par un seul composant. L'estimation de la consommation du microcontrôleur est plus délicate. En effet il faut analyser le code pour être capable de compter le nombre d'exécutions de chaque bloc d'instruction, et il faut faire correspondre chaque bloc d'instructions avec son code en assembleur. De ce fait pendant la simulation, on note le nombre de passages, d'exécution,

de chaque bloc d'instructions. Sachant combien d'instructions élémentaires contenues chaque bloc de base, on en déduit le nombre d'instructions effectuées par le microcontrôleur. Et ceci nous permet d'obtenir sa consommation énergétique. L'inconvénient majeur de PowerTOSSIM et TOSSIM est que les applications doivent obligatoirement être écrites en NesC [44].

Les auteurs de Worldsens [41] propose un autre concept qui consiste à relier deux simulateurs WSNet et WSim. WSNet est un simulateur à événements discrets à base de composants comme le fait le simulateur NS-2. Plusieurs modèles de radio sont disponibles dont certains relativement précis. WSim quant à lui est un simulateur bas niveau, dédié au matériel. Chacun d'eux peut être utilisé seul, mais Worldsens permet une utilisation conjointe. Le but est de choisir dans un premier temps le côté applicatif avec les logiciels dans WSNet puis le choix du matériel dans WSim ensuite simuler les deux ensemble (l'applicatif sur le matériel). Les communications radio sont gérées par WSNet et la synchronisation s'effectue sous forme de rendez-vous. Néanmoins WSNet permet la gestion d'un nombre important de noeuds, cependant WSim quant à lui gère très mal un nombre de noeud important du fait de la précision du simulateur, d'où il devient difficile de simuler une centaine de noeuds. D'autres simulateurs propres aux réseaux de capteurs peuvent être trouvés dans [38].

2.2.3 Simulateurs prenant en compte un modèle d'environnement

Ces simulateurs dédiés aux réseaux de capteurs permettent d'estimer et d'étudier assez finement les réseaux de capteurs comme on le souhaite. Mais l'environnement reste un facteur permettant d'obtenir des données plus réalistes. En effet, dans un réseau de capteurs, le trafic dépend des capteurs et donc de l'environnement. On se retrouve donc avec les simulateurs présentés précédemment, des incertitudes concernant la circulation du trafic lié à l'environnement. Nous allons décrire quelques simulateurs prenant en compte un modèle d'environnement.

Dans [103] les auteurs proposent de connecter le simulateur d'environnements Matlab [3] au simulateur de réseaux de capteurs TOSSIM. Leurs tests ont été réalisés sur la simulation d'un réseau de capteurs dédié au contrôle de la structure d'un bâtiment. Pour cette application, Matlab fournit un bon modèle de l'environnement ; cependant Matlab convient beaucoup moins à la simulation d'environnements qui ne suivent pas des équations différentielles.

Outre ce modèle d'interconnexions, les simulateurs avec modèle d'environnement font surtout une analogie entre la propagation des ondes radio et la propagation du phénomène à observer.

SensorSim [80] est un simulateur à événements discrets pour réseaux de capteurs. Pour simuler l'environnement, les noeuds de ce simulateur contiennent en plus du module radio un module *capteur* qui dispose d'une couche protocolaire (*"Sensor protocol stack"*) qui reçoit des messages venant d'un canal capteur (*"Sensor channel"*). La différence principale entre ce module *capteur* et le module radio est que les noeuds ne peuvent que recevoir sur ce canal, ils ne peuvent pas émettre. Pour effectuer une simulation, il faut déterminer les caractéristiques de propagation du phénomène à observer sur le canal *capteur*.

J-Sim [101] s'inspire de SensorSim pour la modélisation de l'environnement. J-Sim contient donc un canal *capteur*. Le phénomène à capter est créé par un noeud particulier appelé *"Target node"*, ce noeud envoie périodiquement des stimulus qui se propagent sur le canal *"capteur"*. Deux modèles de propagation sont implémentés : un modèle de propagation sismique et un modèle de propagation acoustique.

Dans [37] on retrouve également une extension de NS-2 pour simuler les réseaux de capteurs. Ici aussi, l'auteur fait une analogie entre le canal *capteur* et le canal radio, mais il approfondit encore plus dans cette analogie. En effet, un canal radio est créé pour chaque phénomène. Il y a deux types de noeuds. En plus des noeuds classiques, qui communiquent sur le canal radio et reçoivent des informations des nanaux *capteurs*, L'auteur crée des noeuds PHENOM. Les noeuds PHENOM ne "communiquent" que sur un canal *capteur*. Ces noeuds émettent régulièrement un paquet indiquant leur présence. Les noeuds PHENOM disposent également d'une couche MAC et d'une couche routage. Pour éviter des collisions entre phénomènes qui ne seraient pas réalistes, la couche MAC utilisée est parfaite. Le routage quant à lui détermine quand et avec quelle fréquence les noeuds PHENOM envoient les messages indiquant leur présence. Cependant, l'utilisation de la propagation des ondes radio pour simuler la propagation de phénomènes quelconques ne parait pas si judicieuse que ça, mais quand même astucieuse. De plus, les calculs engendrés par la propagation de l'onde sont très coûteux pour le simulateur et deviennent donc très lourds. Car ceci revient à rajouter d'autres noeuds dans le simulateur qui effectue un certain nombre de calculs pour juste rajoute la contrainte environnementale dans la simulation.

Chapitre 3

Les Protocoles utilisés dans les WSN

Sommaire

Dans ce chapitre nous présentons un état de l'art des protocoles qui sont utilisés dans les réseaux de capteurs. Plus particulièrement ceux que nous utiliserons dans nos prochains chapitres. Nous présenterons tout d'abord dans la section 3.1, le standard IEEE 802.15.4 qui définit les propriétés de la couche physique et d'accès au médium (MAC) d'un capteur. Nous détaillerons les spécifications physiques et les méthodes d'accès au médium. Ensuite nous présenterons dans la section 3.2 un état de l'art des protocoles de routage utilisés dans les réseaux de capteurs, et ceux qui ont été réalisés pendant l'étude. Etant donné le nombre de protocoles de routage existant, nous limiterons le nombre de références bibliographiques.

3.1 Le standard IEEE 802.15.4

Face aux contraintes des réseaux de capteurs qui nous avons déjà cités, notamment la consommation qui montre clairement que les standards sans fil classiques tels que Bluetooth [6] qui n'est pas adapté pour être utilisé dans ce type de réseaux. Le premier standard à avoir été normalisé et prenant en compte ces nouvelles contraintes liées aux réseaux de capteurs est le standard ZigBee. ZigBee est une technologie sans fil à faible consommation énergétique (*LR-WPAN*), gérée par la ZigBee Alliance, relativement récente fondée sur la base de l'IEEE 802.15.4, afin de pallier au problème de trop forte consommation d'énergie comme le Bluetooth.

Un amalgame est souvent réalisé entre le standard ZigBee [15] et le standard IEEE 802.15.4 [70], qui s'appliquent pourtant à des niveaux différents. La distinction est illustrée par la figure 3.1 qui montre que le standard ZigBee est une Alliance qui bénéficie du standard IEEE 802.15.4 pour définir la couche physique et MAC. Le standard ZigBee rajoute une couche réseau en plus.

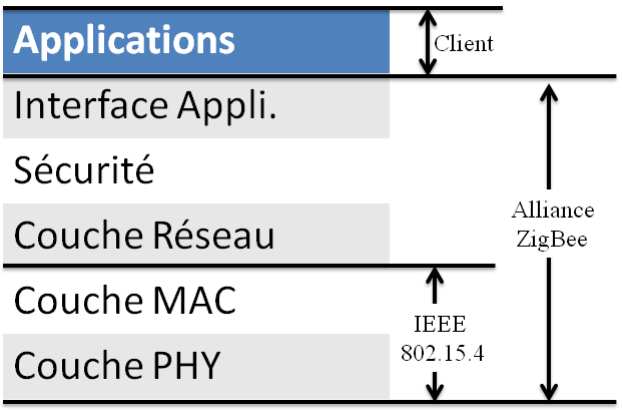


FIGURE 3.1 – Différence entre ZigBee et 802.15.4

Historiquement, concernant le ZigBee et le standard 802.15.4, la première réflexion date de 1999 et la première proposition de 2000. La version finale de l'IEEE 802.15.4 a été publiée en Octobre 2003, puis complétée en 2006 [70]. De son côté le standard ZigBee a créé une alliance en 2002.

Caractéristiques	Wifi	Bluetooth	ZigBee
IEEE	802.11b	802.15.1	802.15.4
Complexité	Très Grande	Grande	Faible
Durée de vie	Heures	Jours	Années
Nombre de noeuds	32	7	65000
Débit	11Mbits/s	1Mbits/s	20 - 250Kbits/s
Portée	100m	10m	10 - 300m

FIGURE 3.2 – Comparaisons avec d'autres standards

La figure 3.2 décrit les principales caractéristiques du ZigBee comparé au Wifi et Bluetooth. Nous allons maintenant détaillé les caractéristiques de la couche PHY et MAC.

3.1.1 Description et spécification de la couche Physique (PHY)

Le standard IEEE 802.15.4 spécifie la couche physique (PHY) dans les bandes ISM 868MHz, 915MHz et 2,4GHz avec un débit respectif de 20, 50 et 250Kbits/s. Avec ces trois plages de fréquences, la couche physique offre 27 canaux de transmission différente, dont 16, sur la plage de fréquences de 2400/2483,5MHz séparés de 5MHz, on a également 10 canaux sur la plage de fréquences de 902/928MHZ séparés de 2MHZ et 1 canal sur la plage de fréquence de 868/868,6MHz. L'usage de ces

	2450 MHz	**915 MHz**	**868 MHz**
Débit	250Kb/s	40Kb/s	20Kb/s
Nb. de canaux	16	10	1
Modulation	O-QPSK	BPSK	BPSK
Code d'étalement	32	15	15
Bit/symbole	4	1	1
Période d'un symbole	16μs	24μs	49μs

TABLE 3.1 – Sépcification de la couche physique

canaux dépend de la législation des pays dans lesquels des solutions conformes à ce standard sont utilisées.

Un codage de bits par étalement de spectre à séquence directe (*Direct Sequence Spread Spectrum*, DSSS) est utilisé avant la modulation sur fréquence porteuse [98, 114]. Cette technique d'étalement permet de faire fonctionner la transmission avec un rapport signal à bruit (SNR) plus important que ceux des systèmes à bande étroite, au prix d'une bande passante plus importante. D'autre part concernant la modulation la bande des 2,4GHz emploie le mode O-QPSK (*Offset Quadrature Phase Shift Keying*) tandis que les bandes de 868 et 915MHz emploient la modulation BPSK (*Binary Phase Shift Keying*) [15]. Le tableau 3.1 résume les spécifications de la couche physique.

Parmi les fonctionnalités de contrôle de cette couche, nous pouvons disposer de celles qui permettent de [15] :

- ➢ Activer et désactiver le module radio,
- ➢ Remonter l'état d'un lien à la couche supérieure,
- ➢ Tester l'occupation du canal en faisant un CCA (*Clear Channel Assessment*),
- ➢ Choisir le canal de transmission.

Activation et désactivation du module radio Le module radio possède trois états de fonctionnement : un état de transmission, un état de réception et un état de sommeil. Le temps de changement d'état entre transmission et réception ne doit pas dépasser les 192μs. Cet état est piloté par la couche MAC. Il est essentiel pour économiser de l'énergie de mettre le module en mode sommeil.

Indication de la qualité du lien Le LQI (Link Quality Indication) caractérise la qualité d'un lien à un instant donné suite à une réception d'une trame. Ce paramètre est essentiel pour les protocoles des couches réseau et application. Par exemple, les protocoles de routage peuvent utiliser cette indication pour le choix de la route.

Test d'occupation du médium ou CCA Le CCA (*Clear Channel Assessment*) permet de savoir l'état du canal radio. Il est essentiel pour le fonctionnement de l'algorithme de CSMA/CA de la couche MAC. La couche physique est capable d'effectuer trois modes de CCA différents :

1. La détection d'un signal avec une puissance reçue supérieure à un certain seuil.
2. La détection d'un signal conforme à la modulation de la couche physique.
3. La détection d'un signal qui répond aux deux conditions.

Remarque : le seuil de détection d'activité est de -95dBm.

Sélection du canal Comme la couche physique offre plusieurs canaux de transmission, il est nécessaire de sélectionner un canal précis, à la demande des couches supérieures. Le changement de canal est aussi fait implicitement par la couche physique suite à une demande de scrutation sur l'ensemble des plages de fréquences chacune constituant un canal de transmission. Cette action est appelée un *scan*.

3.1.2 Description de la Couche MAC

La couche MAC définit deux types de noeuds :

1. ***Full Function Devices* (FFDs)** : Les FFDs sont équipés de toutes les fonctions de la couche MAC il représente le coordinateur PAN ou un routeur. Quand un noeud est un FFD il a l'habilité d'envoyer des *beacons*, permet la synchronisation, la communication et la possibilité de connecter des services réseaux.
2. ***Reduced Function Devices* (RFDs)** : Les RFDs possèdent des fonctionnalités limitées par rapport au FFDs il représente généralement des noeuds puits (end-devices). Ils sont prévus pour des applications simples (signaler l'état d'un capteur, contrôler l'activation d'un actionneur, etc.). Ils ne peuvent interagir qu'avec un seul FFD. Dans tous les cas, les FFDs et RFDs doivent utiliser le même canal physique pour communiquer.

La couche MAC 802.15.4 supporte deux modes de fonctionnement selon les besoins applicatifs : le mode suivi de *beacon* (avec beacon ou beacon-enabled) et le mode non suivi de *beacon* (sans beacon). En mode avec *beacon*, le coordinateur envoie périodiquement un *beacon* pour synchroniser l'activité des noeuds qui lui sont attachés selon une structure de supertrame donnée par la figure 3.4. En mode sans *beacon*, les *beacons* ne sont utilisés que pour la découverte du réseau.

Nous allons décrire dans cette partie le fonctionnement de cette couche MAC, puis décrire les autres fonctionnalités de gestion essentielles de cette couche MAC tels que l'accès au médium, les *scans*, la création du réseau et l'association, la synchronisation avec un coordinateur, les échanges de trames.

Accès au médium

Un coordinateur limite l'accès au médium en utilisant la diffusion de *beacon* délimitant des supertrames. La structure de la supertrame est définie par les deux paramètres BI (*Beacon Interval*) qui définit l'intervalle qui sépare deux *beacons* consécutifs et SD (*Superframe Duration*) qui définit la durée d'activité de la supertrame. Les valeurs de BI et SD sont calculés en fonction de deux paramètres : BO (*Beacon Order*) et SO (*Superframe Order*) selon les formules suivantes :

$$SD = aBaseSuperframeDuration \times 2^{SO}$$
$$BI = aBaseSuperframeDuration \times 2^{BO}$$

avec $0 \leq SO \leq BO \leq 14$, $aBaseSuperframeDuration$ est un attribut de la couche MAC qui définit la durée de la supertrame quand $SO = 0$ ($15.36ms$ pour valeur par défaut pour $aBaseSuperframeDuration$). $aBaseSuperframeDuration = aBaseSlotDuration \times aNumSuperframeSlots$, avec $aBaseSlotDuration$ le nombre de symboles (un symbole vaut 4 bits) constituant un slot de la supertrame quand $SO = 0$. $aBaseSlotDuration$ a 60 comme valeur par défaut. $aNumSuperframeSlots$ est le nombre de slots qui constituent la supertrame, il vaut 16. La figure 3.3 décrit les durées en fonction de SO pour une fréquence de 2400MHz.

SO	SD	2400MHz
0	960	$15,36ms$
1	1920	$30,72ms$
2	3840	$61,44ms$
3	7680	$122,88ms$
4	15360	$245,76ms$
...	...	...
14	$15,7 \times 10^6$	$251,6s$

FIGURE 3.3 – Durée en fonction de SO de la partie active

La portion active de la supertrame est découpée en $aNumSuperframeSlots$ slots de temps égaux et est composée de trois parties : le beacon, la CAP (*Contention Access Period*) et la CFP (*Contention Free Period*). Durant la CAP toutes les stations sont en compétition pour accéder au médium alors que la CFP est constituée de slots de temps, appelés GTS (*Guaranteed Time Slot*) alloués à chaque station pour communiquer avec le coordinateur.

Le début du premier slot est l'instant d'envoi du beacon. La CAP suit la fin de transmission du *beacon* et la CFP, si elle est présente, suit immédiatement la fin de la CAP. Si un coordinateur ne souhaite pas appliquer la structure de la supertrame, il initialise les valeurs de BO et de SO à 15. Dans ce cas, nous nous retrouvons dans un PAN en mode non suivi de *beacon*.

La figure 3.4 montre un exemple d'une supertrame incluant une période de CAP, une période de CFP contenant deux GTS de tailles différentes et une période d'inactivité (car $BO = SO + 1$).

L'accès au médium durant la CAP pour toute trame, sauf les acquittements et les *beacons*, est géré selon l'algorithme de CSMA/CA slotté (figure 3.5). Avant tout envoi, le noeud doit s'assurer de pouvoir finir la transaction (incluant la réception d'un acquittement s'il est demandé) et de pouvoir attendre un IFS (*InterFrame Space*) [1] avant la fin de la CAP. Si ce n'est pas le cas, l'envoi est reporté à la CAP de

1. La durée d'un IFS dépend principalement de la longueur du MPDU (MAC Protocol Data Unit) de la trame envoyée, ce qui correspond à la *payload* physique

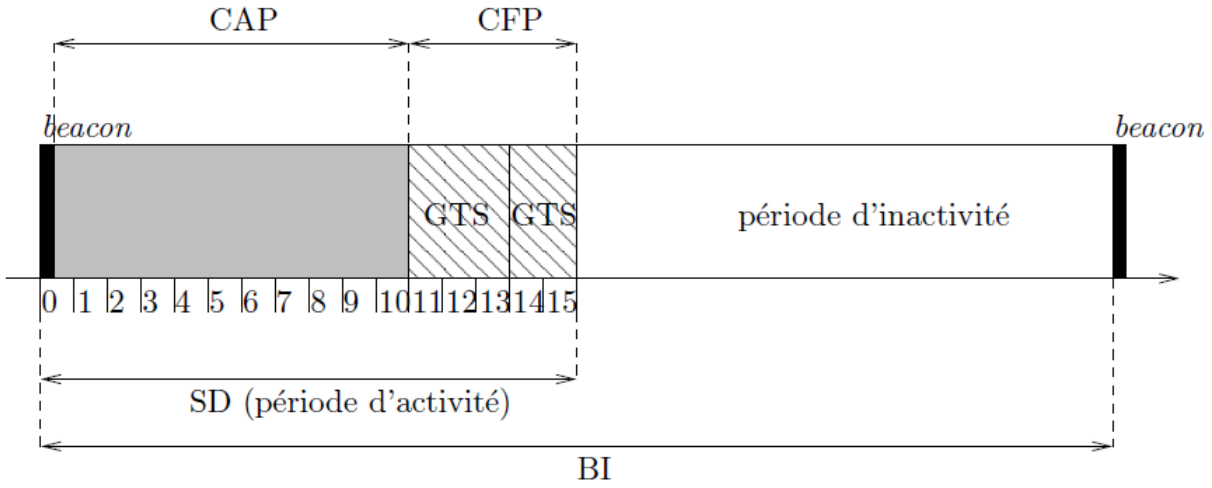


FIGURE 3.4 – La structure de la supertrame

la supertrame suivante. L'envoi de trames durant les slots réservés au GTS s'effectue sans CSMA/CA.

La proportion de temps pendant lequel le système peut travailler représente des "duty cycle". Pour 100% de duty cycle les noeuds sont tout le temps allumés, ce qui représente en fait que $BI = SD$ donc $BO = SO$. En dessous de 100% de duty cycle les noeuds peuvent éteindre leur émetteur pour réduire la consommation d'énergie. Ce pourcentage peut être calculé de cette façon :

$$duty\ cycle = \frac{SD}{BI} = (\frac{1}{2})^{BO-SO}$$

Les types de balayages, la création du réseau, les associations et la synchronisation

Les types de balayages Afin de découvrir les réseaux existants à portée, de créer un réseau, de s'associer à un réseau ou de se synchroniser avec un réseau, la couche MAC effectue un des quatre types de balayages suivants utilisables par les FFDs :

- ➢ Le balayage à détection d'énergie permet de récupérer l'énergie maximum détectée sur chaque canal. Ce balayage est utilisé par un coordinateur souhaitant créer un PAN afin de choisir le canal convenable.
- ➢ Le balayage actif de canal permet de récupérer la liste des identifiants de PAN existants. Le balayage actif suit la diffusion de requête de *beacon*. Un coordinateur fonctionnant selon un mode sans *beacon* répond à cette requête par une diffusion de *beacon*. Un coordinateur fonctionnant selon un mode avec *beacon* l'ignore et continue à envoyer ses *beacons* périodiquement. Ce balayage est utilisé par un coordinateur souhaitant créer un PAN pour choisir le bon identifiant de PAN, ou par une station qui cherche à s'associer à un PAN (dont elle connaît l'identifiant).
- ➢ Le balayage passif de canal comme pour le balayage actif, permet de récupérer la liste des identifiants de PAN existants. En revanche, la station n'envoie pas une requête de *beacon* mais elle effectue une écoute passive de chaque canal

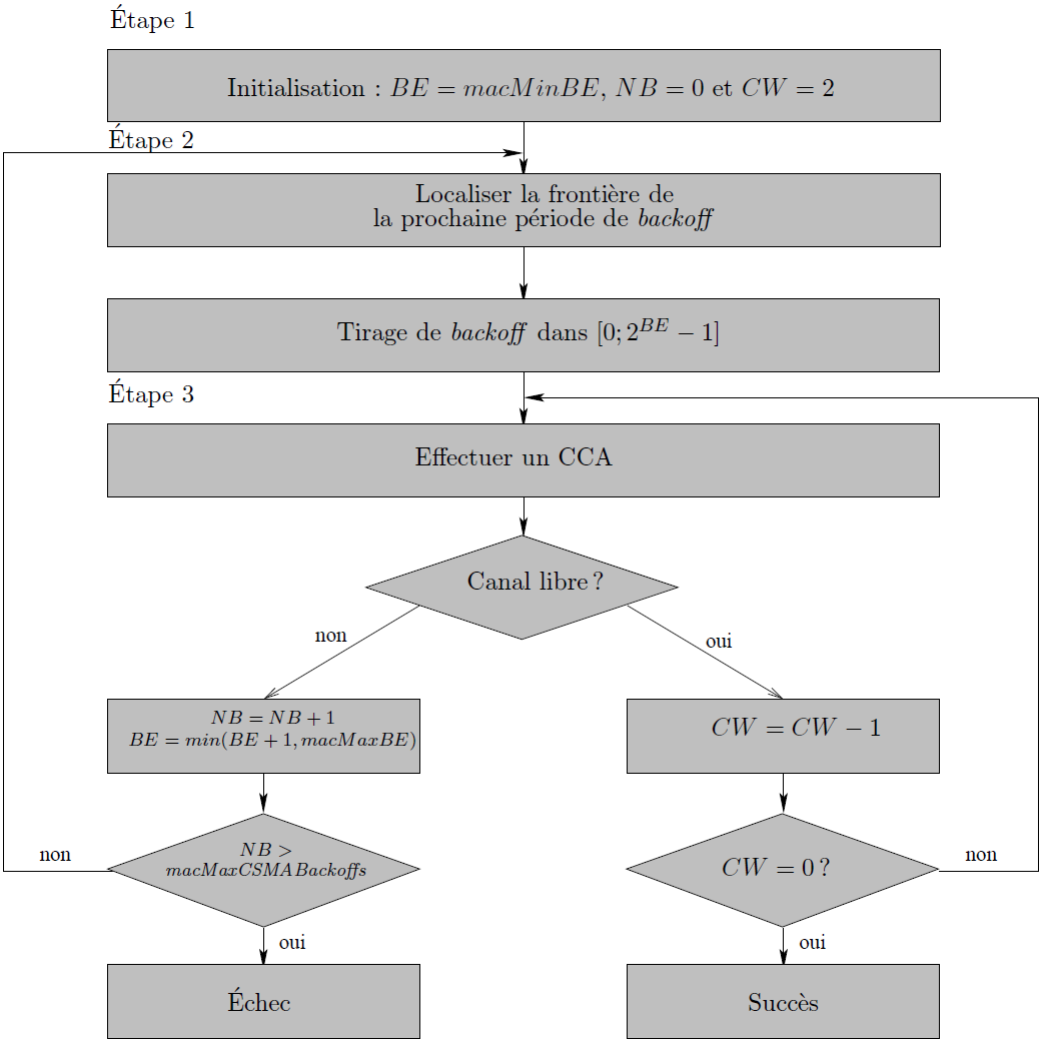


FIGURE 3.5 – Diagramme de l'algorithme de CSMA/CA slotté du standard 802.15.4

à scanner pour une durée donnée. Ce balayage est utilisé par une station qui cherche à s'associer à un PAN.

➢ Le balayage d'orphelin permet à une station de retrouver son coordinateur après une perte de synchronisation avec ce dernier.

Création du réseau La couche supérieure envoie une requête à la couche MAC pour effectuer un balayage actif sur une liste de canaux afin de découvrir les réseaux existants à porter. Une fois le bon canal choisi, la couche supérieure décide d'un identifiant de PAN et demande à la couche MAC d'initier un PAN avec cet identifiant.

Association et dissociation Après avoir effectué un balayage (passif ou actif) et remonté le résultat dû à la couche supérieure, cette dernière demande à la couche MAC de s'associer à un PAN spécifique en précisant son identifiant PAN et l'adresse du coordinateur correspondant. Suite à cette demande, la couche MAC génère une

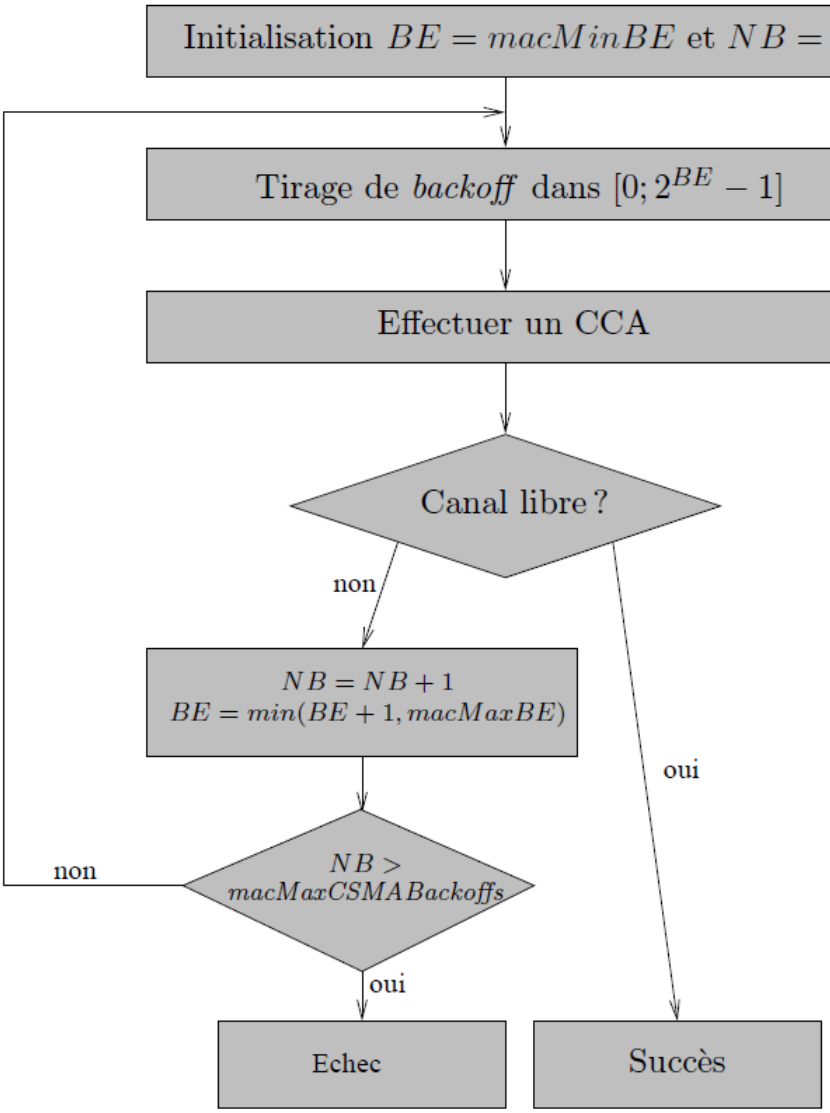


FIGURE 3.6 – Diagramme de l'algorithme de CSMA/CA non slotté du standard 802.15.4

requête d'association à destination du coordinateur.

A la réception d'une requête d'association, la couche MAC du coordinateur remonte l'information à la couche supérieure et c'est à celle-ci de décider d'accepter ou pas cette requête en fonction des ressources qu'il lui reste par exemple. La réponse à cette requête d'association est envoyée en mode de transmission indirecte (sous-section 3.1.2).

Suite à une requête de dissociation envoyée par la couche supérieure, la couche MAC envoie une indication de dissociation au coordinateur pour l'informer que le noeud souhaite se dissocier du PAN. De même, la couche supérieure d'un coordinateur peut décider de dissocier un noeud qui lui est associé, elle lui envoie alors une commande de dissociation pour l'informer de ce fait.

Synchronisation avec le *beacon* du coordinateur Dans le mode avec *beacon*, les entités restent synchronisées au réseau grâce à la transmission périodique d'une trame de *beacon* par le coordinateur. La figure 3.7 montre les différents champs de ce beacon.

Toute station appartenant à un *beacon-enabled* PAN doit pouvoir se synchroniser avec le *beacon* de son coordinateur pour connaître la structure de la supertrame, et aussi pour récupérer les trames la concernant. Si un noeud ne reçoit pas un nombre $aMaxLostBeacons$ consécutifs de *beacons*, la couche MAC détecte une perte de synchronisation et indique ce constat à la couche supérieure.

La couche MAC rejette toute trame de *beacon* dont l'adresse source ou l'identifiant du PAN ne correspondent pas à l'adresse du coordinateur ou à l'identifiant du PAN auquel le noeud est associé, respectivement.

Octets : 2	1	2	2/8	0/5/6/ 10/14	2	varie	varie	varie	2
Frame control	No de séquence	PAN ID source	Adresse source	Entête sécurité	Spécifications de la super-trame	Champs de GTS	Liste des adresses en attente	Payload	FCS
Entête MAC					Payload MAC				Footer MAC

FIGURE 3.7 – Structure du *beacon* du standard 802.15.4

Transmission des données

Dans la primitive de requête de transmission de données envoyée par la couche supérieure, un champ (*txOptions*) spécifie dans quel mode de transmission des données doivent être transmises. Les trois modes de transmissions sont : transmission directe, transmission indirecte, Transmission en utilisant les GTS. Ce champ permet aussi de spécifier si la trame de données doit être acquittée ou pas.

Transmission directe Durant la CAP, la station essaie d'envoyer la trame en CSMA/CA slotté. En cas de transmission avec demande d'acquittement, la couche MAC fait $macMaxFrameRetries$ tentatives (par défaut 3 tentatives) si l'acquittement n'est pas reçu au bout de $macAckWaitDuration$ [2].

Transmission indirecte Pour favoriser l'aspect économie d'énergie, les stations feuilles initient la communication avec leur coordinateur, et cela selon deux procédures : (i) dans un PAN en mode avec *beacon*, le coordinateur indique dans son *beacon* la liste des adresses des entités qui ont des trames en attente chez lui, (ii) dans un

2. $macAckWaitDuration = aUnitBackoffPeriod + aTurnaroundTime + phySHRDuration + 6 \times phySymbolsPerOctet$, où
$aUnitBackoffPeriod = 320\mu s$,
$aTurnaroundTime = 192\mu s$,
$phySHRDuration = 128\mu s$ (pour la fréquence de 2.4GHz), 6 représente le nombre d'octets de l'en-tête PHY incluant la longueur de la *payload* PHY de l'acquittement

PAN en mode sans *beacon*, la station feuille sollicite le coordinateur pour vérifier s'il a des trames en attente pour elle.

Le coordinateur conserve les trames en attente pour une durée de *macTransactionPersistenceTime* avant de les supprimer si la station concernée ne l'a pas sollicité pour les récupérer.

Réservation de temps (GTS) L'allocation des GTS est faite uniquement par le coordinateur du PAN. Un GTS est alloué soit en mode réception (du coordinateur du PAN vers un noeud) soit en mode émission (d'un noeud vers le coordinateur du PAN).

Une station demande l'allocation d'un GTS auprès du coordinateur du PAN auquel elle est affiliée. Le nombre maximal de GTS est limité à 7, à condition que la durée de la CAP restante reste supérieure ou égale à $aMinCAPLength$ ($= 7ms$ pour la fréquence de 2,4GHz).

La réponse d'une requête de GTS est envoyée dans le *beacon* via des descripteurs de GTS. Cette information est gardée dans le *beacon* pour $aGTSDescPersistenceTime$ ($= 4$ par défaut) *beacons* consécutifs.

La désallocation d'un GTS peut être faite soit à la demande de la couche supérieure de la station ayant le GTS, soit par la couche supérieure du coordinateur du PAN suite au constat que la station n'a plus utilisé son GTS durant $2 \times n$ [3] supertrames consécutives.

3.2 Les protocoles de routage

Le routage est une méthode d'acheminement des informations à la bonne destination à travers un réseau de connexion donné. Le problème de routage consiste à déterminer un acheminement optimal des paquets à travers le réseau au sens d'un certain critère de performance. Le problème consiste à trouver l'investissement de moindre coût en capacités nominales et de réserves qui assure le routage du trafic nominal et garantit sa survie en cas de n'importe quelle panne d'arc ou de noeud.

Le problème qui se pose dans le contexte des réseaux ad hoc est l'adaptation de la méthode d'acheminement utilisée avec le grand nombre d'unités existant dans un environnement caractérisé par de modestes capacités de calcul et de sauvegarde et de changements rapides de topologies. Il semble donc important que toute conception de protocole de routage doive étudier les problèmes suivants :

1. La minimisation de la charge du réseau
2. Offrir un support pour pouvoir effectuer des communications multipoints fiables
3. Assurer un routage optimal
4. Offrir une bonne qualité concernant le temps de latence

Suivant la manière de création et de maintenance de routes lors de l'acheminement des données, les protocoles de routage peuvent être séparés en trois catégories :

1. **Les protocoles "centralisés" ou dédiés**

3. $n = 2^{8-BO} si 0 \leq BO \leq 8,\ n = 1 si\ 9 \leq BO \leq 14$

2. **Les protocoles proactifs**
3. **Les protocoles réactifs**

3.2.1 Protocoles centralisés

Les protocoles dits "centralisés" fonctionnent sur des réseaux de petite taille. Or, même si les réseaux de capteurs sans fil sont apparentés aux réseaux ad hoc, les spécificités, les objectifs et les besoins sont différents. Les protocoles dédiés pour les réseaux de capteurs, sont peu nombreux, les implémentations également. Les noeuds demandent une gestion soigneuse des ressources. Ces algorithmes dédiés sont répartis en quatre catégories [43] :

1. **Protocoles hiérarchiques**
2. **Protocoles basés sur la localisation**
3. **Protocoles "*Data centric*"**
4. **Protocoles basé sur la qualité de service (*QoS*)**

Protocoles hiérarchiques

Ici les noeuds ne sont pas tous au même niveau, ils sont de type différent et n'ont pas la même capacité. Les protocoles hiérarchiques fonctionnent sous forme de clusters. Un cluster est composé de plusieurs noeuds avec un chef qui est chargé de la transmission des messages générés par son cluster aux autres chefs de clusters pour atteindre la destination finale. Le chef peut être soit un noeud possédant des capacités accrues, ou se faisant à tour de rôle en fonction du nombre de voisins et du niveau d'énergie résiduelle.

On retrouve notamment le protocole LEACH [49], qui choisit aléatoirement les noeuds cluster-heads et attribue ce rôle aux différents noeuds selon la politique de gestion Round-Robin [107] pour garantir une dissipation équitable d'énergie entre les noeuds. Dans l'optique de réduire la quantité d'informations transmises à la station de base, les *clusters-heads* agrègent les données capturées par les noeuds membres qui appartiennent à leur propre cluster, et envoient un paquet agrégé à la station de base. On retrouve également d'autres protocoles tels que : PEGASIS [67], TEEN [71] ou encore DirQ [26].

Protocoles basés sur la localisation

Dans ce système les capteurs ont un système GPS basse consommation qui leur permet de déduire la distance et le coût. En effet, ce type de protocole considère que les noeuds connaissent leur position respective et sont capables de connaître la position des autres noeuds. Ce qui facilite la diffusion d'informations vers une région dans laquelle se trouve la destination.

On retrouve notamment le protocole GEAR [113] qui découpe le réseau en régions. L'acheminement des paquets est établi en fonction de la distance et l'énergie, le noeud le plus proche de la région parmi ses voisins avec le niveau d'énergie le plus élevé et l'acheminement dans la région destinataire soit par diffusion si le nombre de noeuds n'est pas élevé, sinon on découpe la région en sous-région et on transmet

individuellement à chaque sous-région. Il existe aussi GAF [109] qui lui aussi est un protocole basé sur la localisation.

Protocoles *Data centric*

Ces protocoles communiquent sans avoir d'adresses ou d'identifiant. La communication s'effectue à travers certaines régions, de proche en proche à travers le réseau. En général, pour effectuer une transmission de données, la source envoie sur le réseau une annonce de donnée à transmettre avant l'envoie des données puis les voisins étant intéressés par l'annonce envoie une requête d'obtention des données, une fois toutes les requêtes reçues à ce moment les données sont envoyées.

On retrouve notamment le protocole SPIN [62] qui permet de disséminer des informations sur le réseau de manière ciblée. Son fonctionnement permet de réduire la charge du réseau par rapport aux méthodes de diffusion traditionnelles telle que l'inondation. [4] Son avantage est la connaissance limitée au voisinage à un saut. Néanmoins il ne garantit pas la réception des données si la destination ne se trouve pas à portée du noeud émetteur et donc aucune retransmission n'est possible. Il existe beaucoup d'autres protocoles tels que : Direct Diffusion [54], Rumor [20], COUGAR [110] ou encore ACQUIRE [96].

Protocoles basés sur la qualité de service

Ces protocoles adoptent des approches d'établissement de chemins en considérant la qualité de service. Les performances réseau sont prises en compte pour garantir un délai fiable qui répond aux besoins de l'application. On retrouve ces protocoles surtout dans le cadre d'applications industrielles et militaires.

On retrouve notamment le protocole SAR [12] qui construit des arbres à partir des voisins du puits. Les liens de chaque arbre sont établis en fonction du délai observé et de l'énergie résiduelle des noeuds. Les données se retrouvent associées à un niveau de priorité. Néanmoins l'inconvénient le plus important concerne la création des arbres qui reste une opération très coûteuse. Le protocole SPEED [48] est une amélioration du protocole GEAR qui se base sur les délais d'aller-retour en retranchant le temps de traitement côté récepteur. D'où un choix du prochain saut le plus proche de la destination que le noeud.

3.2.2 Protocoles pro-actifs

Les protocoles de routages pro-actifs constituent une catégorie de routage ad hoc dans laquelle chaque élément du réseau cherche à établir des tables de routages valides en permanence. Pour ce faire, ces protocoles pro-actifs échangent en permanence même si aucun transfert de données n'est demandé. Pour résumer, les performances du routage pro-actif se caractérisent par :

- ➢ les latences les plus faibles, puisque les applications souhaitant émettre peuvent supposer l'existence de routes à jour et valides,

4. Diffusion à tous les voisins, puis retransmission à tous les voisins, ainsi de suite sauf que la destination ne transmet rien

- ➢ une consommation d'énergie importante et une utilisation importante de la bande passante, due au fait que le protocole de mise à jour des routes est employé en permanence, même si les routes ne sont jamais exploitées,
- ➢ une utilisation importante de mémoire vive, due à la nécessité de mémoriser sur le long terme les tables de routages, y compris pour les routes peu ou pas utilisées.

DSDV [84]

Il est basé sur l'idée classique de l'algorithme distribué de Bellman-Ford en rajoutant quelques améliorations. Chaque station mobile maintient une table de routage qui contient :

- ➢ Toutes les destinations possibles.
- ➢ Le nombre de noeuds (ou de sauts) nécessaires pour atteindre la destination.
- ➢ Le numéro de séquences (SN : sequence number) qui correspond à un noeud destination.

Le SN est utilisé pour faire la distinction entre les anciennes et les nouvelles routes, ce qui évite la formation des boucles de routage.

La mise à jour dépend donc de deux paramètres : Le temps, c'est-à-dire la période de transmission, et les évènements. Un paquet de mises à jour contient :

1. Le nouveau numéro de séquence incrémenté, du noeud émetteur. Et pour chaque nouvelle route :
2. L'adresse de la destination.
3. Le nombre de noeuds (ou de sauts) séparant le noeud de la destination.
4. Le numéro de séquence (des données reçues de la destination) tel qu'il a été estampillé par la destination.

DSDV élimine les deux problèmes de boucle de routage "routing loop", et celui du "counting to infinity". Cependant, dans ce protocole, une unité mobile doit attendre jusqu'à ce qu'elle reçoive la prochaine mise à jour initiée par la destination, afin de mettre à jour l'entrée associée à cette destination, dans la table de distance. Ce qui fait que le DSDV est lent. DSDV utilise une mise à jour périodique et basée sur les évènements, ce qui cause un contrôle excessif dans la communication.

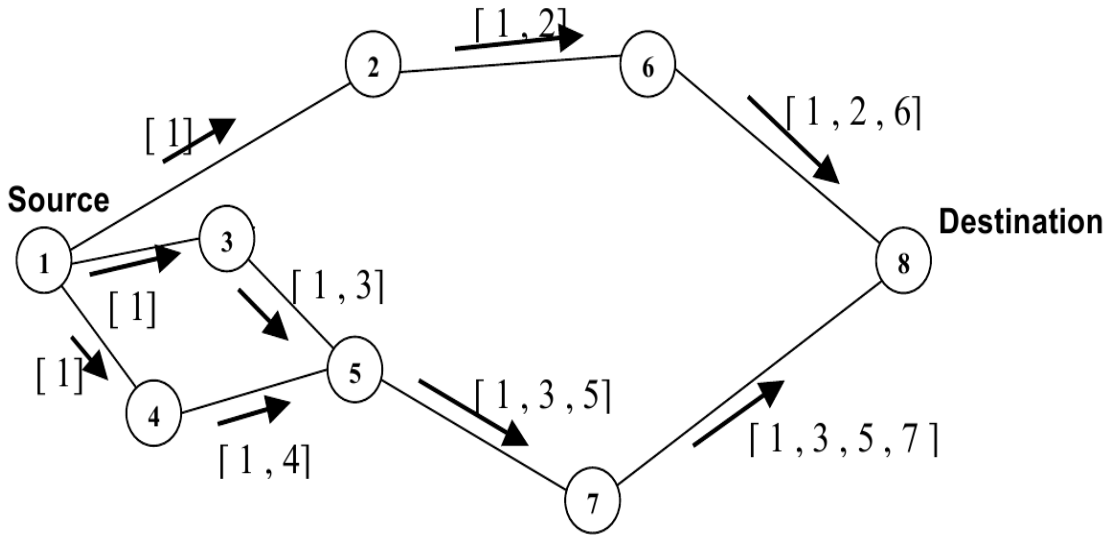
Comme autres protocoles pro-actifs on retrouve les protocoles : OLSR [55], CGSR [30], FSR [81], STAR [42], WRP [74], etc.

3.2.3 Protocoles réactifs

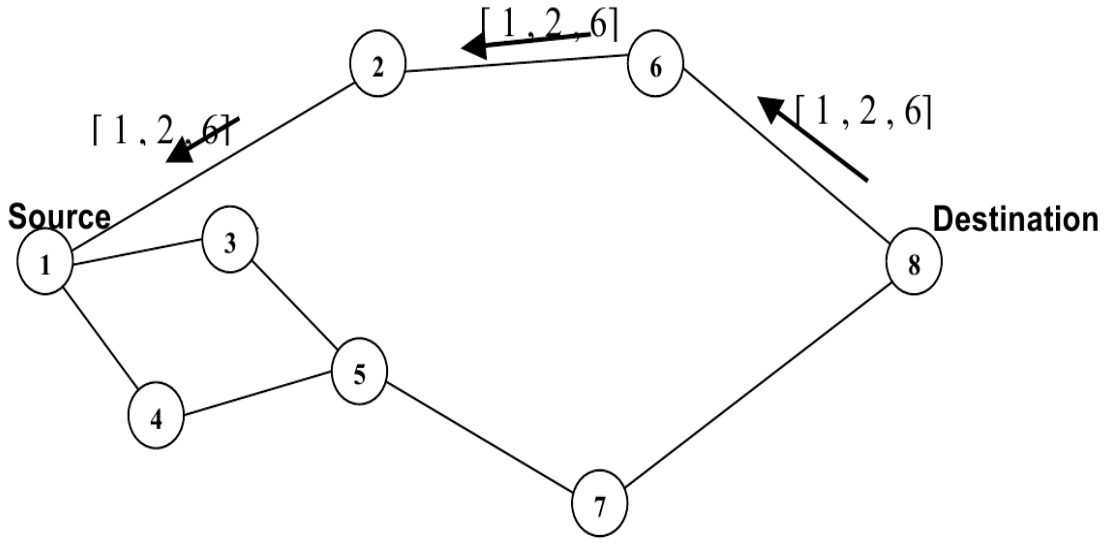
DSR [56]

Le protocole "Routage à Source Dynamique" (DSR), est basé sur l'utilisation de la technique "routage source". Dans cette technique : la source des données détermine la séquence complète des nœuds à travers lesquelles, les paquets de données seront envoyés. Un site initiateur de l'opération de " découverte de routes ", diffuse un paquet requête de routes. Si l'opération de découverte est réussite, l'initiateur reçoit un paquet réponse de routes qui liste la séquence de noeuds à travers lesquels la

destination peut être atteinte. Le paquet requête de routes contient donc un champ enregistrement de routes, dans lequel sera accumulée la séquence des noeuds visités durant la propagation de la requête dans le réseau, comme le montre la Fig. 3.8



(a) Construction de l'enregistrement de route



(b) Le renvoi du chemin

FIGURE 3.8 – La découverte de chemins dans le protocole DSR

Afin d'assurer la validité des chemins utilisés, le DSR exécute une procédure de maintenance de routes :

- ➢ Quand un noeud détecte un problème fatal de transmission, à l'aide de sa couche de liaison, un message erreur de route (route error) est envoyé à l'émetteur original du paquet.
- ➢ Le message d'erreur contient l'adresse du noeud qui a détecté l'erreur et celle du noeud qui le suit dans le chemin.

➢ Lors de la réception du paquet erreur de route par l'hôte source, le noeud concerné par l'erreur est supprimé du chemin sauvegardé, et tous les chemins qui contiennent ce noeud sont tronqués à ce point là. Par la suite, une nouvelle opération de découverte de routes vers la destination est initiée par l'émetteur.

L'utilisation de la technique "routage source", fait que les noeuds de transit n'aient pas besoin de maintenir les informations de mise à jour pour envoyer les paquets de données, puisque ces derniers contiennent toutes les décisions de routage. Dans ce protocole, il y a une absence totale de boucle de routage, car le chemin source-destination fait partie des paquets de données envoyés.

AODV [83]

Le protocole AODV représente essentiellement une amélioration de l'algorithme DSDV. Le protocole AODV, réduit le nombre de diffusions de messages, et cela, en créant les routes lors du besoin, contrairement au DSDV, qui maintient la totalité des routes. L'AODV utilise les principes des numéros de séquence à fin de maintenir la consistance des informations de routage. A cause de la mobilité des noeuds dans les réseaux ad hoc, les routes changent fréquemment ce qui fait que les routes maintenues par certains noeuds deviennent invalides. Les numéros de séquence permettent d'utiliser les routes les plus nouvelles (fresh routes). De la même manière que dans DSR, AODV utilise une requête de route dans le but de créer un chemin vers une certaine destination. Cependant, AODV maintient les chemins d'une façon distribuée en gardant une table de routage, au niveau de chaque noeud de transit appartenant au chemin cherché. Un noeud diffuse une requête de route dans le cas où il aurait besoin de connaître une route vers une certaine destination et qu'une telle route n'est pas disponible. Cela peut arriver :

➢ Si la destination n'est pas connue au préalable, ou

➢ Si le chemin existant vers la destination a expiré sa durée de vie ou il est devenu défaillant.

Le champ numéro de séquence destination du paquet RREQ, contient la dernière valeur connue du numéro de séquence, associé au noeud destination. Cette valeur est recopiée de la table de routage. Si le numéro de séquence n'est pas connu, la valeur nulle sera prise par défaut. Le numéro de séquence source du paquet RREQ contient la valeur du numéro de séquence du noeud source. Afin de maintenir des routes consistantes, une transmission périodique du message "HELLO" est effectuée. Si trois messages "HELLO" ne sont pas reçus consécutivement à partir d'un noeud voisin, le lien en question est considéré défaillant. Le protocole AODV ne présente pas de boucle de routage, en outre il évite le problème "counting to infinity" de Bellman-Ford, ce qui offre une convergence rapide quand la topologie du réseau ad hoc change.

AOMDV [72]

L'objectif primaire du protocole réactif AOMDV (Ad hoc On demand Multipath Distance Vector) est de fournir une tolérance efficace aux fautes, dans le sens où le rétablissement en cas d'échec d'une route est plus rapide. Pour réaliser cet objectif, AOMDV construit plusieurs routes sans boucles de routage allant de la source jusqu'

à la destination ; contrairement à AODV qui construit et maintient une seule route seulement pour chaque couple (source, destination). Les protocoles de routage à la demande multiroutes, telle que AOMDV, essaient d'atténuer la latence élevée de la procédure de découverte de route pouvant affecter les performances défavorablement. AOMDV calcule de multiples chemins lors d'une simple tentative de découverte de route. Ces multiples chemins vont être formés aussi bien dans les sources de trafic que dans les noeuds intermédiaires. Une nouvelle découverte de route n'est nécessaire, que si toutes les routes deviennent non valides. Ceci réduit considérablement les latences et la surcharge de la procédure de découverte.

L'idée principale dans AOMDV consiste à calculer différentes routes, allant de la source de trafic jusqu' à la destination, tout en évitant la formation de boucles de routage. Notons que dans AODV, chaque requête (respectivement réponse) reçue par un noeud pendant le processus de découverte de route définit potentiellement une route alternative vers la source (respectivement destination). Par exemple, chaque copie du paquet de requête RREQ arrivant à un noeud définit une route alternative vers la source. Cependant, accepter naïvement toutes ces copies va mener à la formation de boucles dans les routes.

Afin d'éliminer la possibilité de formation de boucles, les auteurs proposent de maintenir la même condition que dans AODV, et qui consiste à n'accepter une nouvelle requête que si :

- ➢ Le nombre de sauts est inférieur ou que
- ➢ Le numéro de séquence est supérieur.

La construction de différentes routes dans AOMDV est basée sur la notion de "advertised_hopcount". Pour chaque noeud sur la route, ce paramètre représente le nombre de sauts qui ont été nécessaires à la première requête pour atteindre ce noeud. Ce nombre de sauts est alors considéré comme l'advertised_hopcount, et ne peut changer pour le même numéro de séquence. AOMDV n'acceptera par la suite, que les routes alternatives, dont le nombre de sauts, est inférieur ou égal à l'advertised_hopcount. Cette condition est suffisante pour garantir la non-formation de boucles.

destination
numéro de séquence
hopcount
nexthop
expiration_timeout

(a) AODV

destination
numéro de séquence
advertised_hopcount
liste_de_routes $\{(nexthop_1, hopcount_1), (nexthop_2, hopcount_2), ...\}$
nexthop
expiration_timeout

(b) AOMDV

FIGURE 3.9 – Structure des tables de routage dans AODV et AOMDV

La Fig.3.9 montre la structure des tables de routage pour AODV et AOMDV. Dans AOMDV, l'advertised_hopcount remplace le nombre de sauts (hopcount) de AODV. Une liste appelée "liste_de_routes" remplace le prochain saut (nexthop). Cette liste définit pour chaque prochain saut ($nexthop_k$), le nombre de sauts ($hocount_k$)

nécessaire pour atteindre la destination en passant par ce noeud. Notons que tous les prochains sauts pour une même destination disposent du même numéro de séquence. Notons également que le paramètre advertised_hopcount est initialisé à chaque fois que le numéro de séquence est mis à jour.

L'idée principale dans AOMDV est de calculer différents chemins pendant la phase de découverte de route. Ainsi, une nouvelle découverte de route est nécessaire uniquement lorsqu'aucune des routes précédemment établies n'est valide.

Lorsqu'aucune route vers la destination n'est valide, le noeud envoie un paquet d'erreurs RERR à tous ses voisins qui utilisent ce noeud comme prochain saut dans la route vers cette destination. Ces routes sont effacées des tables de routage des noeuds récepteurs de ce paquet d'erreurs. En revanche, contrairement au protocole AODV, ces paquets d'erreurs ne sont pas relayés jusqu' à la source. Si la source de trafic reçoit un paquet RERR, elle lance une nouvelle découverte de route dans le cas où elle souhaite toujours émettre.

Depuis l'élaboration des ces protocoles pro-actifs et réactifs (DSDV, AODV, AOMDV, DSR) beaucoup d'amélioration y ont été apportées, on en retrouve quelques-unes dans [77, 45, 75].

Deuxième partie

Contribution

Chapitre 4

Impact de la couverture radio, de la topologie et du nombre de noeuds sur les performances d'un réseau de capteurs

Sommaire

Comme nous l'avons présenté dans les parties précédentes, le choix de configuration est un facteur déterminant pour les performances d'un réseau de capteurs. On retrouve dans cette configuration : le protocole d'accès au médium (MAC et PHY), le type de capteurs, le placement des capteurs (topologie) avec leur puissance radio, le protocole de routage qui ont tous des conséquences en fonction du type d'application. En effet, un mauvais choix peut impliquer une surconsommation d'énergie ou une fiabilité des données réduite, c'est pour cela que nous proposons un modèle permettant de choisir les paramètres adaptés à un réseau de capteurs. Néanmoins avant la production du modèle, nous analyserons l'impact de différents paramètres non présent dans la littérature.

Dans ce chapitre, nous allons tout d'abord nous intéresser à une étude de l'impact de la couverture radio, de la topologie et de la taille du réseau sur les performances du réseau de capteurs. Le contenu de ce chapitre reprend essentiellement celui des publications [91, 90].

Le chapitre est organisé comme suit : la section 4.1 décrit les différentes hypothèses de modélisation ainsi que la configuration de notre réseau de capteurs qui est étudiée. Puis dans les sections 4.3 et 4.2 nous analyserons les résultats numériques obtenus. Enfin la section 4.4 conclut le chapitre.

4.1 Modélisation du système

Dans [87], les auteurs ont montré que la communication entre les capteurs est l'activité la plus coûteuse. Cependant, c'est la seule méthode pour router l'information vers la station de base. Lorsque les capteurs communiquent entre eux, il existe des raisons au gaspillage d'énergie : les collisions, l'écoute passive, les paquets de contrôle, etc. Un noeud trop souvent sollicité par rapport à sa topologie (noeud routeur) peut être aussi une raison de gaspillage de l'énergie. Cependant, dans un premier temps nous allons voir que la couverture radio influe que de manière modérée sur le réseau en fonction du protocole de routage utilisé. Puis dans un second temps que l'impact de la topologie dépend principalement du modèle applicatif qu'il y a sur le réseau.

4.1.1 Hypothèses pour le réseau de capteurs

Nous commençons par une présentation du contexte propre au réseau de capteurs. Il y a plusieurs types de réseaux de capteurs cités précédemment [52]. Nous considérons qu'un réseau de capteurs sans fil se résume en un noeud (capteur) qui relève une donnée physique (environnemental) puis transmet cette donnée à la station de base pour effectuer les calculs. Cette donnée se retrouve véhiculée suivant le principe du protocole de routage. Les capteurs vont s'auto-organiser afin d'établir des liaisons vers la station de base dynamiquement. Dans ce cas, les capteurs sont déployés d'une manière dense et aléatoire. Périodiquement, les capteurs prennent les mesures et les envoient à la station de base via les communications multi-sauts. Comme les capteurs sont susceptibles d'être disposés de façon aléatoire, ils peuvent se retrouver soit comme des noeuds routeurs, des noeuds dans la même zone ou des noeuds puits (n'ayant aucun voisin). Ceci peut provoquer plusieurs événements (retransmissions liées aux collisions, informations non véhiculées, etc.) qui résultent d'une dégradation des performances.

Nous nous arrêtons uniquement à la partie réseau et circulation des données. Car La puissance de calcul, la quantité de mémoire, etc. a un impact modéré sur les performances et la fiabilité des données si l'application est bien conçue. Nous modélisons un réseau de capteurs possédant les paramètres décrits précédemment. Le but de l'étude est de connaître le réel impact de la couverture radio et de la topologie suivant les différents paramètres sur les performances du réseau de capteurs.

L'application Nous évaluerons les différents impacts suivant un modèle précis. Dans un premier temps, nous supposons que l'application détermine un débit de communication sur le réseau. Pour cela, nous proposons trois types d'applications :

1. Une application dite régulière qui caractérise une application qui envoie des données avec un grand intervalle. Ceci peut représenter le cas d'une surveillance du niveau de l'eau d'une rivière toutes les heures.
2. Une application dite avec un fort débit où les noeuds envoient de manière constante des données à travers le réseau, dans ce cas on obtient un réseau à la limite de la surcharge. Comme exemple, on peut prendre les capteurs qui véhiculent, en "streaming", une vidéo des caméras de surveillance d'une route.
3. Une application dite par à coup qui représente les applications transmettant beaucoup de données, mais à un intervalle de temps régulier. Comme exemple, on a le cas des capteurs permettant de compter les oiseaux [47] et transmettant des fichiers audio toutes les 2 heures.

Les Protocoles D'un point de vue accès au médium, nous utilisons le standard IEEE 802.15.4 en mode sans balises, pour toutes les possibilités de configuration, avec un nombre de noeuds compris entre 25 et 300. Les protocoles de routage utilisés sont les protocoles Ad-hoc On-demand Distance Vector (AODV), Ad-hoc On-demand Multipath Distance Vector (AOMDV), Destination-Sequenced Distance Vector (DSDV) et Dynamic Source Routing (DSR).

4.1.2 Topologie du réseau

Dans un réseau de capteurs orienté détection d'événement, le nombre de capteurs déployés sur la zone de détection dépend de l'événement à surveiller. S'il s'agit de la détection d'un cours d'eau, un nombre faible de capteurs est nécessaire, néanmoins dans le cas de la surveillance d'une forêt, il est nécessaire d'avoir un nombre important de capteurs. La distance entre les capteurs et la station de base est normalement plus longue que la portée radio de transmission. Donc, pour que les capteurs puissent envoyer les alertes à la station de base, il faut utiliser des transmissions multi-sauts.

La Figure 4.1 illustre un scénario simple de transmission d'un réseau de capteurs pour la détection d'événements. Dans ce réseau, il y a deux capteurs qui détectent chacun les différents événements, et deux capteurs routeurs qui véhiculent les messages vers la station de base. Nous supposons que les noeuds transmettent les messages vers le noeud le plus proche de la station de base. On veut que le réseau applique un protocole de routage optimal pour minimiser le nombre de transmissions multi-sauts. Pour cela ici, les deux capteurs relevant l'événement envoie leurs informations au noeud voisin le plus proche qui correspond au premier noeud routeur puis ces informations sont véhiculées jusqu'à atteindre la station de base. Néanmoins, même si le routeur R2 ne se trouve pas dans la portée radio des noeuds S1 et S2, il ne peut pas transmettre simultanément avec les noeuds S1 et S2 à cause du problème de la station cachée qui crée une collision [106]. En effet, au moment où le capteur S1 ou S2 va transmettre ses données au routeur R1 et si le routeur R2 transmet à la station de base, cette transmission va interférer la réception du

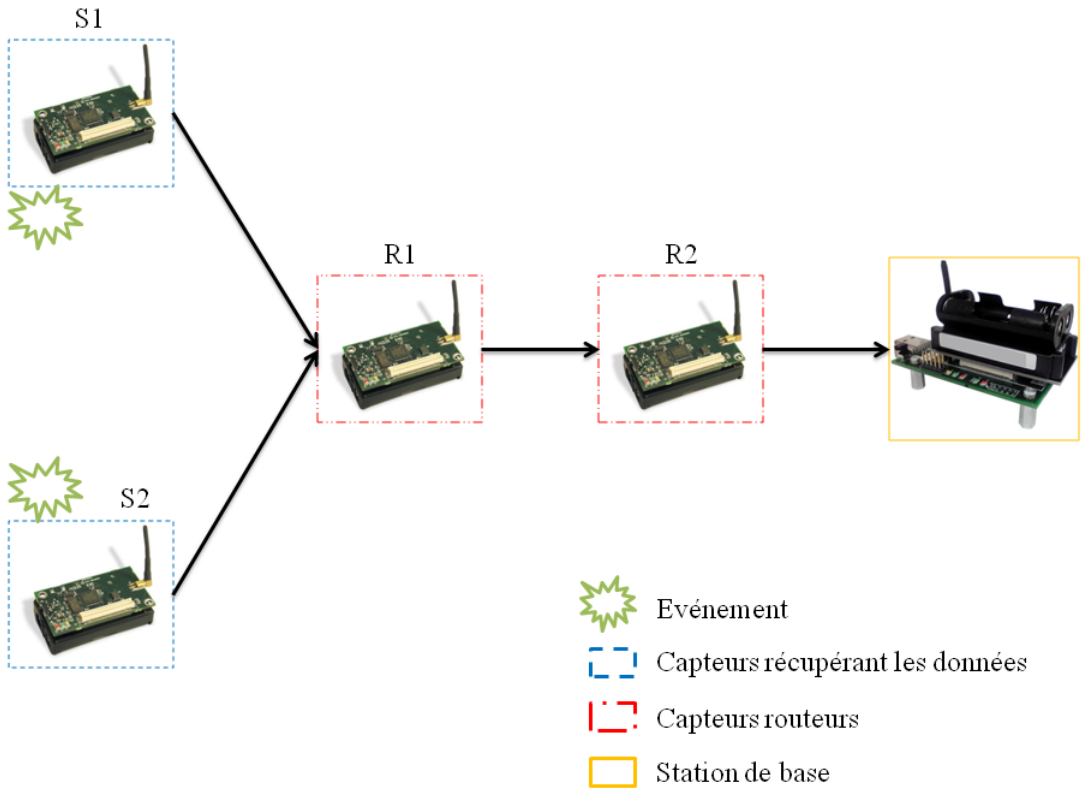


FIGURE 4.1 – Scénario de transmission d'un réseau de capteurs orientés événement

noeud routeur R1. Et donc dans ce scénario, il y a seulement un noeud qui a le droit de transmettre. Tous les autres doivent rester silencieux, sinon il y aura collisions. Depuis plusieurs mécanismes d'accès au canal ont été faits pour éviter les collisions, on en trouve certains dans [60, 66, 29].

D'après le principe de la Figure 4.1, pour garantir l'équité dans le réseau et permettre une fiabilité accrue, nous avons pris comme topologie standard une topologie sous forme de grille où chaque noeud est positionné à une certaine distance de ces voisins comme le montre la Figure 4.2. Dans cette topologie, n'importe quel noeud peut communiquer avec un autre et le risque de noeud défaillant n'influe pas sur la fiabilité, car un autre noeud est toujours prêt pour prendre la relève. De même, au niveau communication, les noeuds ont toujours le choix du chemin à adopter. Dans la majeure partie des cas, la station de base se retrouve au centre de la topologie, les noeuds relevant les événements sont en périphérie de cette topologie et le reste des noeuds sont considérés comme des noeuds routeurs.

Pour les autres topologies, nous considérons que nous sommes dans un cas où les capteurs sont déployés de façon aléatoire comme lors de déploiement de capteurs par avion pour une surveillance des fonds des océans. Dans ce cas, la station de base se retrouve n'importe où et les capteurs peuvent se retrouver très proches l'un de l'autre ce qui peut conduire à une augmentation du nombre de collisions. Nous

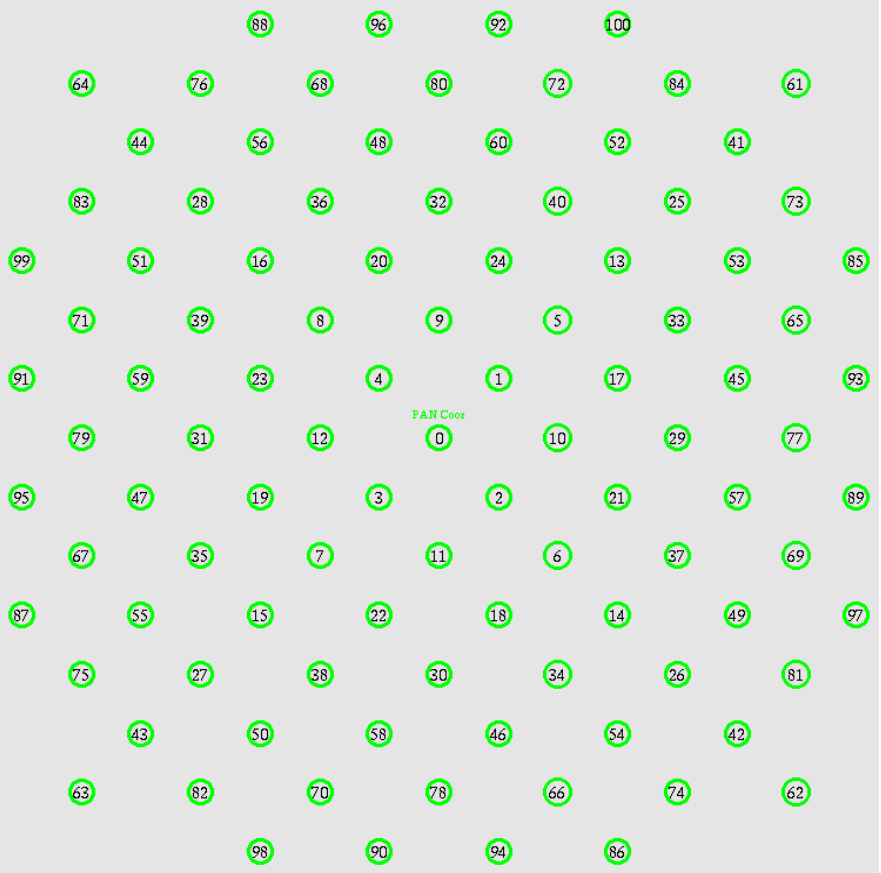


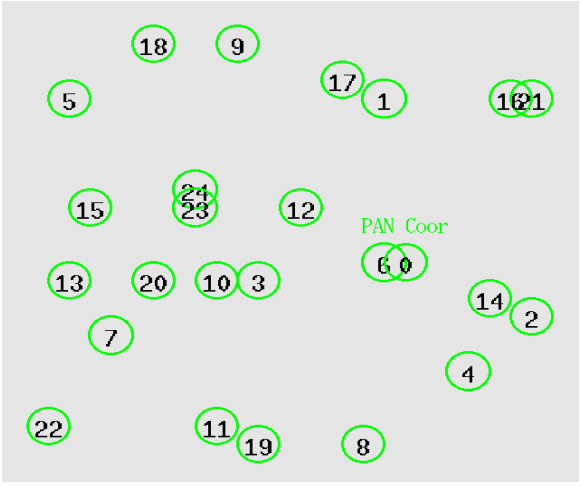
FIGURE 4.2 – Topologie sous forme de grille

évitons le cas extrême où la station de base n'a pas du tout de voisins. Dans ce cas l'application ne serait pas fonctionnelle, d'où on s'assure un minimum d'un voisin pour la station de base. La Figure 4.3 montre des exemples de topologies aléatoires pour 25 noeuds, où l'on remarque que la station de base qui représente le noeud 0 (PAN Coord.) se situe à différents endroits.

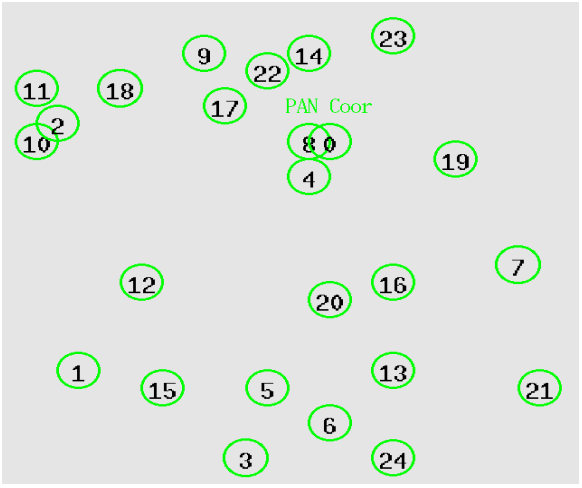
Etant donné que les noeuds accèdent au canal aléatoirement pour transmettre, il y a différents scénarios de transmissions pour les données. Dans le meilleur des cas, on suppose que la transmission s'est effectuée dans les meilleurs délais, c'est-à-dire que durant la transmission il n'y a eu aucune collision et retransmission. Dans le pire des cas, la transmission s'est alors déroulée pendant une longue période. Ceci implique qu'il y a eu des problèmes de transmissions qui peuvent être liés à des collisions, des problèmes radio, de capteurs et donc plusieurs retransmissions ont été nécessaires.

4.1.3 Couverture Radio

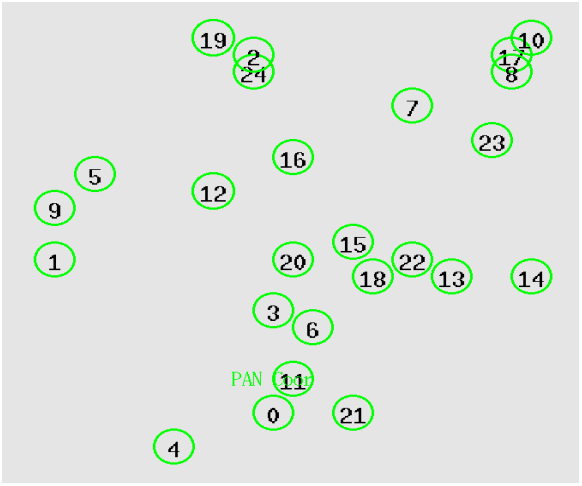
L'économie d'énergie et la prolongation de la durée de vie sont des tâches fondamentales dans les réseaux de capteurs. En outre, pour surveiller une zone importante, on étend le réseau de capteurs pour qu'il ait une couverture maximale. On retrouve une couverture pour le WiFi de l'ordre de 300 mètres, mais dans le Zigbee, une



(a) Topologie aléatoire 1



(b) Topologie aléatoire 2



(c) Topologie aléatoire 3

FIGURE 4.3 – Topologies aléatoires avec 25 noeuds

portée d'environ 100 mètres qui devient très sensible dès l'apparition d'obstacles. Dans cette optique, plusieurs stratégies ont été proposées dans la littérature pour ordonnancer l'activité des capteurs déployés dans la zone d'intérêt afin d'assurer la couverture totale de la zone [10]. Par conséquent, le choix des capteurs actifs dans le but d'économiser l'énergie et le maintien d'un niveau élevé de couverture sont deux aspects orthogonaux, car la minimisation du nombre de capteurs actifs tout en garantissant un certain taux de couverture de la zone d'intérêt est un problème NP-difficile.

La couverture de zone peut prendre plusieurs formes. La plus simple, lorsque tout point p de la zone des points cibles est couvert par un et un seul capteur c'est-à-dire que tout point p se trouve dans la zone de détection d'un seul capteur. Dans ce cas, on parle de la 1-couverture. En outre, on parle de la k-couverture ($k > 1$) lorsque tout point p de la zone des points cibles est couvert par plus d'un capteur. Par ailleurs, quand les capteurs sont déployés en grand nombre dans une zone de points cibles, la couverture de tout point cible reflète la couverture de la zone des points cibles, c.-à-d. si la zone de détection de tout capteur est couverte par un ensemble de capteurs actifs, la zone des points cibles est entièrement couverte. Ici, nous abordons le problème de performances lié à la couverture radio en terme de paquets perdus et de la consommation énergétique.

Pour cela nous étudierons 4 types de couvertures radios : 10, 25, 50 et 100 mètres. Sur la topologie fixe que nous avons montrée plus haut (fig. 4.2) avec la couverture minimale de 10 mètres, un noeud ne voit que ses voisins, puis avec 25, 50 et 100 mètres la couverture permet de voir une partie, voire la totalité du réseau et donc ceci permet de réduire la quantité de sauts nécessaires à atteindre la destination.

4.1.4 Paramètres de simulation

Pour effectuer notre étude nous avons utilisé le simulateur NS-2 [76] comme présenté précédemment qui est un simulateur à événement discret développé en C++. Nous l'utilisons pour la performance du code, la qualité des résultats et parce qu'il contient une large bibliothèque des protocoles les plus utilisés [38]. NS-2 propose une configuration très pointilleuse de chaque élément du réseau. Nous allons donc citer les différents paramètres de simulations :

Concernant la configuration d'un noeud on a :

- ➢ Le modèle de transmission sans-fil,
- ➢ Le modèle de propagation de l'onde radio (standard) qui représente le modèle Two Ray Ground [93].
- ➢ Le modèle de la couche MAC et la couche PHY qui représente les paramètres du standard IEEE 802.15.4 en mode sans balises utilisant la bande de fréquence des 2400MHz avec un débit de 250kps [15].
- ➢ Le protocole de routage qui est soit AODV, AOMDV, DSDV ou DSR.
- ➢ La quantité initiale d'énergie fixée à 2.5J (Joules).

Concernant l'application on a :

- ➢ Pour une application dite régulière un flux de type CBR (Constant Bit Rate) généré en UDP avec une taille des paquets constante et un envoi de paquets toutes les 2 secondes.

- ➢ Pour une application dite à fort débit, on a toujours un flux de type CBR mais avec des paquets plus gros à un intervalle de 0.2 secondes pour représenter la surcharge du réseau.
- ➢ Pour une application dite par à coup, on utilise un flux de type "Poisson" avec un envoi en continu de 0.5 sec et une période inactive de 2.5 secondes avec un débit de $50Kbs$. Le flux de type poisson génère un trafic exponentiel selon une période active et une inactivité d'envoi pendant la période inactive.

En se basant sur ces paramètres, nous proposons une approche qui représente la configuration des MicaZ [4], et les applications les plus couramment utilisées dans le monde des réseaux de capteurs.

4.1.5 Critère de performance

Les réseaux de capteurs étant des systèmes critiques, nos critères de performance sont basés sur la quantité de paquets perdus qui correspond à un élément déterminant concernant la fiabilité des données, la sécurité du réseau et la vitesse de réaction en fonction de l'événement reçu. Pour cela nous analyserons les paquets perdus d'un point de vue applicatif. La consommation d'énergie est également prise en compte dans ce critère de performance, car l'énergie correspond à un élément incontournable des réseaux de capteurs pour la longévité des applications.

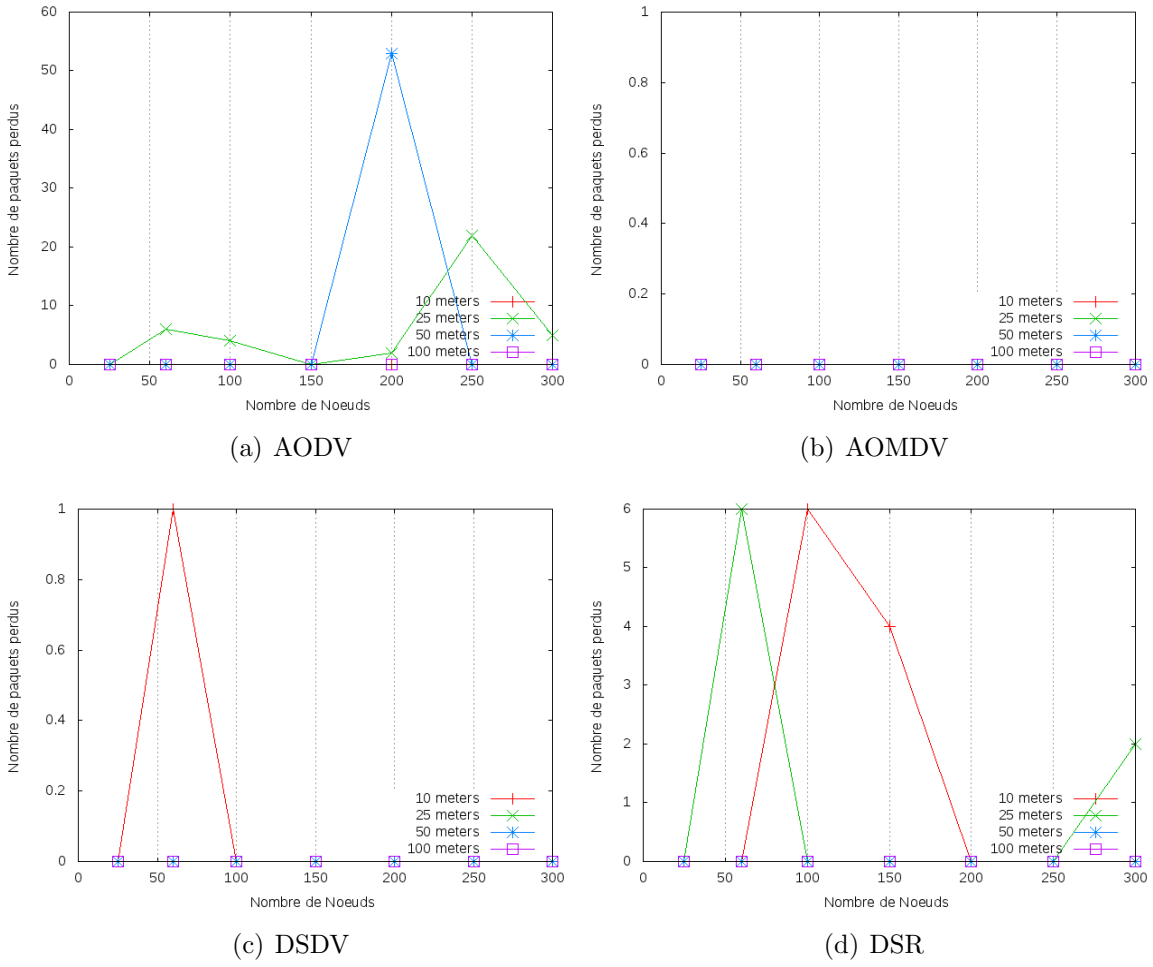
4.2 Impact de la couverture radio

4.2.1 Evaluation

Dans cette partie, nous allons effectuer notre évaluation pour montrer l'impact de la couverture radio sur les performances des réseaux de capteurs en utilisant le simulateur NS-2. Pour cela, nous ferons varier chacun des paramètres pour en sortir les résultats. Dans un premier temps, nous travaillerons avec une topologie fixe avec une distance de 10 mètres entre chaque capteur(en grille Fig. 4.2) avec 25, 60, 100, 150, 200, 250 et 300 capteurs (noeuds). Tous les noeuds sont des FFDs (Full Function Devices) comme vu précédemment, configurés comme des MicaZ. Nous testerons la couverture radio pour différents protocoles de routage : AODV, AOMDV, DSDV et DSR avec les trois types d'applications présentées précédemment. D'où un total de 336 simulations exécutées. En utilisant des scripts d'analyse, nous calculons notre critère de performance pour chacune de ces simulations.

4.2.2 Résultats numériques

Nous traçons les courbes de 4.4 à 4.9 correspondant à nos simulations en fonction du nombre de capteurs. Nos critères de performance correspondent d'un côté à la quantité de paquets perdus d'un point de vue applicatif et d'un autre côté au pourcentage d'énergie restant du réseau. Ce critère nous permet donc de comparer les performances des différents protocoles de routage en fonction des différentes applications et de la couverture radio.



(a) AODV (b) AOMDV

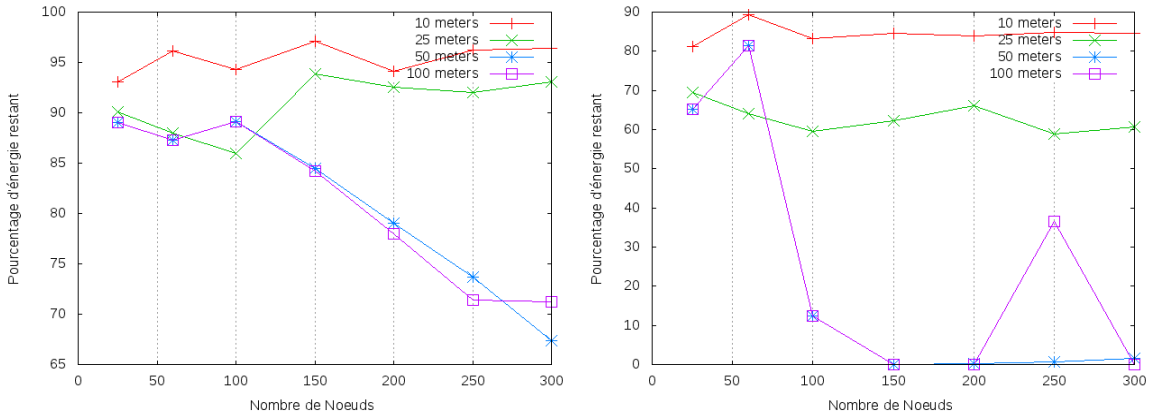
(c) DSDV (d) DSR

FIGURE 4.4 – Application régulière paquets perdus

Pour une application régulière (Fig. 4.4) avec le protocole de routage AODV, on obtient pour 10 et 100 mètres de couverture aucune perte de paquets. Néanmoins pour 25 mètres, nous observons dans la plupart des cas une perte de paquets variant de 0 à une vingtaine de paquets perdus. Pour 50 mètres, mis à part 200 noeuds où nous observons une perte conséquente des paquets applicatifs, pour tous les autres nous n'avons aucune perte. Avec une couverture de 25 mètres, les noeuds se retrouvent dans la configuration du noeud caché. Ceci provoque dans certains cas des collisions Fig. 2.2 [106].

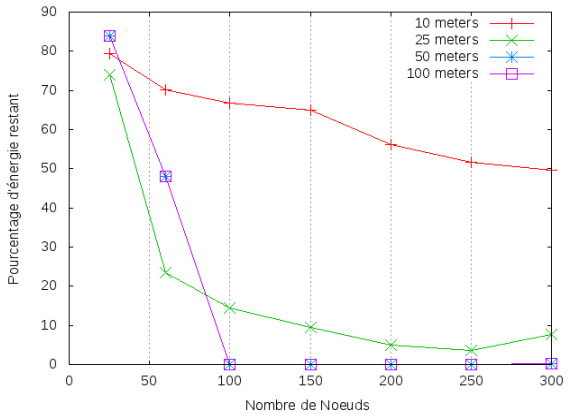
Concernant le protocole AOMDV, quelque soit la couverture radio, nous n'avons aucun paquet perdu. Ceci est le résultat de l'utilisation de plusieurs routes pour atteindre la destination, ce qui évite les collisions et réduit considérablement le nombre de paquets perdus dans le cadre d'une application avec un débit régulier.

Pour le protocole DSDV, on obtient sensiblement les mêmes résultats que le protocole AOMDV sauf pour 10 mètres avec 60 noeuds nous avons 1 paquet perdu. L'application étant régulière on a un réseau non surchargé ce qui permet au NPDUs, qui effectue une régulation des paquets de contrôle, de réduire la quantité de trafic sur le réseau. Le routage proactif permet de toujours assurer des routes fonctionnelles

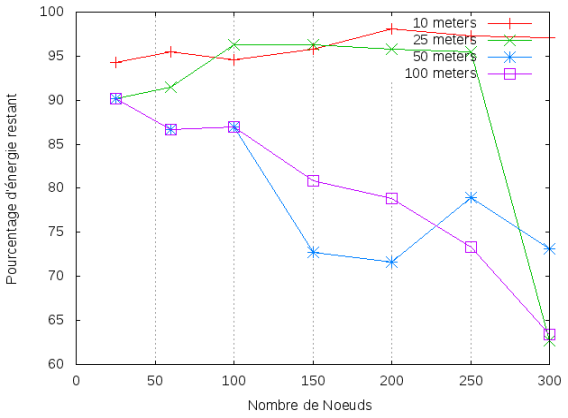


(a) AODV

(b) AOMDV



(c) DSDV



(d) DSR

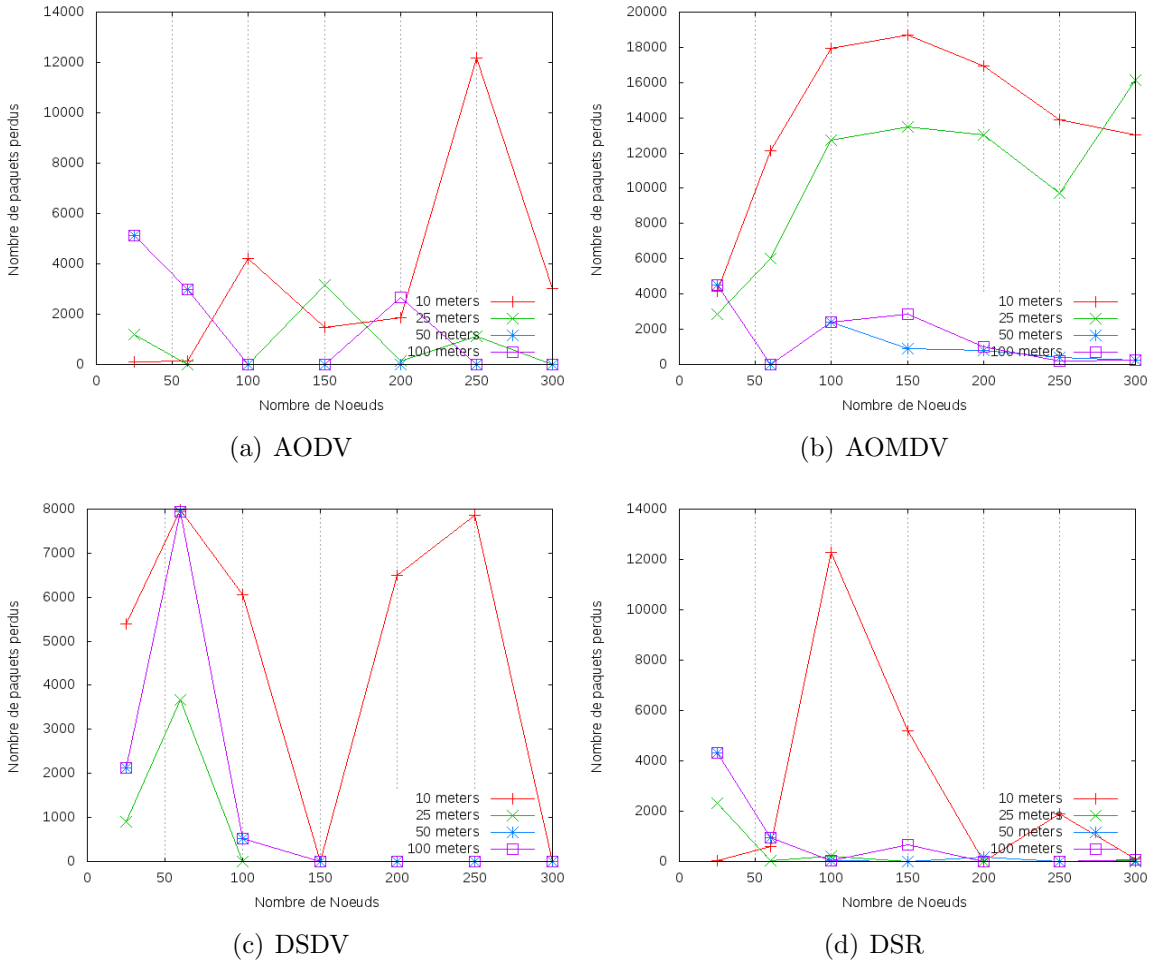
FIGURE 4.5 – Application régulière énergie

ce qui réduit la perte de paquets.

On observe avec le protocole DSR une perte légère de paquets pour 10 et 25 mètres avec 60, 100, 150 et 300 noeuds. Cette perte ne dépasse pas 6 paquets perdus ce qui correspond à une perte légère qui n'influe pas spécialement sur l'application, mais provoque des retransmissions intempestives.

Du point de vue de l'énergie (Fig. 4.5) pour le protocole AODV, on remarque que pour une couverture de 10 mètres, l'énergie conserve une certaine moyenne environ entre 94 et 96% d'énergie restant, ce qui représente les meilleurs résultats par rapport aux autres courbes. En effet, plus la couverture augmente plus le niveau d'énergie restant est bas. Pour 25 mètres, on remarque pour un nombre important de noeuds à partir de 150, le niveau d'énergie reste assez proche des résultats avec 10 mètres de couverture. Pour une couverture de 50 et 100 mètres, nous obtenons sensiblement les mêmes résultats, les niveaux d'énergie se dégradent dès que le nombre de noeuds augmente à partir de 150 noeuds. Pour 25, 60 et 100 noeuds, on obtient à peu près les mêmes résultats qu'avec 25 mètres de couverture.

Concernant le protocole AOMDV, on observe sensiblement les mêmes résultats que le protocole AODV. Pour 10 mètres de couverture, on observe une certaine



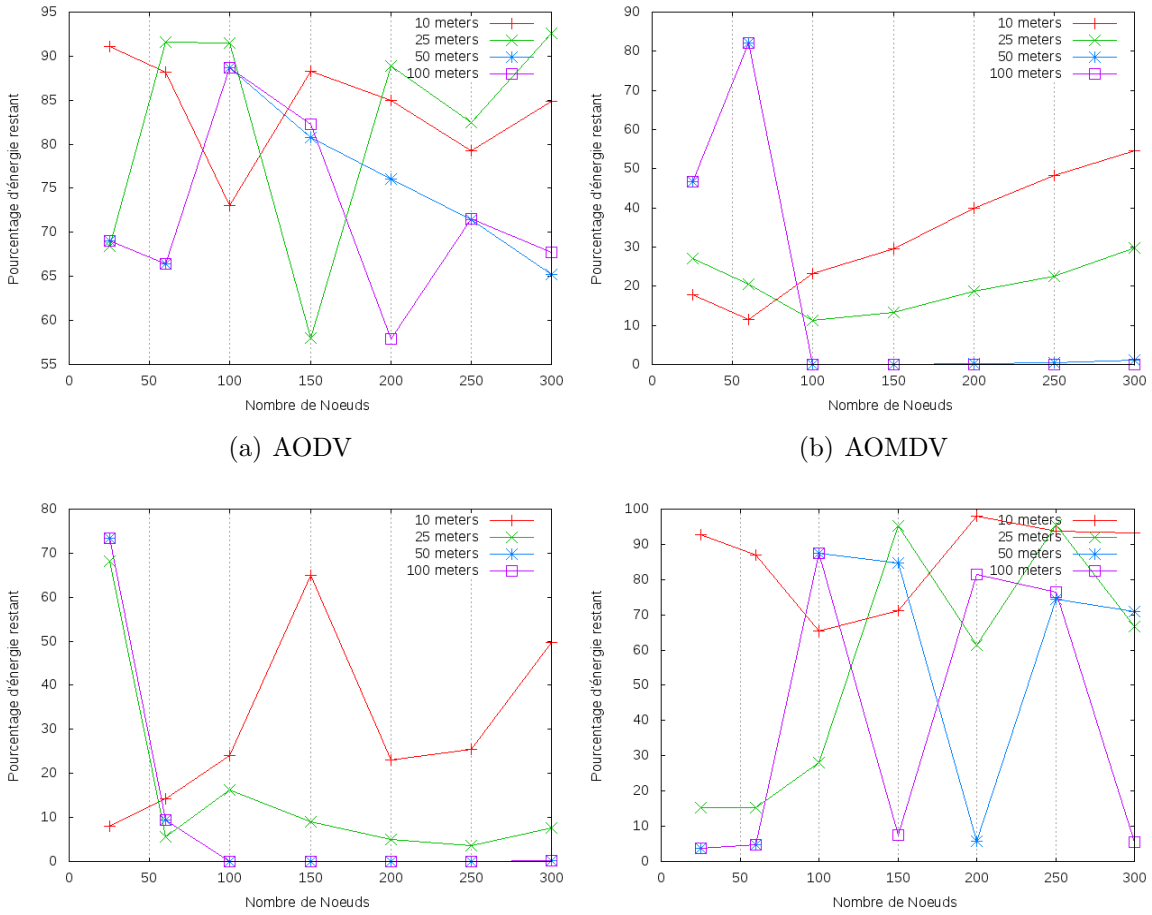
(a) AODV (b) AOMDV

(c) DSDV (d) DSR

FIGURE 4.6 – Application à fort débit paquets perdus

moyenne des résultats aux alentours de 85% d'énergie restant. Pour 25 mètres, on se retrouve à environ 60 à 70% d'énergie restant et pour une couverture de 50 et 100 mètres, à partir de 100 noeuds, on obtient des niveaux d'énergie très bas. Ceci est le résultat, comme pour AODV, de la perte de paquets au niveau de la couche MAC qui est plus importante quand la couverture augmente. En effet, lors des processus de découverte de route, et notamment les "brodcasts", les paquets retransmis déjà reçus sont supprimés au niveau MAC, car le rayon d'action étant plus grand, les noeuds reçoivent beaucoup de paquets ne leur étant pas destinés. Ceci augmente considérablement la quantité de paquets perdus et augmente la consommation d'énergie liée à la quantité de paquets reçus.

On observe sensiblement le même résultat avec le protocole DSDV mais, d'une part on peut noter que quand le nombre de noeuds augmente la quantité d'énergie restante diminue. D'autre part, on remarque une grande chute de l'énergie à partir de 60 noeuds pour une couverture supérieure à 25 mètres, où on a environ 10% d'énergie restant pour 25 mètres de 100 à 300 noeuds. De même, pour 50 et 100 mètres de couverture, on observe à partir de 100 noeuds environ 0% d'énergie restant sur le réseau. L'augmentation du nombre de connexions et de noeuds produit une



(a) AODV (b) AOMDV

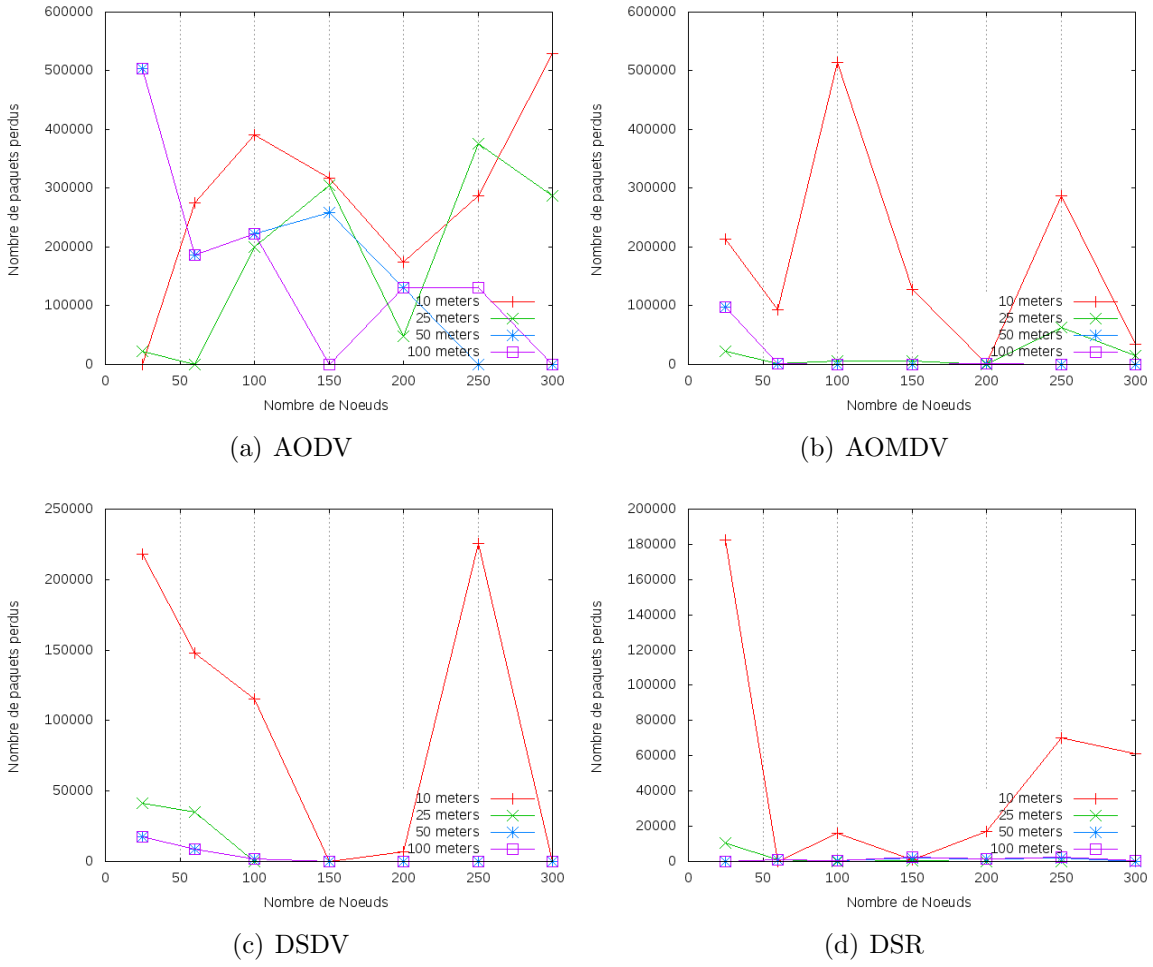
(c) DSDV (d) DSR

FIGURE 4.7 – Application à fort débit énergie

augmentation des délais de transmission et une surcharge du réseau par les paquets de routage, ce qui donne une consommation énergétique très importante.

Le protocole DSR quant à lui reste légèrement plus stable quand la couverture varie de 10 et 25 mètres. On observe sensiblement les mêmes résultats sauf pour 300 noeuds où nous avons une chute brutale de l'énergie pour 25 mètres de couverture. Pour 50 et 100 mètres, on observe les mêmes résultats pour 25, 60 et 100 noeuds puis les valeurs varient mais restent sensiblement les mêmes et surtout nettement en dessous de 10 et 25 mètres de couverture.

Ces résultats montrent pour une application régulière que le protocole AOMDV convient parfaitement pour réduire la perte de paquets (on a aucune perte). Avec le protocole DSDV, on remarque une unique perte avec 60 noeuds et pour AODV et DSR quelques pertes de paquets. D'un point de vue énergétique, on obtient une bonne stabilité pour AOMDV avec une couverture basse. On observe aussi un bon niveau énergétique restant pour AODV, qui est supérieur à 65% dans tous les cas, et DSR qui est supérieur à 60%. DSDV quant à lui perd rapidement trop d'énergie quand le nombre de noeuds et le nombre de connexions augmentent.



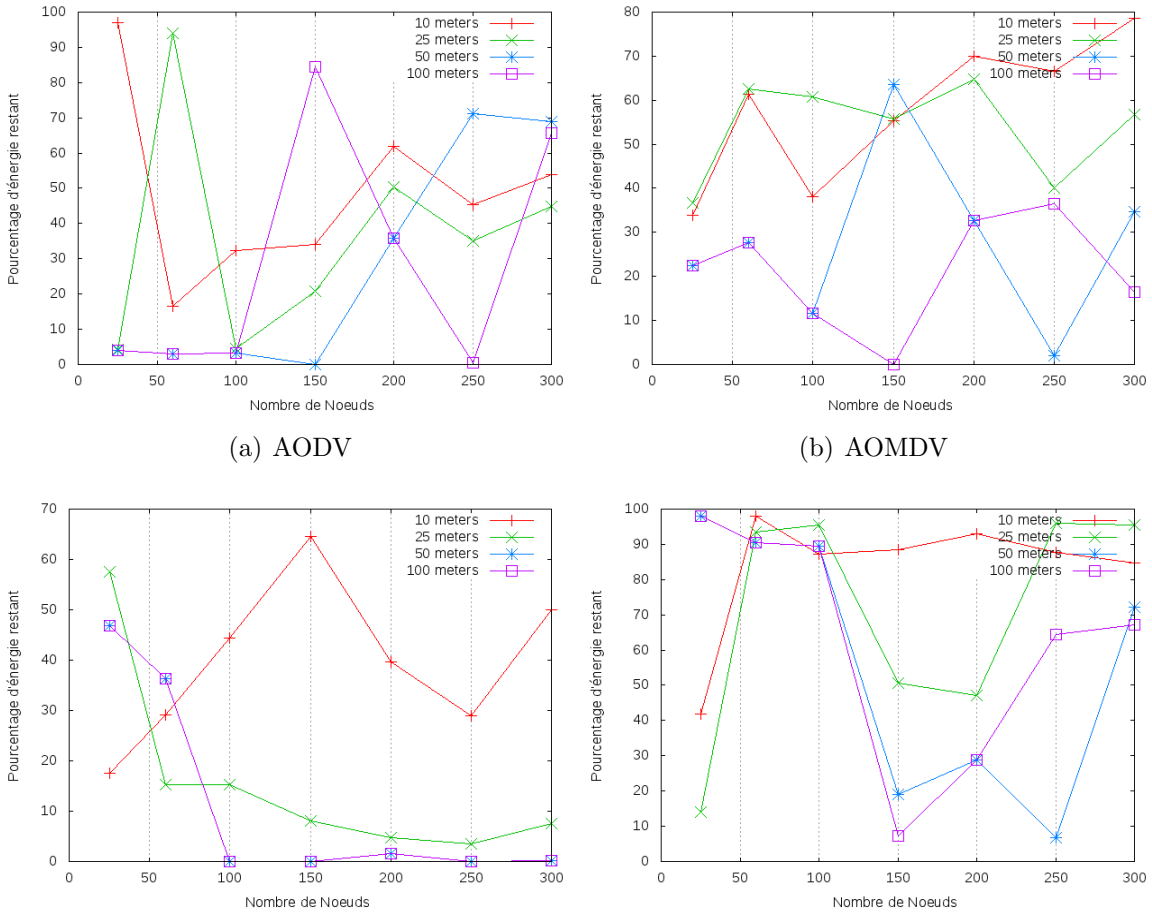
(a) AODV (b) AOMDV

(c) DSDV (d) DSR

FIGURE 4.8 – Application par à coup paquets perdus

Sur les figures 4.6 et 4.7 nous avons, concernant l'application avec un fort débit, les paquets perdus applicatifs et le pourcentage d'énergie restant. Tout d'abord concernant les paquets reçus (Fig. 4.6), pour le protocole AODV nous avons une disparité concernant les résultats. En effet, nous remarquons pour 25 et 60 noeuds que les couvertures de 10 mètres et 25 mètres obtiennent le plus faible nombre de paquets perdus. Pour 100 noeuds, nous observons sensiblement les mêmes résultats pour 25, 50 et 100 mètres de couverture. Pour 150 noeuds, il s'agit uniquement de 50 et 100 mètres de couverture. Pour 200 noeuds, nous avons 25 et 50 mètres et pour 250 et 300 noeuds, les meilleurs valeurs pour 25, 50 et 100 mètres. Ceci nous permet de dire que pour un nombre de noeuds faible, une couverture faible est suffisante, cependant pour un nombre de noeuds élevé il est nécessaire d'avoir une grande couverture radio pour réduire considérablement le nombre de paquets perdus pour le protocole AODV.

Pour le protocole AOMDV, nous remarquons plus aisément la tendance d'une couverture importante pour une application avec un fort débit. En effet, pour 25 noeuds les résultats sont sensiblement les mêmes pour une couverture de 10, 50 et 100 mètres, 25 mètres possédant une perte légèrement moins importante. Mais dès



(a) AODV (b) AOMDV

(c) DSDV (d) DSR

FIGURE 4.9 – Application par à coup énergie

que nous passons les 60 noeuds une couverture de 100 mètres ou 50 mètres (pour 150 noeuds) est privilégiée.

Avec le protocole DSDV, nous remarquons sensiblement les mêmes résultats que le protocole AODV. En effet, nous avons des résultats assez variés, tout d'abord pour 25 noeuds où on a la couverture de 25 mètres qui obtient le moins de paquets perdus puis nous avons 50 et 100 mètres ensuite 10. Pour 60 noeuds, on observe que les résultats sont identiques pour 10, 50 et 100 mètres, mais 25 mètres obtient toujours le moins de paquets perdus. A partir de 100 noeuds, nous observons à peu près les mêmes résultats pour une couverture de 25, 50 et 100 mètres qui se retrouvent assez proche de 0. Néanmoins la couverture de 10 mètres se rapproche sensiblement des autres pour un nombre de noeuds égal à 150 et 300 noeuds.

Le protocole DSR obtient des résultats sensiblement proche avec des couvertures de 25, 50 et 100 mètres, 10 mètres ici possède deux pics à 100 et 150 noeuds. Cependant pour 25 noeuds, on obtient le moins de paquets perdus pour une couverture de 10 mètres.

Concernant l'énergie (Fig. 4.7), nous observons pour le protocole AODV beaucoup de diversité en terme de résultats. Dans un premier temps, le niveau d'énergie

restant ne descend pas en dessous de 55% du niveau d'énergie. Nous remarquons une certaine similitude au niveau de la chute brutale de l'énergie pour chacune des couvertures. On la remarque à 100 noeuds pour 10 mètres de couverture, puis 150 noeuds pour 25 mètres de couverture et 200 noeuds pour 100 mètres de couverture. Néanmoins, nous avons le même constat qui est que, quand le nombre de noeuds augmente l'énergie baisse pour une couverture importante.

Pour le protocole AOMDV concernant l'énergie nous voyons nettement dans un premier temps que pour une large couverture radio et un grand nombre de noeuds, l'énergie restante est très proche de 0. De même, plus le nombre de noeuds augmente, plus on a une énergie restante qui augmente. Sauf pour 25 et 60 noeuds où nous avons des résultats légèrement différents.

Pour le protocole DSDV, on observe sensiblement la même chose qu'avec le protocole AOMDV où on a une énergie restante très faible pour un nombre de noeuds variant de 150 à 300 avec une couverture de 50 et 100 mètres. Ces résultats diffèrent avec 25 et 60 noeuds où on a un niveau d'énergie plus important avec 50 et 100 mètres pour 25 noeuds et pour 60 noeuds on a sensiblement les mêmes valeurs, mais avec un très léger avantage pour 10 mètres.

Avec le protocole DSR, on observe le même style et la même tendance de résultats que le protocole AODV où nous avons pour 25 et 60 noeuds une énergie restante importante pour 10 mètres puis pour 100 noeuds c'est 50 et 100 mètres, pour 150 noeuds il s'agit de 25 mètres puis les autres 200, 250 et 300 il s'agit de 10 mètres.

Nous pouvons dire pour une application avec un fort débit qu'il est nécessaire de privilégier une large couverture radio pour perdre le moins de paquets, néanmoins certains protocoles de routage consomment énormément d'énergie dans ce cas de figure. Le protocole AODV correspond à un bon compromis en terme de perte de paquets et consommation énergétique.

Dans le cadre d'une application par à coup (Fig. 4.8) pour le protocole AODV nous remarquons une grande irrégularité des résultats concernant la perte des paquets. En effet, nous observons sensiblement la même tendance qu'avec une application avec un fort débit. Pour 25 noeuds on obtient les meilleurs résultats pour 10 et 25 mètres de couverture. Pour 60 noeuds on a 25 mètres de couverture, pour 100 noeuds on obtient sensiblement les mêmes résultats pour 25, 50 et 100 mètres. Puis à partir de 150 noeuds, il s'agit des couvertures de 50 et 100 mètres qui obtiennent les meilleurs résultats sauf pour 200 noeuds où la courbe de 25 mètres se trouve légèrement en dessous des autres.

Pour les protocoles AOMDV, DSDV et DSR nous voyons clairement que la tendance est similaire à une application à fort débit. Néanmoins nous remarquons que les couvertures de 25, 50 et 100 mètres représentent les meilleurs résultats quelque soit le nombre de noeuds. Une couverture de 10 mètres provoque dans la plupart des cas un nombre important de paquets perdus ou un nombre quasi-similaire aux autres.

Concernant l'énergie sur la figure 4.9, on remarque pour le protocole AODV une irrégularité des résultats comme pour les paquets perdus, néanmoins nous observons une perte d'énergie plus importante pour les couvertures de 50 et 100 mètres.

Le protocole AOMDV nous montre également la même tendance où on a une

perte d'énergie plus importante avec une couverture de 50 et 100 mètres sauf pour 150 noeuds où on a environ 65% d'énergie restant pour 50 mètres de couverture.

Concernant le protocole DSDV, nous remarquons que, mis à part 10 mètres de couverture, pour 25, 50 et 100 mètres quand le nombre de noeuds augmente la quantité d'énergie diminue considérablement. Pour 10 mètres de couverture, nous avons une augmentation de l'énergie restant de 25 à 150 noeuds puis une chute de 150 à 250 noeuds, ensuite une augmentation pour 300 noeuds.

Pour le protocole DSR, nous remarquons que pour 25 noeuds, il s'agit des couvertures de 50 et 100 mètres qui possèdent la plus grande quantité d'énergie restante. Pour 60 et 100 noeuds, les résultats sont quasiment les mêmes pour les 4 couvertures. Pour 150 noeuds, nous remarquons une chute de l'énergie pour les couvertures de 25, 50 et 100 mètres, puis une irrégularité des résultats jusqu'à 300 noeuds. Mais les couvertures de 50 et 100 mètres obtiennent les plus mauvais résultats.

L'application par à coup provoque pour chaque configuration une irrégularité des résultats, mais il vaut mieux privilégier dans un premier temps une couverture importante pour réduire la quantité de paquets perdus. Dans un second temps le nombre de noeuds influe sur les performances donc il est judicieux, pour un nombre de noeuds faible, de privilégier le protocole DSR pour une consommation énergétique performante et une perte de paquets réduite. Et dans le cas d'un nombre plus important de noeuds, il est plus judicieux d'utiliser le protocole AOMDV pour la réduction de la perte des paquets et la régularité de la consommation d'énergie.

Pour résumer, on retrouve dans le tableau 4.10 les meilleures performances entre chaque protocole de routage.

	App. Régulière	App. par à coup	App. avec un fort débit
25 noeuds	DSDV	DSR	AODV
60 noeuds	DSDV	DSR	AODV
100 noeuds	AOMDV	DSR	AODV
150 noeuds	AOMDV	AOMDV	AODV
200 noeuds	AOMDV	AOMDV	AODV
250 noeuds	AOMDV	AOMDV	AODV
300 noeuds	AOMDV	AOMDV	AODV

FIGURE 4.10 – Meilleurs résultats obtenus en fonction des différents protocoles de routage

4.3 Impact de la topologie

4.3.1 Evaluation

Dans cette partie, nous allons effectuer notre évaluation pour montrer l'impact de la topologie sur le réseau de capteurs suivant les paramètres présentés précédemment en utilisant le simulateur NS-2. Pour cela, nous nous baserons sur les résultats obtenus précédemment pour les protocoles de routage en fonction du nombre de noeuds qu'on retrouve dans le tableau 4.10. Notre étude est toujours focalisée pour un

nombre de noeuds variant de 25 à 300 noeuds avec 3 topologies aléatoires et une topologie fixe sous forme de grille et pour les trois types d'application : régulière, avec un fort débit et par à coup. D'où un total de 84 simulations. En utilisant des scripts d'analyse, nous calculons notre critère de performance pour chacune de ces simulations.

4.3.2 Résultats numériques

Dans cette section, nous avons tracé les courbes correspondant à nos simulations. Nous traçons donc en fonction du nombre de capteurs notre critère de performance, c'est-à-dire la quantité de paquets perdus et le pourcentage d'énergie restant présentés précédemment. Ce critère permet donc de comparer l'efficacité des différentes topologies sur la fiabilité et la longévité du réseau. Celles-ci sont indispensables, car un réseau efficace en énergie n'est pas forcément fiable. De même un réseau fiable n'est pas forcément efficace en énergie d'où l'importance de trouver un juste milieu. Donc une combinaison du nombre de paquets perdus bas et une quantité d'énergie restant élevée assurent de bonnes performances pour le système.

Pour tracer les figures de 4.11 à 4.16, nous faisons varier les différentes topologies en fonction de chaque application (régulière, fort débit et par à coup).

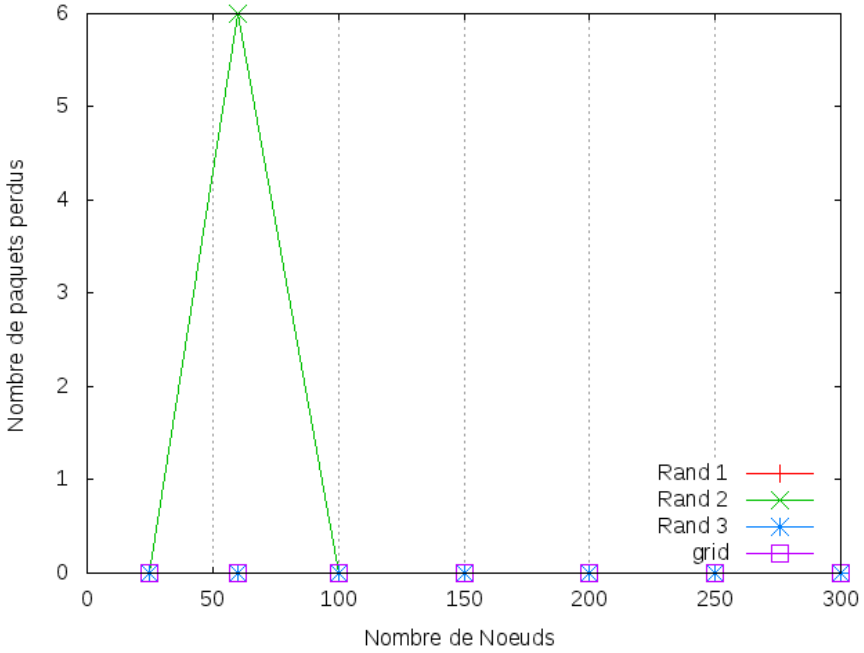


FIGURE 4.11 – Application régulière paquets perdus

Sur la figure 4.11, on peut constater que l'application régulière n'a pas de réel impact sur la topologie. En effet, mis à part la topologie aléatoire 2 pour 60 noeuds où nous avons 6 paquets perdus, les autres résultats sont identiques, c'est à dire aucune perte de paquets. D'un point de vue énergétique (Fig. 4.12), nous observons que les résultats sont sensiblement les mêmes, mis à part pour 100 noeuds, où la

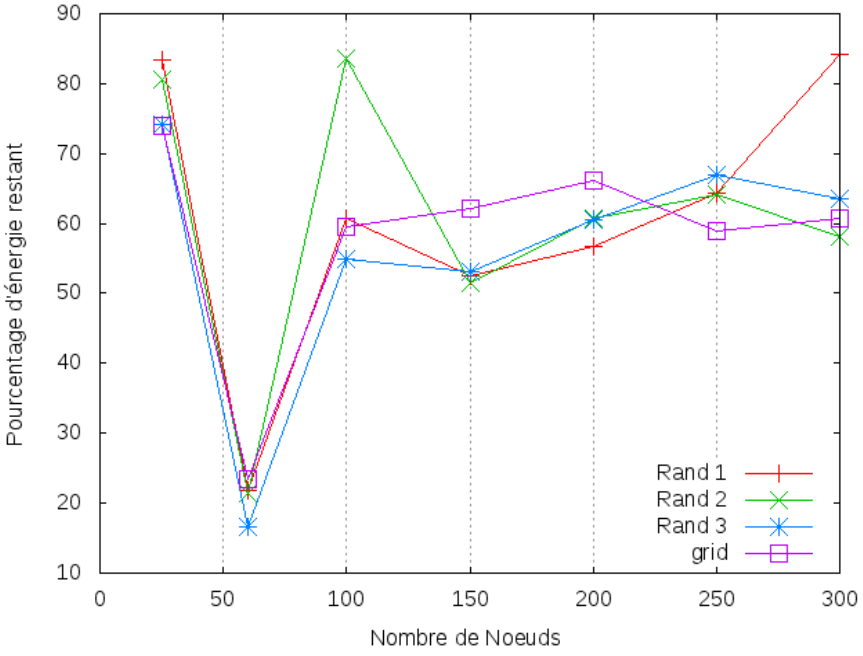


FIGURE 4.12 – Application régulière pourcentage d'énergie restant

topologie 2 possède une quantité d'énergie plus importante et pour 300 noeuds ou la topologie 1 obtient le même constat. Sinon dans la plupart des cas, l'écart d'énergie restant ne dépasse pas 10%.

Concernant une application avec un fort débit (Fig. 4.13), on remarque un impact certain pour quasiment toutes les valeurs. En effet, mis à part 25 noeuds où on a des résultats très proches voire quasi-identique, pour les autres valeurs nous obtenons des résultats très différents. Pour 60, 100 et 150 noeuds, nous obtenons le moins de perte avec une topologie sous forme de grille, et nous observons une irrégularité pour les topologies aléatoires. Pour 200, 250 et 300 noeuds, nous observons des résultats irréguliers et notamment la topologie sous forme de grille qui n'est pas la meilleure pour la perte de paquets. D'un point de vue énergétique (Fig. 4.14), nous observons une certaine similitude pour certains résultats. Pour 25 et 60 noeuds, on observe un écart de près de 25 à 30% entre chacune des topologies, la topologie en grille n'obtenant pas forcément les meilleurs résultats (25 noeuds où il est plus mauvais). A partir de 100 noeuds, nous observons un écart de 10 à 20% entre chacune des topologies, cet écart varie et ne possède aucune relation avec la quantité de paquets perdus.

Pour une application par à coup (Fig. 4.15), nous remarquons une irrégularité moins forte par rapport à une application avec un fort débit pour certaines valeurs. En effet pour 100, 150 et 300 noeuds, nous obtenons sensiblement les mêmes résultats. Pour 25 noeuds, on observe une grande disparité entre chacune des topologies et la topologie aléatoire 1 possède le plus grand nombre de paquets perdus. Pour 60 noeuds, il s'agit de la topologie 3 également pour 200 et 250 noeuds. Du

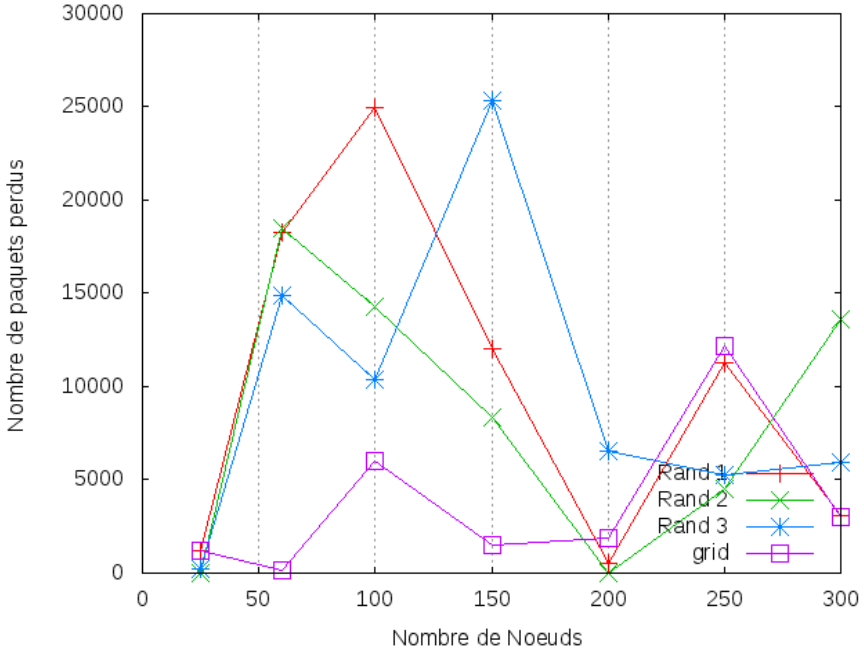


FIGURE 4.13 – Application à fort débit paquets perdus

point de vue de l'énergie (Fig. 4.16), nous observons une plus grande irrégularité des niveaux d'énergie restant que dans les autres applications notamment pour un nombre de noeuds de 25 avec un écart de près de 90%, 60 et 250 noeuds avec un écart de 40%, les autres possédant un écart d'environ 20%.

4.4 Conclusion

Dans cette partie, nous avons présenté et analysé l'impact de la couverture radio, de la topologie et du nombre de capteurs pour les réseaux de capteurs sans fil. Nous avons décrit notre approche qui consiste à faire varier plusieurs couvertures radio, protocoles de routage et topologies avec un certain nombre de noeuds sur différentes applications.

Nous avons utilisé quatre couvertures radios différentes, quatre protocoles de routage, trois topologies aléatoires, et une topologie fixe sous forme de grille, où les critères de performance les plus importants à prendre en compte sont la consommation d'énergie et la quantité de paquets perdus d'un point de vue applicatif. Nous avons effectué ces tests grâce au simulateur NS-2, ce qui nous a permis d'effectuer un grand nombre de simulations, en faisant varier le nombre de noeuds et le type d'application.

Ce chapitre fournit, dans un premier temps, l'étude de l'impact de la couverture radio pour le mode sans balises de l'IEEE 802.15.4, basée sur quelques protocoles de routage usuel utilisés dans les réseaux de capteurs. Nous avons présenté des résultats numériques sur les performances du réseau pour chaque cas. Nos observations générales des simulations nous ont montré que quand la couverture augmente les protocoles de routage perdent beaucoup plus d'énergie et pour chaque

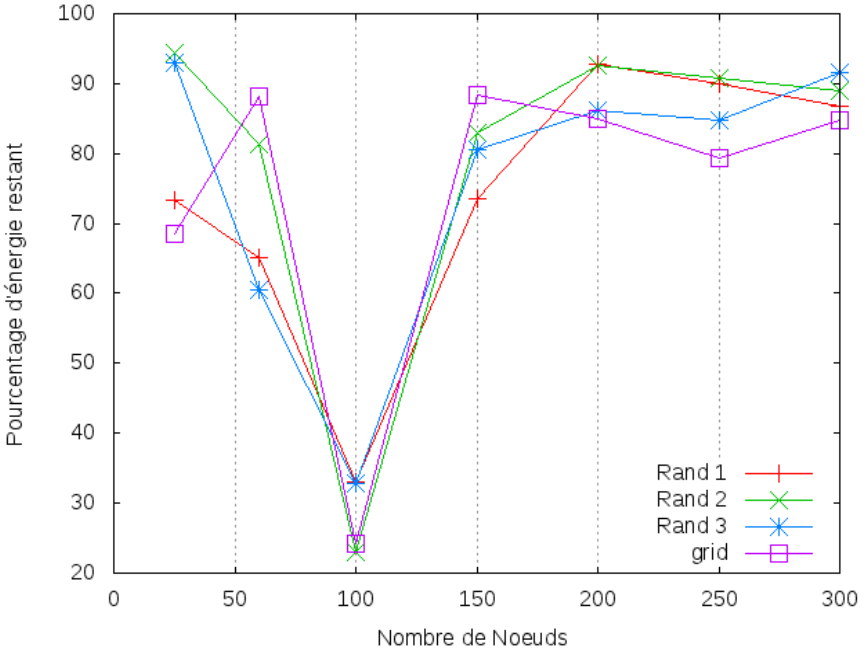


FIGURE 4.14 – Application à fort débit pourcentage d'énergie restant

type d'application permettent ou non la réduction du nombre de paquets perdus. Par ailleurs, AOMDV est le meilleur dans les situations stressantes (surcharge réseau et beaucoup de noeuds). Pour les autres situations, ceci dépend du type d'application, on retrouve concernant une application régulière, DSDV pour un nombre faible de noeuds et AOMDV pour un nombre plus important. Pour une application avec un fort débit on retrouve AODV, et pour une application par à-coup c'est AOMDV pour la régularité des résultats et DSR pour un nombre faible de noeuds.

Nous avons aussi remarqué un impact modéré sur les performances, mais le choix de la couverture doit être lié à la topologie utilisée. Une couverture radio forte est utile quand la zone d'intérêt est grande, mais une couverture faible est plus judicieuse quand la zone d'intérêt est réduite.

Dans un second temps, nous avons montré que la topologie a un impact modéré sur les performances du réseau en fonction du type d'application. En effet, pour une application régulière et avec un fort débit, on observe un impact faible voire modéré sur les performances du réseau notamment d'un point de vue énergétique. Néanmoins pour une application avec un fort débit on a parfois des résultats plus intéressants avec une topologie aléatoire qu'avec une grille fixe. Pour une application par à coup l'impact reste présent, compte tenu des écarts des niveaux d'énergie en fin de simulation qui peuvent atteindre près de 90%.

Ces résultats représentent une première contribution dans le but d'effectuer notre classification pour sélectionner les paramètres de configuration optimaux pour les réseaux de capteurs.

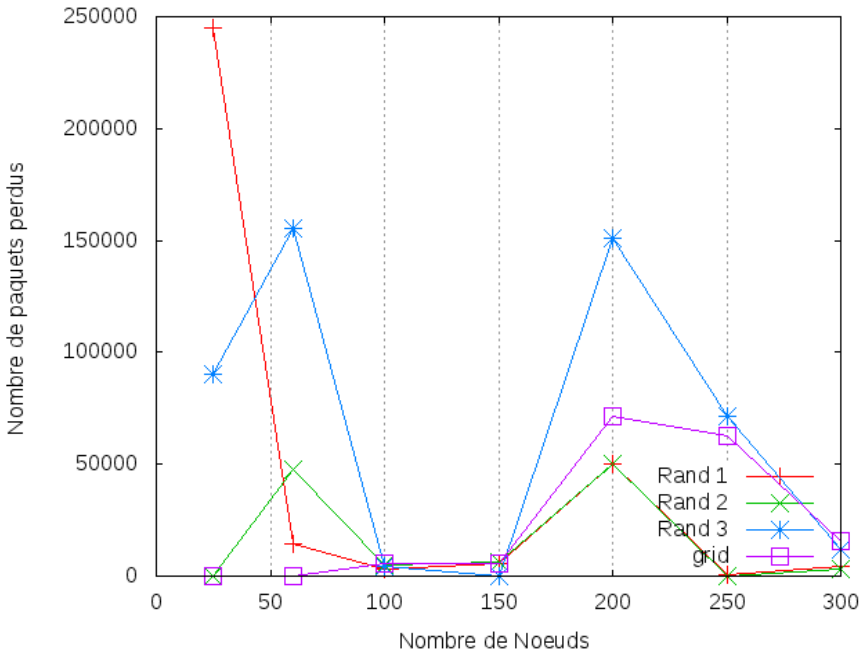


FIGURE 4.15 – Application par à coup paquets perdus

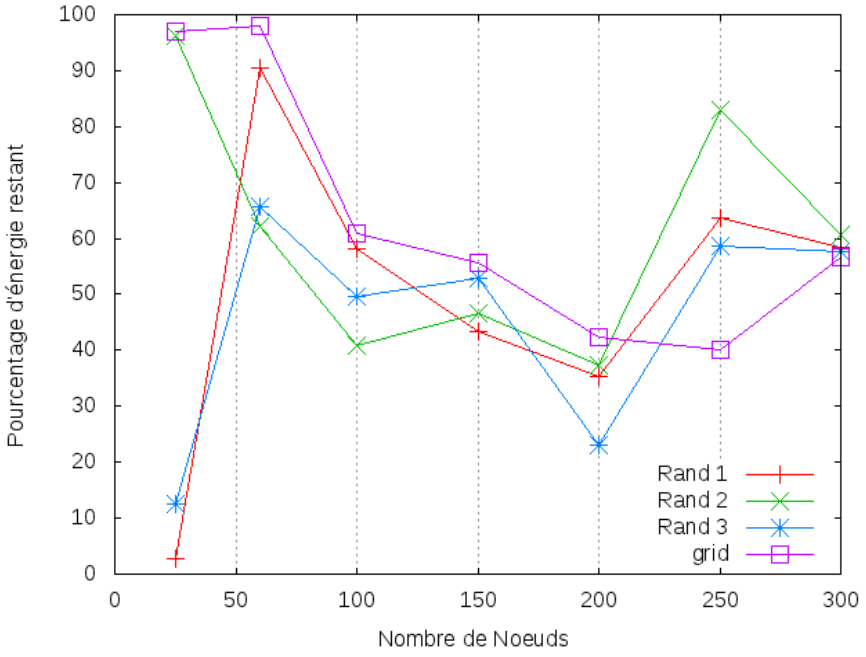


FIGURE 4.16 – Application par à coup pourcentage d'énergie restant

Chapitre 5

Classification des différents protocoles

Sommaire

La conception de protocoles de communications est un domaine de recherche à part entière. De nombreux outils et méthodes ont été développés et apportent des aides évidentes aux chercheurs et industriels lors de conception de nouveaux produits. Suivant les compétences des acteurs et leurs origines, on utilise des outils de spécification et de validation formelle [16], ou plutôt des outils de simulation de réseaux et de protocoles [76], voire du prototypage réel [92]. Chaque technique apporte son lot d'avantages en fonction des besoins et du niveau d'abstraction utilisé. Plus les protocoles à développer et valider sont proches des couches basses, plus il semble naturel d'utiliser un niveau de granularité fin, proche du matériel et des caractéristiques intrinsèques des médias.

A la fin du chapitre précédent, nous avons présenté une approche d'évaluation pour déterminer l'impact de certains paramètres tels que : la topologie, la couverture radio et le nombre de noeuds sur les performances d'un réseau de capteurs. Pour cela nous avons effectué une étude des paquets perdus et de l'énergie consommée. Cependant notre objectif dans ce chapitre est de produire un modèle de classification permettant pour une situation de fonctionnement d'un réseau de capteurs : topologie, type d'application, environnement, etc. de produire les paramètres optimaux pour un fonctionnement fiable et une longue durée de vie. Notre modèle est une base

nous permettant d'obtenir à partir des différents paramètres les taux de perte et niveau d'énergie en simulation ce modèle évolue au fil du temps, car nous pouvons ajouter d'autres paramètres, ce qui nous permettrait d'être plus précis dans certaines situations telles un outil d'aide à la décision. Nous avons alors décidé d'utiliser de façon ciblée et complémentaire plusieurs protocoles pour correspondre dans la plupart des cas à une utilisation courante dans les domaines d'utilisation des réseaux de capteurs. Le contenu de ce chapitre reprend essentiellement celui de la publication [89].

Le chapitre est organisé comme suit : la section 5.1 décrit nos choix concernant les différents protocoles pour réaliser notre classification. Puis dans la section 5.2 nous présentons notre étude avec nos simulations et l'analyse des résultats. Enfin la section 5.3 conclut le chapitre.

5.1 Choix des protocoles

5.1.1 Accès au médium

Pour réaliser notre classification, nous avons fait le choix de travailler avec la technologie de communication sans fil courte portée IEEE 802.15.4 qui propose une norme de communication bas débit pour les réseaux de capteurs. Outre les caractéristiques de base, la spécification ZigBee propose une pile protocolaire propriétaire et légère qui est déclinable dans plusieurs versions. Grâce à l'optimisation des périodes de mise en veille, du matériel ZigBee réalise de fortes économies d'énergie et se retrouve utilisé dans la plupart des capteurs que l'on retrouve dans le monde de la recherche.

Pour rappel, la norme IEEE 802.15.4 prévoit deux modes complémentaires pour l'accès au médium radio : un mode sans contention, de type *Best Effort*, par utilisation du protocole CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance* et un mode avec contention où il y a synchronisation entre les noeuds et une durée fixe pour émettre des données. Dans ce mode les dispositifs possèdent deux phases : une phase active où les noeuds peuvent communiquer et une phase inactive où les noeuds éteignent leur radio. La qualité de service se retrouve donc garantie par le coordinateur du réseau, qui est le dispositif chargé de répartir les accès au médium des autres dispositifs dans le réseau. Il détermine les durées des périodes actives et inactives, et d'autres informations relatives à la distribution des temps de parole.

5.1.2 Protocoles de routage

Le protocole de routage permet d'acheminer les données d'une source à un destinataire à travers le réseau. Un bon protocole de routage doit acheminer les données : avec une faible latence, en consommant peu d'énergie, en minimisant la charge du réseau et être fiable. Nous supposons que notre réseau de capteurs fonctionne dans la classe des réseaux ad hoc que dans la classe des réseaux de diffusion. Dans un réseau de diffusion, le principe est basé sur le routage de l'information vers le puits ce qui ne correspond pas suffisamment à un cas général de notre classification. Tandis que

dans la classe des réseaux ad hoc les éléments du réseau doivent être en mesure de s'autoorganiser pour pouvoir communiquer. Tous les éléments du réseau assurent donc la fonction de routage de l'information au sein du réseau et peuvent être des sources ou des destinataires.

Dans cette catégorie nous nous sommes basés sur deux types de protocole qu'on retrouve dans les réseaux ad hoc : les protocoles de routage pro-actifs et réactifs. Dans le cadre des protocoles de routage pro-actifs, chaque élément du réseau cherche à établir des tables de routage valides en permanence (vue en section 3.2). Nous avons privilégié le protocole DSDV pour son utilisation courante. Dans le cadre des protocoles de routage réactif, les routes sont déterminées uniquement au moment où une transmission de données doit être réalisée. Nous avons donc privilégié les protocoles AODV, AOMDV et DSR car ils représentent des protocoles usuels.

Pour résumer le fonctionnement de chacun on a : DSDV qui conserve les routes en permanence, AODV qui construit la route à la demande, AOMDV qui construit toutes les routes vers une destination pour une source en se basant sur AODV et DSR qui construit la route de la destination à la source en possédant à l'inverse des autres toutes les routes intermédiaires entre la source et la destination.

5.1.3 Applications

Concernant l'application nous nous basons sur le modèle présenté dans la sous-section 4.1.1 que nous allons rappeler. En supposant qu'une application détermine un débit d'utilisation des communications on peut classer les applications en types distincts :

1. Une application dite régulière qui caractérise une application qui envoie des données avec un grand intervalle. Ceci peut représenter le cas d'une surveillance du niveau de l'eau d'une rivière toutes les heures.
2. Une application dite avec un fort débit ou les noeuds envoie de manière constante des données à travers le réseau, dans ce cas on obtient un réseau à la limite de la surcharge. Comme exemple on peut prendre les capteurs qui véhiculent une vidéo en "streaming" des caméras de surveillance d'une route.
3. Une application dite par à coup qui représente les applications transmettant beaucoup de données, mais à un intervalle de temps régulier. Comme exemple, on a le cas des capteurs permettant de compter les oiseaux [47] et transmettant 2 heures de fichier audio toutes les 2 heures.

5.2 Modélisation des différents paramètres

Dans cette partie, nous allons présenter notre approche pour la classification. Pour arriver au but de notre outil d'aide à la décision des paramètres de configuration, il est important de passer par différentes phases : une phase d'apprentissage, une phase de classification, production de l'outil et évolution de l'outil pour s'adapter dans n'importe quelle situation. La phase d'apprentissage consiste à une phase de simulation des paramètres de base présentés précédemment en utilisant l'outil NS-2.

Puis la classification, elle se fera en fonction des résultats obtenus ou nous choisirons les meilleurs paramètres pour chaque situation que nous avons réalisés. Par la suite l'outil produira aux utilisateurs pour quel que soit la situation les paramètres adéquats de configuration. Notre critère de performance évalué est le même utilisé pour l'étude de l'impact de la couverture et la topologie, qui correspond aux taux de perte de paquets applicatifs et du pourcentage du niveau restant de l'énergie. Ces critères permettront à l'utilisateur durant la phase d'apprentissage et d'évolution de l'outil d'intégrer ses propres résultats dans l'outil pour d'autres paramètres de configuration récoltés depuis un test en environnement réel ou en simulation par un simulateur de réseaux. Nous allons détailler tout d'abord nos scénarios de simulation, puis montrer les résultats sous forme de courbe et les analyser pour en tirer la classification.

5.2.1 Scénarios de simulations

Dans nos simulations nous faisons varier pour un nombre de noeuds, les paramètres de la couche MAC, les protocoles de routage et le type d'application présentée précédemment. Pour la couche MAC, nous utilisons le mode sans balises et avec balises. Dans le second cas, il existe deux valeurs (BO et SO) qui permettent de déterminer la durée active et inactive des noeuds présentés dans la section 3.1 où la figure 3.4 résume le principe.

Exemple d'un fichier de simulation

```
# ======================
# Options de base
# ======================

set val(chan)    Channel/WirelessChannel;
set val(prop)    Propagation/TwoRayGround;
set val(netif)   Phy/WirelessPhy/802_15_4
set val(mac)     Mac/802_15_4
set val(ifq)     Queue/DropTail/PriQueue
set val(ll)      LL;
set val(ant)     Antenna/OmniAntenna;
set val(en)      EnergyModel
set val(ifqlen)  50;
set val(nn)      25;
set val(rp)      AODV;
set val(x) 80;
set val(y) 80;
set val(initenergy) 2.5;
set val(totalenergie) [expr $val(nn) * $val(initenergy)]

proc getopt {argc argv} {
        global val
        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue
                set name [string range $arg 1 end]
                set val($name) [lindex $argv [expr $i+1]]
                puts "val($name): $val($name)"
        }
}

getopt $argc $argv

set val(tr) Trace/$val(NB).tr;
set val(nam) Trace/$val(NB).nam;
set starTime     $val(starTime)
set stopTime     $val(stopTime)

# ============================
# Initialisation variables globales
# ============================

set ns_ [new Simulator]
set tracefd [open $val(tr) w]
#$ns_ use-newtrace
$ns_ trace-all $tracefd
set namtrace [open $val(nam) w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
$ns_ puts-nam-traceall {# nam4wpan #} ;
Mac/802_15_4 wpanCmd verbose on
Mac/802_15_4 wpanNam namStatus on ;


# Valeur pour les couvertures Radio
```

```
set dist(5m)  7.69113e-06
set dist(9m)  2.37381e-06
set dist(10m) 1.92278e-06
set dist(11m) 1.58908e-06
set dist(12m) 1.33527e-06
set dist(13m) 1.13774e-06
set dist(14m) 9.81011e-07
set dist(15m) 8.54570e-07
set dist(16m) 7.51087e-07
set dist(20m) 4.80696e-07
set dist(25m) 3.07645e-07
set dist(30m) 2.13643e-07
set dist(35m) 1.56962e-07
set dist(40m) 1.20174e-07
set dist(50m) 1.11548e-08
set dist(100m) 2.78869e-09

# ======================================
# Configuration Couche Physique
# ======================================

Phy/WirelessPhy set CSThresh_ $dist(25m)
Phy/WirelessPhy set RXThresh_ $dist(25m)
Phy/WirelessPhy set Pt_ 0.281838
Phy/WirelessPhy set freq_ 2.4e09
Phy/WirelessPhy set L_ 1.0
Phy/WirelessPhy set lambda_ 0.125
Phy/WirelessPhy set bandwidth_ 250*10e3;

# ======================================
# Parametres Antenne
# ======================================

Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0


#Taille de la topologie
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
set chan_1_ [new $val(chan)]

# ================================
# Configuration des Noeuds
# ================================

$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
```

```
                    -ifqLen $val(ifqlen) \
111                 -antType $val(ant) \
                    -propType $val(prop) \
113                 -phyType $val(netif) \
                    -topoInstance $topo \
115                 -agentTrace ON \
                    -routerTrace OFF \
117                 -macTrace ON \
                    -movementTrace OFF \
119                 -energyModel $val(en) \
                    -rxPower 0.0831 \
121                 -txPower 0.0762 \
                     -sleepPower 0.000048 \
123                  -transitionPower 0.02406 \
                     -transitionTime 0.002 \
125                 -initialEnergy $val(initenergy) \
                    -channel $chan_1_
127

129 # ==========================
    # Creation des noeuds
131 # ==========================

133 for {set i 0} {$i < $val(nn) } {incr i} {
      set node_($i) [$ns_ node]
135   $node_($i) random-motion 0 ;
    }
137

    # ================================
139 # Repartition aleatoire des nodes
    # ================================
141

    # Ou utiliser un fichier source pour des positions fixes
143 source Topo.scn

145 # ================================
    # Processus des Applications
147 # ================================

149 set sink_(0) [new Agent/LossMonitor]
    eval $ns_ attach-agent \$node_(0) \$sink_(0)
151

    proc cbr_basestation { src interval starttime } {
153    global ns_ node_ sink_ cbr_
       set udp_($src) [new Agent/UDP]
155    eval $ns_ attach-agent \$node_($src) \$udp_($src)
       set cbr_($src) [new Application/Traffic/CBR]
157    eval \$cbr_($src) set packetSize_ 32
       eval \$cbr_($src) set interval_ $interval
159    eval \$cbr_($src) set random_ 0
       eval \$cbr_($src) attach-agent \$udp_($src)
161    eval $ns_ connect \$udp_($src) \$sink_(0)
       $ns_ at $starttime "$cbr_($src) start"
163 }
```

```
165 proc poisson_basestation { src interval starttime } {
    global ns_ node_ sink_
167     set udp($src) [new Agent/UDP]
    eval $ns_ attach-agent \$node_($src) \$udp($src)
169     set expl($src) [new Application/Traffic/Exponential]
    eval \$expl($src) set packetSize_ 32
171     eval \$expl($src) set burst_time_ [expr $interval]ms
    eval \$expl($src) set idle_time_ [expr $interval*5]ms;
173     eval \$expl($src) set rate_ 50k
    eval \$expl($src) attach-agent \$udp($src)
175     eval $ns_ connect \$udp($src) \$sink_(0)
    $ns_ at $starttime "$expl($src) start"
177  }

179 proc record_init { dst } {
    global sink_ f0_ f3_ val node_ timin_
181     set ns_ [Simulator instance]
    set time 1
183     set timin_ 0
    set energie 0
185     set f0_($dst) [open  "Result/packet_lost_$val(NB)_$dst.tr" w]
    set f3_($dst) [open "Result/energy_$val(NB)_$dst.tr" w]
187
    set nblost [$sink_($dst) set nlost_]
189     set now [$ns_ now]

191     for {set i 0} {$i < $val(nn) } {incr i} {
      set energie [expr $energie + [$node_($i) energy]]
193     }

195     set e [expr $energie / $val(totalenergie) * 100]

197     puts $f0_($dst) "$timin_ \t $nblost";
    puts $f3_($dst) "$timin_ \t $e";
199     set timin_ [expr $timin_ +1]
    $ns_ at [expr $now+$time] "record $dst"
201 }

203 proc record { dst } {
    global sink_ f0_ f3_ val node_ timin_
205     set ns_ [Simulator instance]
    set time 1
207     set energie 0

209     set nblost [$sink_($dst) set nlost_]
    set now [$ns_ now]
211
    for {set i 0} {$i < $val(nn) } {incr i} {
213       set energie [expr $energie + [$node_($i) energy]]
    }
215
    set e [expr $energie / $val(totalenergie) * 100]
217
    puts $f0_($dst) "$timin_ \t $nblost";
219     puts $f3_($dst) "$timin_ \t $e";
```

```
    set timin_ [expr $timin_ +1]
    $ns_ at [expr $now+$time] "record $dst"
}

$ns_ at 0.0     "$node_(0) NodeLabel PAN Coor"
$ns_ at 0.0     "$node_(0) sscs startCTPANCoord 1 $val(BO) $val(SO)"

for {set k 1} {$k < $val(nn) } {incr k} {
  $ns_ at [expr $k+$val(interval_sync)] "$node_($k) sscs startCTDevice
      1 1 1 $val(BO) $val(SO)"
}

cbr_basestation 16 2 [expr $starTime]
cbr_basestation 9 2 [expr $starTime + 0.2]
cbr_basestation 13 2 [expr $starTime + 0.4]
cbr_basestation 10 2 [expr $starTime + 0.6]
cbr_basestation 14 2 [expr $starTime + 0.8]
cbr_basestation 11 2 [expr $starTime + 1]
cbr_basestation 15 2 [expr $starTime + 1.2]
cbr_basestation 12 2 [expr $starTime + 1.4]

# ===================================
# Taille des noeuds dans NAM
# ===================================
 for {set i 0} {$i < $val(nn)} {incr i} {
 $ns_ initial_node_pos $node_($i) 2
 }

for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $stopTime "$node_($i) reset";
}

$ns_ at $stopTime "stop"
$ns_ at $stopTime "puts \"\nNS EXITING...\n\""
$ns_ at $stopTime "$ns_ halt"

proc stop {} {
  global ns_ tracefd val env namtrace
  $ns_ flush-trace
  close $tracefd
  foreach index [array names f0_ f1_ f2_] {
    close $f0_($index)
    close $f1_($index)
    close $f2_($index)
  }
  set hasDISPLAY 0
    foreach index [array names env] {
        if { ("$index" == "DISPLAY") && ("$env($index)" != "") } {
                set hasDISPLAY 1
        }
    }
    if {("$hasDISPLAY" == "1")} {
      close $namtrace
    }
}
```

```
275 puts "\nStarting Simulation..."

277 $ns_ run
```

Exemple d'un fichier de simulation

Le fichier 5.2.1 montre un exemple de fichier de simulation où nous détaillons tous les paramètres nécessaires à la configuration du réseau de capteurs. Ici, les paramètres décrivent toutes les couches du modèle OSI de la couche Physique d'un noeud à la couche Application [73]. On retrouve dans notre fichier de la ligne 1 à la ligne 145 les paramètres de configuration des noeuds. Ces paramètres contiennent le nombre de noeuds, les paramètres physiques, liaison, réseau, topographique. On retrouve par ailleurs l'énergie initiale d'un noeud avec la quantité nécessaire pour émettre, recevoir et la consommation en veille d'un noeud. Dans le reste du fichier, on retrouve dans un premier temps les processus qui décrivent les applications, des scripts qui récupèrent les taux de perte et énergie et par la suite le scénario d'exécution de la simulation : les temps de démarrage des noeuds, le démarrage de l'application avec ses propriétés. Et pour finir on retrouve le processus d'arrêt de la simulation qui enregistre tous les détails de la simulation dans un fichier trace, et un autre fichier trace permettant une visualisation graphique dans NAM [76].

5.2.2 Résultats des simulations

Nous avons effectué notre étude en effectuant des simulations avec 25, 60, 100 et 300 noeuds, cependant nous obtenons énormément de résultats de ces simulations. Dans un premier temps nous avons effectué une moyenne de la consommation énergétique et du taux de perte de paquets puis intégré ces résultats sous forme de graphique, mais nous obtenions 72 graphiques que l'on peut retrouver en annexe 9. Cependant pour un utilisateur voulant avoir des informations bien précises il serait trop complexe pour lui d'analyser les graphiques c'est pour cela que nous proposons un outil permettant à un utilisateur en sélectionnant des paramètres d'obtenir sous forme de graphique les éléments dont il a besoin pour sa décision. On retrouve l'aspect de cette application sur la figure 5.14. Cependant nous proposons aussi un affichage d'une partie des résultats pour un type d'application et un nombre de noeuds. On retrouve dans cette partie un affichage du nombre de paquets perdus et du pourcentage d'énergie restant en fonction du paramètre de la couche MAC et du protocole de routage, ceci est visible sur la figure 5.15. Cette partie des résultats est stockée dans un fichier Excel dont nous allons vous montrer le contenu. Nous avons exécuté un total de 1872 simulations. Nous présenterons uniquement les résultats pour les protocoles de routage AODV, AOMDV, DSDV et DSR et pour les couches MAC sans balises et avec balises pour les valeurs $BO = 6, SO = 6; BO = 7, SO = 6; BO = 7, SO = 7; BO = 8, SO = 6; BO = 8, SO = 7; BO = 8, SO = 8; BO = 9, SO = 8; BO = 9, SO = 9;$. Dans ces résultats nous avons des périodes actives ("duty cycle") représentant 100%, 50% et 25% de la durée totale. Le calcul de BO pour chacun des pourcentages de la durée active est décrit dans la section 3.1, où pour 100% on a $BO = SO$, 50% $BO = SO - 1$ etc.

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	0	0	11	0	37	77	1	93	0
AOMDV	0	0	6	0	49	37	0	182	0
DSDV	0	1	10	0	128	52	0	187	2
DSR	0	0	17	0	416	119	0	77	0

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	93,371	92,191	92,027	93,999	92,287	93,345	91,762	93,939	92,989
AOMDV	86,98	87,248	85,704	86,567	85,692	86,986	88,7	85,184	88,346
DSDV	85,148	84,852	67,488	86,858	56,66	84,342	83,623	79,737	85,954
DSR	95,989	93,825	91,271	94,43	90,314	93,505	93,38	92,824	95,498

(b) Pourcentage d'énergie restant

FIGURE 5.1 – Application régulière avec 25 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	1	0	3	0	156	125	0	160	0
AOMDV	0	0	17	0	187	122	0	173	0
DSDV	2	1	67	1	253	162	1	219	5
DSR	0	0	35	0	710	182	0	223	0

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	95,387	95,587	93,792	93,787	90,967	94,232	94,857	92,143	94,714
AOMDV	88,908	85,245	86,349	87,116	84,706	85,874	88,495	84,641	88,481
DSDV	82,557	81,677	77,197	83,708	45,767	80,869	80,393	47,29	81,617
DSR	96,647	94,405	93,393	95,027	89,033	93,915	94,707	92,104	94,871

(b) Pourcentage d'énergie restant

FIGURE 5.2 – Application régulière avec 60 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	0	0	184	4	145	99	0	100	0
AOMDV	0	0	34	0	82	78	0	141	0
DSDV	1	1	25	1	61	140	2	142	31
DSR	0	0	239	0	895	160	0	524	0

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	95,064	94,119	91,701	94,933	88,29	94,277	95,351	91,168	96,097
AOMDV	88,676	86,813	85,679	87,328	82,9	84,381	87,419	83,796	88,012
DSDV	80,335	78,373	67,373	81,816	37,485	77,989	78,556	59,513	82,367
DSR	97,048	93,983	92,447	95,7	88,526	93,318	96,421	92,206	96,248

(b) Pourcentage d'énergie restant

FIGURE 5.3 – Application régulière avec 100 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	0	3	150	1	140	326	26	360	8
AOMDV	0	0	40	0	129	84	0	217	0
DSDV	0	4	63	30	313	161	7	383	0
DSR	0	0	669	0	646	740	88	408	0

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	94,28	92,26	93,003	93,734	74,296	87,88	96,196	76,763	96,509
AOMDV	73,527	85,789	88,397	86,963	82,16	87,545	90,09	83,377	90,813
DSDV	8,313	53,842	61,14	33,95	56,221	40,72	69,103	64,734	70,83
DSR	81,448	15,733	20,236	94,387	68,045	30,697	9,287	48,667	98,561

(b) Pourcentage d'énergie restant

FIGURE 5.4 – Application régulière avec 300 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	4610	5657	11568	5558	8562	9499	7283	7649	5538
AOMDV	5588	4675	6045	4726	15016	9255	5572	5816	4816
DSDV	3734	3754	4466	4218	11689	6972	5059	13597	5716
DSR	13330	29223	26805	28597	27341	28236	23433	26954	27648

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	58,795	29,294	33,863	35,162	68,376	58,155	33,942	67,13	30,49
AOMDV	34,505	31,088	27,948	33,019	37,305	29,918	31,191	48,502	32,405
DSDV	35,888	32,901	23,035	36,844	23,182	36,264	29,916	44,68	45,661
DSR	67,423	34,706	27,091	36,156	46,795	28,185	40,34	38,24	37,685

(b) Pourcentage d'énergie restant

FIGURE 5.5 – Application avec un fort débit avec 25 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	8603	9433	16093	9404	8138	16725	9196	7337	9217
AOMDV	15230	13258	11200	14111	11987	13252	13927	4217	15066
DSDV	10764	10571	8548	12806	6802	9975	11798	5976	11627
DSR	30824	40750	29819	44907	33035	42162	45034	32623	42727

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	63,408	39,262	39,696	33,548	73,296	55,46	37,891	73,319	36,056
AOMDV	27,464	22,347	24,062	24,113	43,852	26,296	28,39	58,78	29,28
DSDV	33,589	29,327	28,598	39,737	14,12	35,394	28,87	13,229	39,02
DSR	54,49	34,602	27,665	33,609	48,339	25,181	36,074	40,15	35,347

(b) Pourcentage d'énergie restant

FIGURE 5.6 – Application avec un fort débit avec 60 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	10818	6809	14572	11566	4223	10880	9055	3840	8482
AOMDV	14458	12516	11384	13704	8179	8350	14281	3157	14682
DSDV	8790	10809	5094	12160	5485	11162	12196	9868	5458
DSR	16201	41987	24544	36406	49945	28744	42710	38901	44750

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	74,777	39,189	51,372	45,157	75,634	67,346	58,941	76,864	34,793
AOMDV	30,778	34,102	22,363	26,384	53,13	35,618	26,902	66,499	21,393
DSDV	37,161	38,717	26,08	35,288	21,117	34,014	35,333	24,935	65,659
DSR	78,73	38,402	30,459	36,727	48,253	22,203	36,411	49,376	36,169

(b) Pourcentage d'énergie restant

FIGURE 5.7 – Application avec un fort débit avec 100 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	11956	1840	14158	1840	1533	10053	3393	5674	3577
AOMDV	11634	20878	20139	26020	6186	21028	26953	8227	26596
DSDV	2561	17816	25170	16896	17273	24934	16927	38479	18967
DSR	7744	7744	16415	7744	6682	7733	583	11867	7732

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	73,6	68,169	53,11	68,169	60,182	54,581	55,748	46,332	22,946
AOMDV	33,295	63,128	64,683	59,914	67,721	64,514	65,201	69,397	64,434
DSDV	7,803	37,625	47,853	17,924	47,771	27,708	54,485	57,228	53,619
DSR	95,864	10,507	14,183	10,507	62,352	32,521	9,74	41,065	6,894

(b) Pourcentage d'énergie restant

FIGURE 5.8 – Application avec un fort débit avec 300 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	137475	273988	171273	205799	336075	229547	282638	235483	218399
AOMDV	248490	216924	166775	276247	215487	289569	412212	135757	199200
DSDV	196810	367477	130745	157838	135641	360005	165486	193524	250750
DSR	592423	335552	269294	310012	425387	211033	379235	164148	439854

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	13,602	16,718	18,961	21,356	37,152	22,415	9,084	28,793	7,453
AOMDV	40,864	25,878	27,076	21,616	38,646	33,221	17,776	42,993	15,43
DSDV	23,841	29,819	18,464	36,56	26,065	31,713	23,171	32,486	26,831
DSR	52,437	38,178	27,447	39,863	51,656	36,222	38,735	52,297	41,569

(b) Pourcentage d'énergie restant

FIGURE 5.9 – Application par à coup avec 25 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	410617	234986	235993	279760	97806	424795	105768	106974	515724
AOMDV	400074	503717	272312	718885	191577	241390	546481	47973	445200
DSDV	442037	503318	197483	305203	46899	379053	441874	90363	284434
DSR	357737	625143	344854	369805	256571	501773	507655	182794	483162

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	23,234	33,865	32,551	32,211	70,475	42,008	30,033	48,458	27,935
AOMDV	53,186	36,96	27,074	28,17	58,839	47,948	28,048	59,839	25,78
DSDV	37,504	30,853	40,595	43	20,903	35,675	24,334	18,462	45,631
DSR	58,529	43,389	32,624	37,207	63,246	36,998	43,434	56,221	40,873

(b) Pourcentage d'énergie restant

FIGURE 5.10 – Application avec un fort débit avec 60 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	280594	356522	193693	725113	136195	248854	324252	65077	344811
AOMDV	161164	874515	315008	716619	112104	104902	342890	15898	886191
DSDV	559274	457166	420774	613998	87084	168827	701614	126158	389377
DSR	555598	423935	283409	405390	276006	307200	488054	203554	578554

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	36,408	38,151	32,608	27,919	72,94	52,267	37,19	68,49	33,93
AOMDV	61,894	31,595	34,502	29,189	65,787	54,48	35,661	70,273	30,091
DSDV	45,992	34,166	32,226	41,621	19,596	54,878	38,743	36,912	43,539
DSR	70,151	45,345	38,727	39,03	60,858	44,449	42,81	61,093	41,828

(b) Pourcentage d'énergie restant

FIGURE 5.11 – Application avec un fort débit avec 100 noeuds

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	260511	1164450	143200	1164680	70467	229361	847266	112218	2904264
AOMDV	125855	947106	537934	3123789	148461	335812	2636266	69463	3052546
DSDV	38802	1939831	499903	608946	116243	359212	2383976	152145	2488228
DSR	185	212161	76404	99471	49068	37865	136506	55360	74

(a) Paquets perdus

	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	27,472	65,489	79,161	64,072	79,476	80,339	67,389	83,416	62,209
AOMDV	60,618	61,478	67,681	57,758	78,953	71,161	56,48	78,147	56,965
DSDV	7,446	27,797	49,812	13,326	51,287	28,812	44,476	64,416	43,209
DSR	74,712	14,021	18,54	10,734	66,157	33,708	9,875	45,398	9,369

(b) Pourcentage d'énergie restant

FIGURE 5.12 – Application avec un fort débit avec 300 noeuds

5.2.3 Analyse et classification

Application régulière Concernant une application régulière comportant 25 noeuds (Fig. 9.1) nous remarquons qu'au niveau des paquets perdus une perte plus importante quand la période d'inactivité est 75% de la période globale où nous voyons DSR obtenir les plus mauvais résultats. Nous remarquons également quand il n'y a pas de période d'inactivité les protocoles AOMDV et DSR ne perdent aucun paquets, AODV ayant perdu un paquet et DSDV ayant perdu trois paquets. D'un point de vue de la perte des paquets, le protocole AOMDV en moyenne est le plus bas pour toutes les valeurs. Concernant l'énergie le protocole DSR et AODV sont les moins consommateurs en énergie, DSDV étant le plus mauvais et AOMDV se situant en troisième position.

Pour 60 noeuds (Fig. 9.4), nous observons toujours le même constat avec un avantage en moyenne pour les paquets perdus pour AOMDV et au niveau de l'énergie AODV et DSR, AODV étant en moyenne légèrement supérieur à DSR. DSDV quant à lui se retrouve très consommateur d'énergie et peut se révéler un mauvais choix dans certains cas.

Pour 100 noeuds (Fig. 9.7), il n'y a pas une grande différence pour les paquets perdus AOMDV ayant toujours une moyenne inférieure aux autres et DSR dès qu'il y une période d'inactivité obtient le plus grand nombre de paquets perdu. Pour l'énergie la tendance est revenue comme pour 25 noeuds c'est à dire que DSR possède le plus haut niveau d'énergie, AODV suivant de très près puis AOMDV et enfin DSDV.

Pour 300 noeuds (Fig. 9.10), AOMDV correspond toujours au meilleur choix pour la quantité de paquets perdus, DSR le plus mauvais choix dès qu'il y a de l'inactivité. Néanmoins du point de vue de l'énergie il s'agit d'AODV, puis d'AOMDV qui obtiennent les meilleurs résultats, DSR ayant le plus bas niveau d'énergie en moyenne. Ceci nous montre que dans un premier temps AOMDV peut représenter un atout considérable dès que le nombre de noeuds est très grand. D'autre part la technique de routage source utilisée par DSR se retrouve dans tous les cas pas suffisamment efficace pour réduire la perte de paquets quand les noeuds entrent en phase d'inactivité. Ceci est le résultat sans doute de la complexité du réseau à communiquer. De plus durant les périodes d'inactivité, les paquets générés se retrouvent en mémoire tampon et sont transférés au début de la période d'activité en un coup, ce qui provoque à ce moment une augmentation des collisions et donc implique une fiabilité des résultats réduite à cause des délais de transmission.

Application avec un fort débit Pour une application avec un fort débit comportant 25 noeuds (Fig. 5.5), nous remarquons que le protocole DSDV obtient en moyenne le moins de paquets perdus suivis du protocole AOMDV puis AODV et loin derrière DSR. Concernant l'énergie, AODV possède le ratio le plus élevé, mais la différence n'excède pas 10% avec DSDV et AOMDV. Nous remarquons également que DSDV est le seul pour qui les périodes d'inactivité ne permettent pas de réduire la consommation énergétique, AODV étant le seul à bénéficier d'un gain important (passage de 33% à 68%).

Pour 60 noeuds (Fig. 5.6), comme pour 25 noeuds, DSDV se retrouve légèrement

en avance sur AODV et AOMDV sur la quantité de paquets perdus. Par contre en terme d'énergie il se retrouve en dernière position là où AODV obtient de meilleurs résultats. Une fois de plus le gain accumulé pendant les périodes d'inactivité pour le protocole AODV reste vraiment très élevé.

Pour 100 noeuds (Fig. 5.7), AODV se retrouve légèrement devant DSDV en terme de perte de paquets et ceci notamment avec de grandes périodes inactives. DSR, lui obtient de très mauvais résultats comparés aux autres. Pour l'énergie, AODV gère mieux l'énergie que tous les autres il se retrouve largement devant surtout pendant les périodes inactives.

Pour 300 noeuds (Fig. 5.8), AODV reste toujours devant en terme de perte de paquets, ensuite on retrouve DSR puis AOMDV et enfin DSDV. Cependant en terme d'énergie AOMDV se retrouve devant AODV avec une énergie restante d'environ 61% contre 55% après on retrouve DSDV et DSR avec les plus mauvais résultats. Ceci prouve une fois de plus que pour une application avec un fort débit il est plus judicieux de prendre le protocole AODV pour une meilleure fiabilité des résultats et une consommation énergétique régulière.

Application par à coup Pour une application par à coup avec 25 noeuds (Fig. 9.3), nous observons dans un premier temps une moyenne du taux de perte très proche pour DSDV, AODV et AOMDV. Les résultats montrent un avantage pour des périodes où on a les noeuds toujours allumés et pour des périodes où la durée active est égale à la durée d'inactivité. Concernant l'énergie le protocole DSR possède une bonne maîtrise de l'énergie malgré une perte élevée de paquets dans la plupart des cas. Juste après DSR, le protocole AOMDV est celui qui possède des résultats très moyens mais acceptables dans certains cas.

Pour 60 noeuds (Fig. 9.6), on remarque que le protocole AODV possède les meilleurs résultats notamment pour des périodes actives de 25% et 50%, on retrouve d'ailleurs ces résultats dans chacun des protocoles de routage. Concernant l'énergie, DSR possède un niveau plus élevé que les autres, mais quand on a 25% d'activité ($BO = 8, SO = 6$) AODV obtient le meilleur niveau avec plus de 70% d'énergie restants.

Pour 100 noeuds (Fig. 9.9), AODV possède le moins de paquets perdus, mais ceci varie très rapidement en fonction du paramètre de la couche mac. En effet le constat est toujours le même pour 25% et 50% d'activité, mais le reste est très irrégulier. L'énergie montre une valeur toujours élevée pour DSR (49% en moyenne), mais AOMDV se retrouve pas très loin derrière avec 45% d'énergie restant.

Pour 300 noeuds (Fig. 9.12), DSR et AODV se partage la première place pour le taux de perte, mais avec un léger avantage pour DSR. La tendance est toujours la même pour des périodes de 25 et 50% d'activité. Concernant l'énergie le niveau d'énergie est meilleur avec le protocole AODV pour toutes les valeurs puis AOMDV qui n'est pas très loin et en dernière position DSR ne lui restant pas beaucoup d'énergie sur la fin. Ces résultats nous montrent que l'application avec un fort débit varie beaucoup quand on passe du mode avec balise et sans balises.

5.2.4 L'outil d'aide à la décision

Pour permettre une meilleure vue des résultats en fonction des simulations nous avons créé un outil qui nous permet de visualiser l'évolution du taux de perte ainsi que le pourcentage du niveau d'énergie restant au cours de la simulation. Cet outil permet également l'affichage d'une partie des résultats qui est stocké dans un fichier, ceci dans le but de faciliter et d'accélérer le choix des paramètres pour un fonctionnement optimal.

On retrouve deux types de vue :

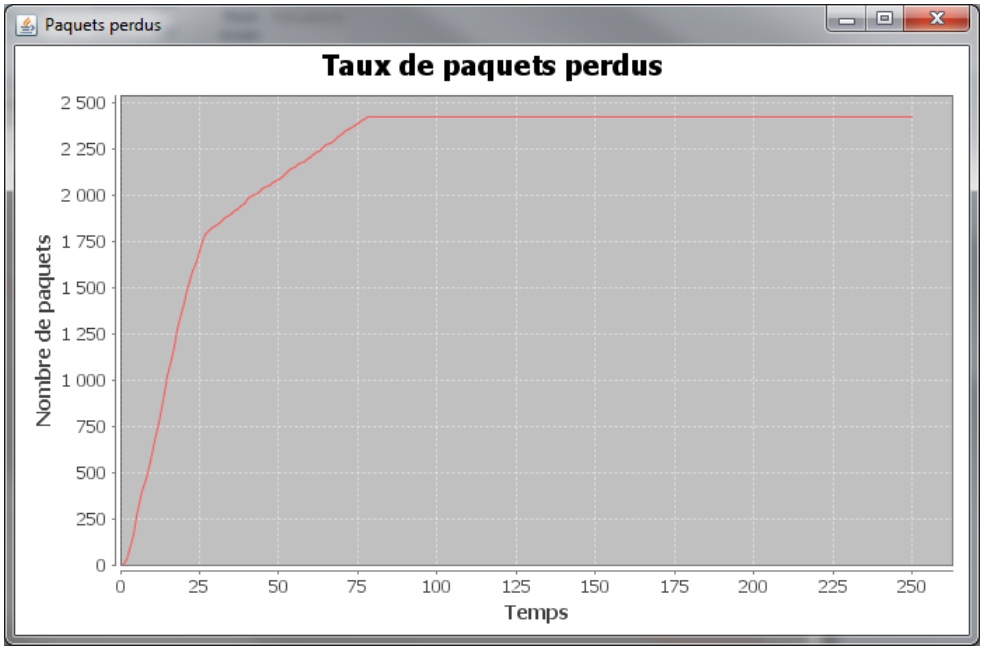
1. La vue 1 (Fig. 5.14) qui nous permet de choisir une série de paramètres puis de produire deux graphiques caractérisant l'évolution du taux de perte et de l'énergie au cours de la simulation, ceci dans un but de faire des choix rapides en fonction des possibilités de l'utilisateur. On peut voir un exemple de graphiques obtenus sur la figure 5.13.
2. La vue 2 (Fig. 5.15) qui permet d'avoir une vision partielle et globale pour un type d'application et un nombre de noeuds. Ceci dans le but d'établir un choix rapide de la couche MAC et le protocole de routage conseillé.

5.3 Conclusion

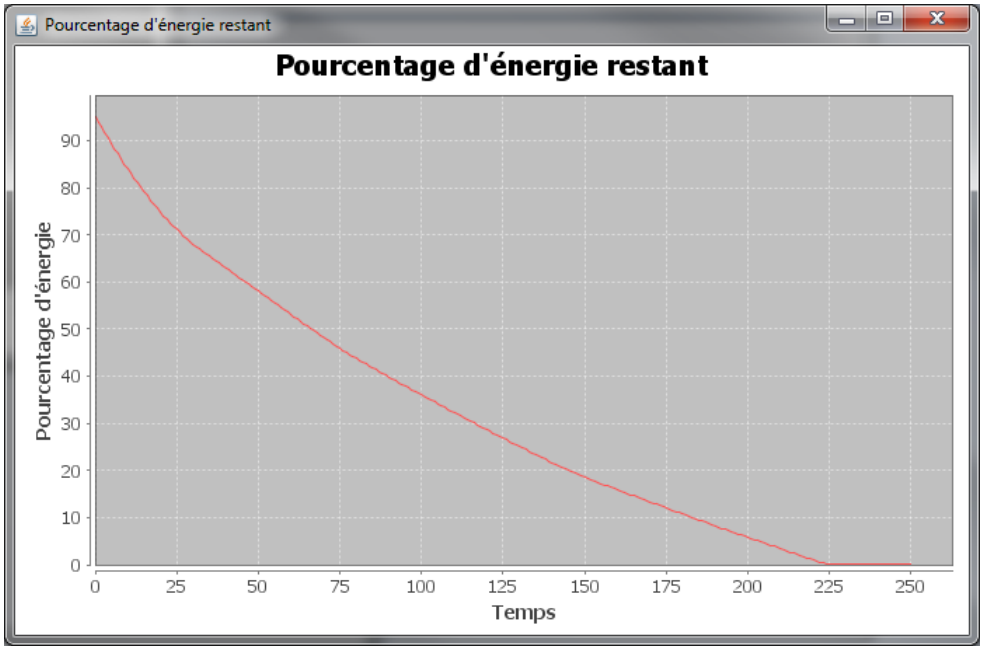
Dans cette partie, nous avons effectué notre analyse en vue d'une classification des protocoles usuels via notre outil pour la production d'un modèle de décision. Pour cela, nous avons analysé l'impact des paramètres systèmes du standard 802.15.4 et des protocoles de routages usuels dédié aux réseaux de capteurs. Nous avons donc effectué un nombre important de simulations en faisant varier plusieurs paramètres dans différentes couches en présentant des résultats numériques sur les performances du réseau dans chacun des cas. Cette contribution permet de donner la base nécessaire contenant un ensemble de traces pour la sélection appropriée des bons paramètres de configurations pour les réseaux de capteurs.

Nous montrons dans notre analyse qu'une utilisation de période de sommeil peut être faite dans le cadre d'application avec un fort débit et par à coup à condition que le nombre de capteurs reste raisonnable. De plus un mode sans balises de synchronisation peut être utilisé sur une application régulière pour toutes les tailles de réseaux. Néanmoins dans le cas d'un grand nombre de capteurs (≥ 300) on observe une dégradation des résultats quand on utilise un mode avec balises de synchronisation. Le mode sans balises reste privilégié pour tous les types d'applications, mais dans certains restes très gourmand en énergie.

Concernant les protocoles de routage les différences des résultats restent modérées dans certain cas voir très importante pour d'autres. On remarque que les protocoles réactifs tels que AODV, AOMDV et DSR sont beaucoup plus adaptés que les protocoles pro-actifs tel que DSDV. En effet pour un nombre faible de noeuds les protocoles AODV et DSR sont les plus performants, mais à mesure que la taille du réseau s'agrandit le protocole AOMDV devient plus performant ceci grâce à l'utilisation de plusieurs routes pour l'acheminement des paquets.



(a) Taux de perte



(b) Pourcentage d'énergie restant

FIGURE 5.13 – Graphiques générés par l'outil

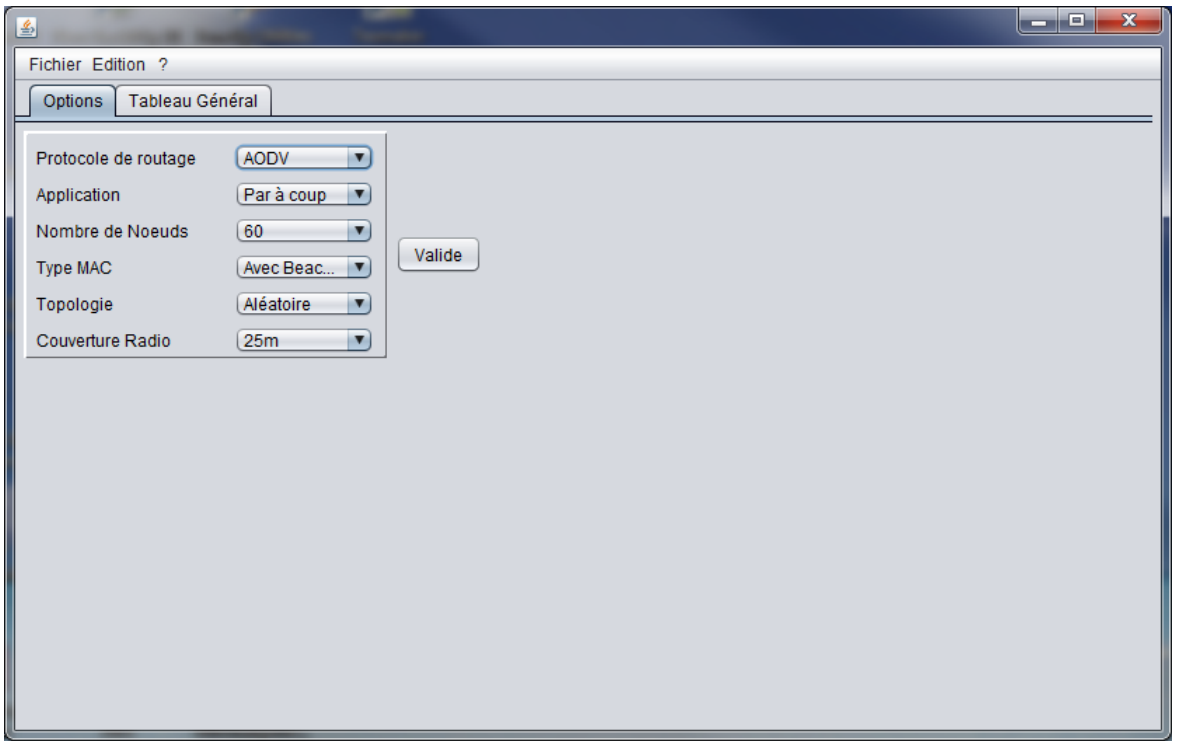
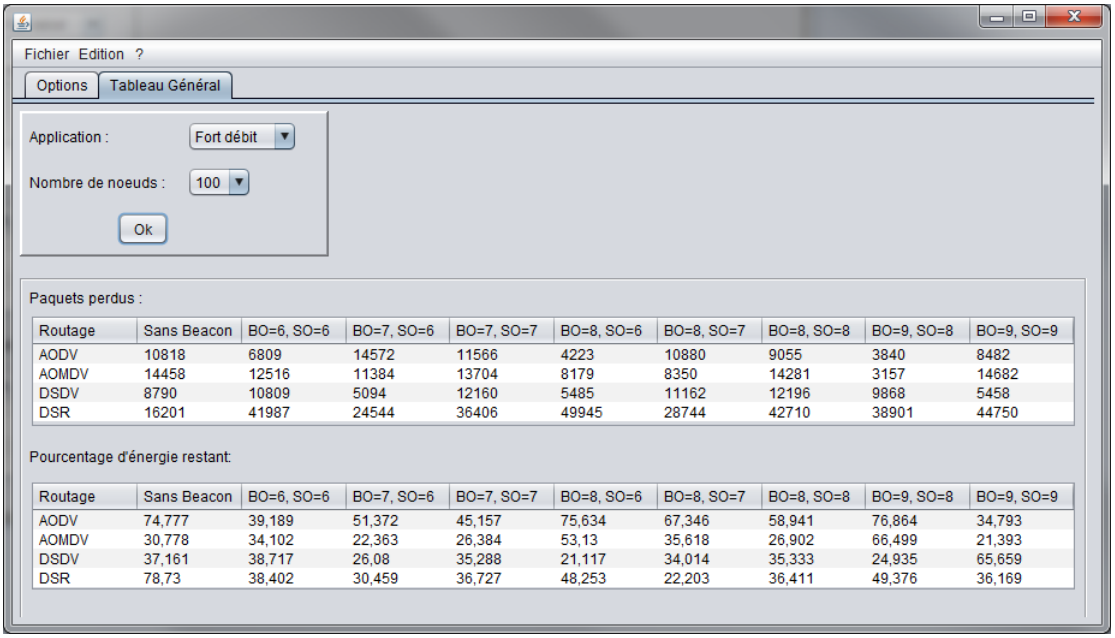


FIGURE 5.14 – Vue du module de création de graphiques



Routage	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	10818	6809	14572	11566	4223	10880	9055	3840	8482
AOMDV	14458	12516	11384	13704	8179	8350	14281	3157	14682
DSDV	8790	10809	5094	12160	5485	11162	12196	9868	5458
DSR	16201	41987	24544	36406	49945	28744	42710	38901	44750

Routage	Sans Beacon	BO=6, SO=6	BO=7, SO=6	BO=7, SO=7	BO=8, SO=6	BO=8, SO=7	BO=8, SO=8	BO=9, SO=8	BO=9, SO=9
AODV	74,777	39,189	51,372	45,157	75,634	67,346	58,941	76,864	34,793
AOMDV	30,778	34,102	22,363	26,384	53,13	35,618	26,902	66,499	21,393
DSDV	37,161	38,717	26,08	35,288	21,117	34,014	35,333	24,935	65,659
DSR	78,73	38,402	30,459	36,727	48,253	22,203	36,411	49,376	36,169

FIGURE 5.15 – Vue du module d'affichage partiel des résultats

Chapitre 6

Test des réseaux de capteurs

Sommaire

Le test de nos jours est un moyen incontournable pour vérifier la conformité, la robustesse, les performances et l'interopérabilité d'un système. Le test de conformité est réalisé en 2 phases : la première qui consiste à générer des séquences de tests à partir de la spécification, la seconde représente l'exécution de ces séquences de tests sur l'implémentation qui représente l'implémentation sous test (IUT) pour détecter les réactions dans le but de détecter les fautes. Beaucoup de formalismes connus permettent de décrire le système pour représenter la spécification. On retrouve notamment les LTS (*Labelled Transition System*) et les FSM (*Finite State Machine*) pour les systèmes non temporisés et les TIOA (*Timed Input Output Automaton*) et ETIOA (*Extended Timed Input Output Automaton*) pour les systèmes temporisés [13, 63, 69]. Il existe une panoplie de méthodes de tests qui ont fait leurs preuves grâce aux diverses techniques mises en place. Par exemple, dans [17], les auteurs proposent un modèle pour décrire les systèmes et une méthode de test pour les communications systèmes. Dans [18], les auteurs proposent une méthode de test pour les systèmes décrits sous forme de EFSM (*Extended Finite State Machine*). Ou encore dans [19], les auteurs proposent une généralisation de la méthode W (de T. S. Chow [28]) pour les spécifications ne possédant pas d'ensemble de caractérisation nécessaire pour exécuter la méthode W. On retrouve des études plus approfondies concernant le test des RTS (Real-Time Systems) dans [22, 31, 50], le test des FSM dans [57, 63] et les systèmes embarqués et systèmes distribués dans [24, 59, 58, 82, 53].

Comme nous avons vu précédemment les réseaux de capteurs représentent un cas particulier des systèmes distribués et embarqués, ces petites unités de calcul peuvent,

durant leur fonctionnement, rencontrer plusieurs problèmes : de connexions, d'autonomie, puissance de calcul, etc. Nous avons étudié et proposé un modèle permettant de choisir les bons paramètres de configuration pour un réseau de capteurs ce qui permet d'augmenter la fiabilité, la robustesse et la longévité du réseau. Dans ce chapitre, nous proposons une approche pragmatique permettant de tester la conformité et l'interopérabilité d'un réseau de capteurs dans un environnement réel. Pour tester la conformité et l'interopérabilité dans un environnement réel, nous proposons une architecture d'exécution de test sur un réseau de capteurs réel, ceci dans un but d'assurer un niveau correct de conformité et la fiabilité de celui-ci durant son fonctionnement. En utilisant cette méthode, l'utilisateur peut visualiser le fonctionnement du système, il peut exécuter des séquences de tests qui définissent des scénarios de fonctionnement du réseau de capteurs. Cette méthode permet d'assurer que le système fonctionne correctement et que les scénarios permettent d'assurer la conformité et l'interopérabilité du système. Ce chapitre reprend essentiellement celui de la publication [92].

Le chapitre est organisé comme suit : la section 6.1 présente et décrit notre architecture de test ainsi que notre modèle de séquence de test et notre exécution de test. La section 6.2 présente notre prototype de cas d'étude pour illustrer notre approche. Enfin la section 6.3 conclut le chapitre.

6.1 L'approche

6.1.1 Définitions

Avant de détailler notre approche, nous allons introduire quelques notions nécessaires telles que : un observateur (observer), un scénario et quelques notions et notations utilisées dans ce chapitre.

Définition 6.1.1 Temps de réaction :
Le temps de réaction est une limite supérieure de la quantité de temps entre :

(i) Le moment où un événement e est reçu par un noeud.

(ii) Le moment où un noeud a terminé d'envoyer toutes ses sorties (le cas échéant) qui correspondent à la réaction de la réception de l'événement e.

Ce temps est utilisé pour les applications qui ont des contraintes de temps comme les applications temps-réel.

Définition 6.1.2 Observateur :
Un observateur (Observer) est un noeud spécifique qui est dédié à l'écoute, la réception et l'analyse de messages reçus dans le réseau de capteurs.

Ce noeud est particulier. Son rôle est de surveiller le réseau d'une manière distante. Il ne participe pas à l'exécution de l'application.

Définition 6.1.3 Temps de transfert :
Le temps de transfert noté Δ, défini la quantité de temps entre :
(i) Le moment où le message M est envoyé
(ii) Le moment où M est reçu par la destination.

En général cette durée dépend uniquement de la complexité de la topologie du réseau. Mais dans la plupart des cas, on suppose qu'il n'y a aucun délai entre les observateurs et les capteurs.

Définition 6.1.4 Scénario de test :
Un scénario est décrit par une séquence de test. Une séquence de test est un ensemble ordonné d'actions/réactions où chaque événement correspond à la réception d'une action d'entrée, et, une réponse est une action de sortie générée par la réception d'un événement.

Définition 6.1.5 Scénario correct :
Soit une séquence d'actions d'entrées $t_{ia} = a_1, a_2, ..., a_n$ et une réaction de sortie $t_{rm} = b_1, b_2, ..., b_n$. L'exécution d'un scénario dans un réseau de capteurs est correcte si et seulement si :

1. b_i est la réaction de a_i,
2. a_i est exécuté avant a_{i+1} et b_i avant b_{i+1},
3. Les temps de réaction entre les entrées et sorties sont respectés.

6.1.2 Architecture de Test

Notre approche de test est basée sur la surveillance de scénarios d'exécution sur un réseau de capteurs. A cause de la couverture radio des noeuds, nous privilégions une utilisation d'observateurs distribués pour être sûr de couvrir tous les capteurs dans le réseau. Les observateurs sont contrôlés par un utilisateur (ou par un processus automatique). Dans le but de déterminer la position des observateurs dans le réseau, nous effectuons un découpage du réseau en plusieurs sous-réseaux. Nous plaçons dans chaque sous-réseau une station de base qui est utilisée pour récupérer les données de chaque sous-réseau. Nous connectons à chacune de ces stations de base un observateur pour contrôler les données.

La station de base permet de relayer les données des noeuds à l'observateur et vice-versa. L'observateur quant à lui va envoyer des actions d'entrée (du scénario) et contrôler les réactions de sortie qu'il reçoit. Il vérifie si les contraintes de temps sont respectées, dans le but de s'assurer qu'un scénario s'est exécuté correctement sur le réseau. L'utilisateur contrôle l'exécution du scénario. Il peut vérifier si une anomalie survient dans le réseau tel que : un lien mort, une perte d'énergie, de la congestion, surcharge, etc. Le verdict est produit par l'observateur qui conclut si le réseau de capteurs a eu des anomalies. On suppose, en amont, qu'une génération de séquences de test pour le système est réalisée comme décrit dans [58] ou dans [97]. Cependant, nous supposons que nous n'avons pas besoin de générer toutes les séquences de tests

possibles, mais uniquement les cas critiques d'utilisation du réseau de capteurs. En effet, en fonction de la complexité de l'application la quantité de séquences de test peut être exponentielle. La figure 6.1 décrit notre approche et la figure 6.2 décrit l'échange des messages entre les entités.

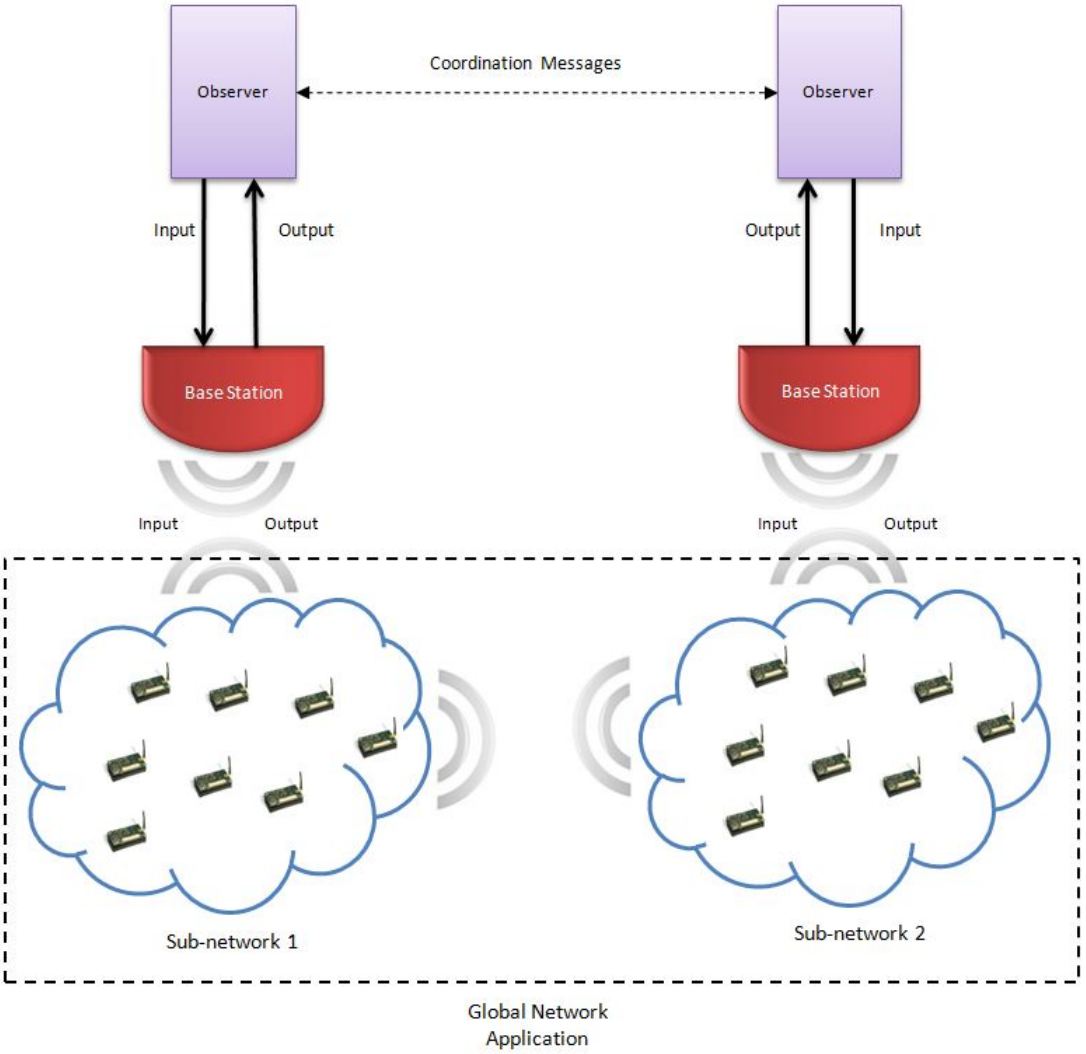


FIGURE 6.1 – Architecture de Test

6.2 Résultats d'expérimentations

6.2.1 Configuration de notre réseau de capteurs

Pour tester notre approche, nous avons implémenté un prototype. Ce prototype est déployé sur un vrai réseau de capteurs composé de capteurs MicaZ communicants avec le protocole ZigBee (802.15.4). L'application déployée est codée en NesC [44]. Notre plateforme de test est composée de deux stations de base, cinq capteurs et deux ordinateurs (observateurs). Pour permettre les communications entre la station de base et l'ordinateur, nous utilisons un câble USB. Nous testons deux architectures,

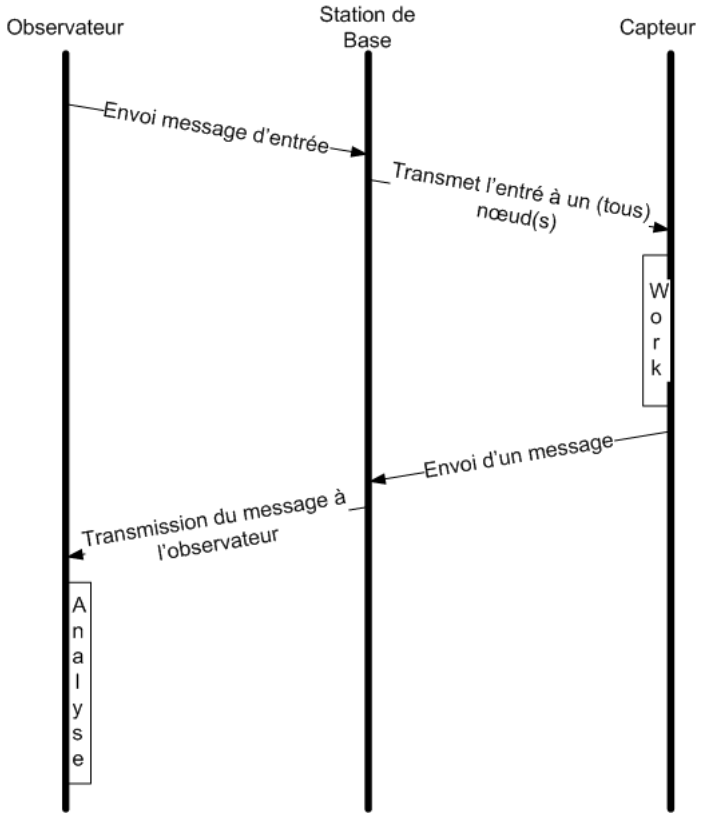


FIGURE 6.2 – Séquence d'échange des messages

les figures 6.3 et 6.4 décrivent leur fonctionnement. Dans la première représentation, nous utilisons un observateur qui peut voir tous les noeuds, tandis que dans la seconde représentation on utilise deux observateurs pour s'assurer que l'information est transmise de bout en bout du réseau de capteurs.

FIGURE 6.3 – Première représentation

L'application de nos observateurs est écrite en JAVA et nous utilisons la librairie de TinyOS pour les communications USB entre la station de base et l'ordinateur. L'application est séparée en deux processus : un premier dédié à envoyer des actions d'entrée dans le réseau et un second dédié à la réception des réactions comme le montre la figure 6.5. L'écoute du réseau est assurée en fonction de la couverture radio de la station de base. Nous écoutons et récupérons les messages durant leurs envois sur le réseau et nous analysons leur contenu. Le paquet étant un paquet brut il est nécessaire de connaître la structure pour en récupérer les informations

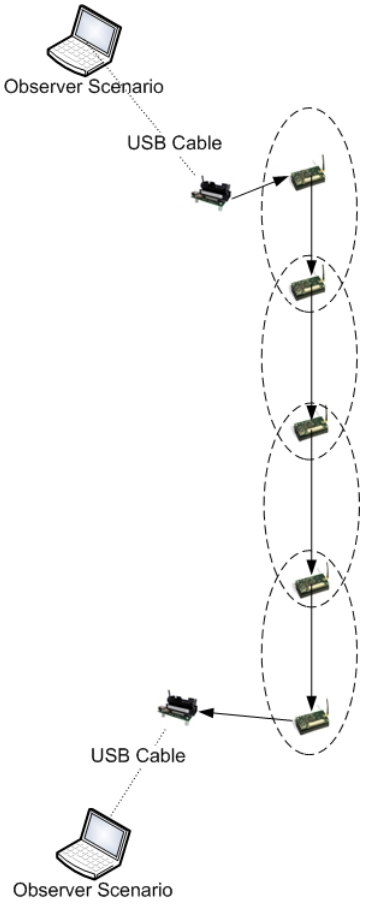


FIGURE 6.4 – Seconde représentation

essentielles. La structure d'un paquet se détaille comme l'indique le tableau 6.1.

Src	Dest	AM	Grp	Len	Data
(2)	(2)	(1)	(1)	(1)	(0...29)

TABLE 6.1 – Format des paquets de TinyOs (taille en octets)

6.2.2 Modèle applicatif

Notre application déployée sur le réseau est une application simple avec une possibilité de trois événements tels que : la gestion du "timer", la gestion de la radio (on/off) et la gestion d'envoi et réception de messages.

L'application est similaire à une application simple de récupération de température. Un capteur dédié à la capture de la température récupère la température puis transfère le message de capteur en capteur jusqu'à la station de base. Chaque noeud doit mettre à jour sa valeur puis la transférer en effectuant un *broadcast*. Le premier objectif est de garantir qu'à un instant t les messages observés sur tous les noeuds d'un sous-réseau sont identiques.

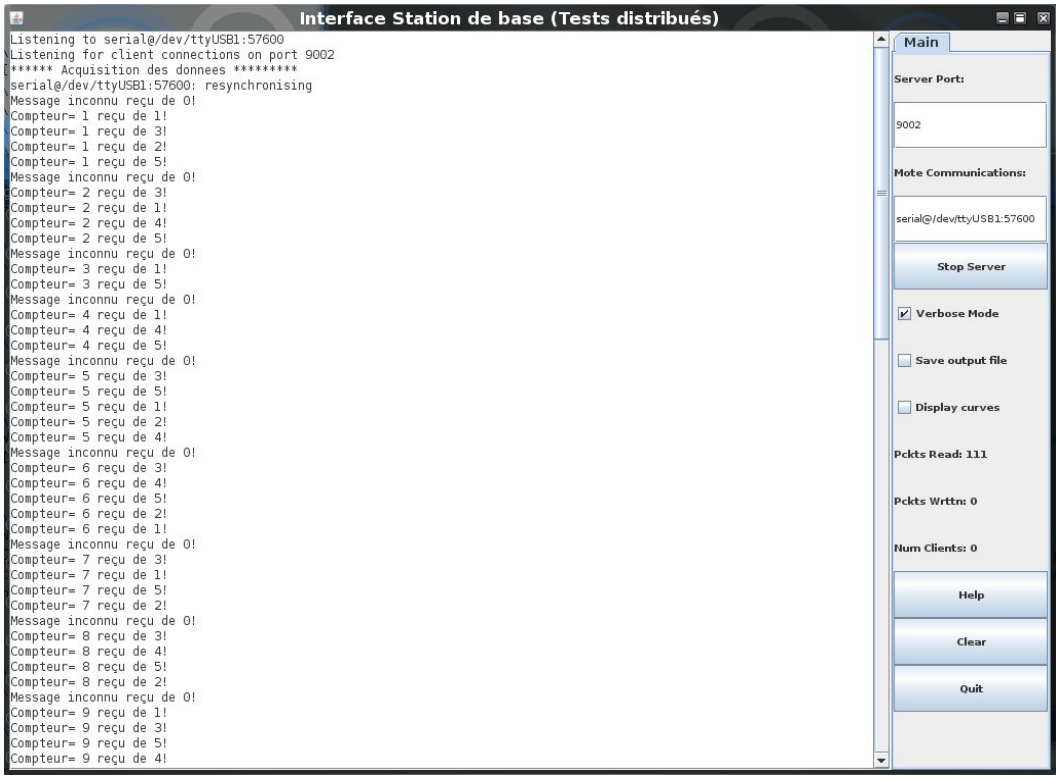


FIGURE 6.5 – Processus de réception des réactions

Scénario de test

Notre scénario de test est en relation avec l'application. L'objectif est d'envoyer toutes les 5 secondes un message dans le réseau. Ce message représente un compteur qui se retrouve incrémenté après chaque itérations(toutes les 5 secondes). Ce message est envoyé en brodcast à tous les capteurs dans la couverture radio de l'observateur.

Dans notre première architecture de test (figure 6.3), nous devons nous assurer qu'à tout instant t tous les capteurs possèdent le même message. Dans la seconde architecture (figure 6.4), l'objectif est de s'assurer que le message parcourt tout le réseau et arrive au dernier capteur qui est un noeud puits.

6.2.3 Résultats

Concernant la première architecture, on remarque que les noeuds morts et les mauvaises communications sont détectés par l'observateur. Néanmoins, nous n'avons pas de garantie sur les paquets perdus. De plus, l'application peut vérifier durant la première itération que le capteur a rencontré quelques problèmes. Mais si durant la seconde itération le message est reçu par le capteur à ce moment, l'observateur peut décider si le capteur se trouve dans une mauvaise position par rapport à sa couverture radio.

Dans la seconde architecture, on observe les mêmes constats qu'avec la première architecture. Mais dans le cas où on se retrouve avec un noeud routeur, comme le montre la figure 6.6, on observe seulement des problèmes de communications et non

un problème de noeud.

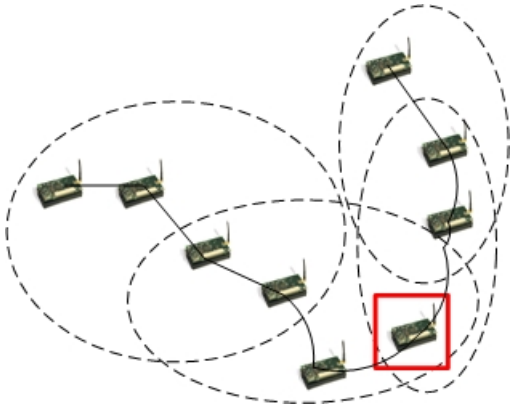


FIGURE 6.6 – Noeud Routeur

6.3 Conclusion

Dans ce chapitre, nous avons décrit une architecture distribuée de test pour les réseaux de capteurs en utilisant des observateurs. Nous avons présenté une méthodologie pour vérifier et détecter un certain nombre d'anomalies sur le réseau. Le principal avantage de cette architecture est basé sur l'utilisation d'observateur qui permet d'établir une semi-conformité des applications en effectuant une surveillance d'exécution de scénarios sur le réseau de capteurs.

Nous montrons dans notre analyse que les observateurs sont capables de détecter certaines anomalies comme : les problèmes de noeuds morts, problèmes de communications, problèmes de positionnement. Ces anomalies représentent une succession d'actions qui a conduit à des fautes. Ces fautes sont les résultats de problème de fiabilité, longévité et robustesse du réseau de capteurs.

De notre approche, nous avons tiré deux principales orientations :

- ➢ La première qui consiste à établir un modèle de fautes nous permettant une détection plus rapide et efficace des anomalies. En effet, une suite d'actions conduit dans ces cas à une faute qui génère une anomalie. Dans cette optique, on retrouve des études sur la détection de fautes dans [99] qui peut nous indiquer les méthodes à adopter.
- ➢ D'autre part, nous devons nous assurer de la robustesse de nos applications de surveillance et donc des observateurs. En effet, il est important de s'assurer que la surveillance du réseau de capteurs soit fonctionnelle même si un observateur a des soucis.

L'objectif final permettra d'obtenir une plateforme de test et de surveillance en temps réel des niveaux d'énergie, de la qualité du réseau, de la charge de travail des capteurs...

Chapitre 7

Conclusion et perspectives

Les réseaux de capteurs sont de nouveaux systèmes distribués que les dernières avancées technologiques des systèmes embarqués ont rendus possibles. Néanmoins le manque d'outil de modélisation et la complexité de leur configuration en font un frein au déploiement de ces réseaux. Les noeuds d'un réseau de capteurs sont autonomes en énergie, alors on souhaite avoir, pour déployer un réseau, certaines garanties sur la fiabilité, la robustesse et la durée de vie de celui-ci.

7.1 Bilan

7.1.1 L'impact de certains paramètres

Le choix de configuration est un facteur déterminant pour les performances d'un réseau de capteurs. On retrouve dans cette configuration : le protocole d'accès au médium (MAC et PHY), le type de capteurs, le placement des capteurs (topologie) avec leurs puissances radio, le protocole de routage qui ont tous des conséquences en fonction du type d'application. Nous avons travaillé tout d'abord sur l'étude de l'impact de la couverture radio, de la topologie et de la taille du réseau sur les performances du réseau de capteurs. Pour la réalisation de cette étude nous avons utilisé le simulateur NS-2. Ce simulateur permet de faire varier de nombreux paramètres correspondait au choix le plus adéquat à la situation.

Nous sommes arrivés aux conclusions suivantes :

- ➢ Les protocoles de routage perdent beaucoup plus d'énergie quand la couverture radio augmente.
- ➢ En fonction du type d'application, les protocoles de routage peuvent être adaptés pour réduire la quantité de paquets perdus.
- ➢ AOMDV se retrouve être le meilleur dans les situations stressantes (surcharge et nombre de noeuds). Dans les autres cas, ce résultat dépend principalement du type d'application utilisé, DSDV et AODV étant les plus adaptés.

Nous avons donc conclu que le choix de la couverture doit être lié à la topologie utilisée pour bénéficier d'un gain significatif. Car l'impact de la couverture radio est lié à la topologie et reste relativement modéré. Ces travaux ont été présentés dans [90, 91].

7.1.2 Classification

Nous avons effectué une classification des protocoles usuels pour la production d'un modèle d'aide à la décision. Ceci dans le but de produire les paramètres optimaux de configuration pour un réseau de capteurs. Pour cela, notre étude s'est focalisée sur l'impact des paramètres systèmes sur les performances du réseau de capteurs. Nous avons fait varier un grand nombre de paramètres de simulation concernant la couche MAC, le protocole de routage, le nombre de noeuds et l'application.

Nous sommes arrivés aux conclusions suivantes :

- ➢ Une utilisation de période d'inactivité peut servir dans le cadre d'application stressante (par à coup ou à fort débit) dans le but de réduire la consommation d'énergie et diminué la perte de paquets.
- ➢ Un mode sans balises de synchronisation peut être utilisé sur une application régulière pour toutes les tailles de réseaux. Néanmoins dans le cas d'un grand nombre de capteurs (≥ 300) on observe une dégradation des résultats quand on utilise un mode avec balises de synchronisation.
- ➢ Le mode sans balises reste privilégié pour tous les types d'applications.

Concernant les protocoles de routage les différences des résultats restent modérées dans certain cas voir très importante pour d'autres. On remarque que :

- ➢ Les protocoles réactifs tels que AODV, AOMDV et DSR sont beaucoup plus adaptés que les protocoles pro-actifs tel que DSDV.
- ➢ Pour un nombre faible de noeuds les protocoles AODV et DSR sont les plus performants, mais à mesure que la taille du réseau s'agrandit le protocole AOMDV devient plus performant ceci grâce à l'utilisation de plusieurs routes pour l'acheminement des paquets.

Cette étude a été publiée dans [89].

7.1.3 Le test de conformité et d'interopérabilité

Nous avons également décrit une architecture distribuée de test de conformité et d'interopérabilité pour les réseaux de capteurs en utilisant des observateurs. Nous avons présenté une méthodologie pour vérifier et détecter un certain nombre d'anomalies sur le réseau. Le principal avantage de cette architecture est basé sur l'utilisation d'observateurs qui permettent d'établir une conformité des applications en appliquant des séquences de test sur l'implantation, puis vérifier la conformité des réponses par rapport à la spécification. D'autre part, ces observateurs permettent également de vérifier le fonctionnement correct de l'interaction entre les capteurs. Ceci s'effectue en analysant les communications entre les capteurs.

Nous montrons dans notre analyse que les observateurs sont capables de détecter certaines anomalies comme : les problèmes de noeuds morts, problèmes de communications, problèmes de positionnement. Ces anomalies représentent une succession d'actions qui a conduit à des fautes. Ces fautes sont les résultats de problèmes de fiabilité, longévité et robustesse du réseau de capteurs. Notre modèle a été présenté dans [92].

7.2 Perspectives

Dans un premier temps nous pourrions élargir nos différentes études en variant d'autres paramètres pour obtenir plus de résultats. Néanmoins nous avons remarqué qu'il est nécessaire de produire un modèle d'analyse des résultats autonomes ce qui nous permettrait de classer nos résultats de façon plus rapide. Dans la même optique, améliorer le modèle qui permettrait au fil du temps de s'enrichir en terme de paramètres et résultats pour choisir la configuration la plus adaptée pour une situation bien précise. Cette application doit pouvoir prendre en compte les résultats pouvant venir d'un autre simulateur ou même d'un test en environnement réel. Par ailleurs, il devra proposer une certaine liberté à l'utilisateur le choix en fonction des coûts des solutions proposées. Car une solution optimale peut avoir un certain coût, tandis qu'une solution tout à fait correcte, mais moins optimale peut avoir des coûts de fabrications et d'assemblage nettement inférieur à l'autre.

Dans un second temps l'amélioration de notre architecture de test pour produire une plateforme de test autonome. Cette plateforme pourra assurer les critères de conformité et robustesse d'un réseau de capteurs. Pour cela il serait important d'étudier 2 pistes :

1. La première piste consiste à établir un modèle de fautes pouvant être détecté dans un réseau de capteurs. Ce modèle de fautes sera composé des fautes conduisant à une anomalie applicative, une anomalie du noeud et des transmissions puis de manière plus générale le réseau complet. Ce modèle sera détaillé des différents actions ou événements qui conduisent à l'apparition de ces anomalies.
2. La seconde piste consiste à établir un modèle de représentation du réseau de capteurs dans son intégralité de façon la plus simple et la plus efficace que possible. Puis de cette représentation, nous produirions les méthodes d'extraction des différentes séquences de tests permettant la validation de conformité ou de robustesse de l'IUT (*Implementation Under Test*) du réseau.

Bibliographie

[1] Csma/cd du standard 802.3. http ://standards.ieee.org/getieee802/download/802.3an-2006.pdf.

[2] Imote specification. http ://www.intel.com/research/exploratory/motes.htm.

[3] The mathworks. matlab. http ://www.mathworks.com/products/matlab/.

[4] Mica sensors specification. http ://www.xbow.com.

[5] Spec specification. http ://www.jlhlabs.com/jhill_cs/spec/index.htm.

[6] Specification of the bluetooth system. http ://www.bluetooth.org/.

[7] Sun spot motes. http ://www.sunspotworld.com/.

[8] Tinyos home page. http ://www.tinyos.net.

[9] Ieee standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements - part 11 : Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, pages C1–1184, 12 2007.

[10] Zoë Abrams, Ashish Goel, and Serge Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 424–432, New York, NY, USA, 2004. ACM.

[11] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks : a survey. *Computer Networks*, 38(4) :393 – 422, 2002.

[12] Jamal N. Al-karaki and Ahmed E. Kamal. Routing techniques in wireless sensor networks : A survey. *IEEE Wireless Communications*, 11 :6–28, 2004.

[13] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126 :183–235, 1994.

[14] Khaled A. Arisha. Energy-aware tdma-based mac for sensor networks. In *IEEE Workshop on Integrated Management of Power Aware Communications, Computing and Networking (IMPACCT 2002*, 2002.

[15] Paolo Baronti, Prashant Pillai, Vince W.C. Chook, Stefano Chessa, Alberto Gotta, and Y. Fun Hu. Wireless sensor networks : A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer Communications*, 30(7) :1655 – 1695, 2007. Wired/Wireless Internet Communications.

[16] J. Bengtsson and F. Larsson. *UPPAAL - a Tool Suite for Symbolic and Compositional Verification of Real-Time Systems (Draft).*

[17] Ismail Berrada, Richard Castanet, and Patrick Félix. Testing communicating systems : a model, a methodology and a tool. In *17th IFIP International Conference on Testing of Communicating Systems ; Lecture Notes in Computer Science*, Montreal, Canada, may 2005. Elsevier.

[18] Johan Blom, Anders Hessel, Bengt Jonsson, and Paul Pettersson. Specifying and generating test cases using observer automata. In *Proc. 4 th International Workshop on Formal Approaches to Testing of Software 2004 (FATES'04), volume 3395 of Lecture Notes in Computer Science*, pages 125–139. Springer–Verlag, 2005.

[19] Adilson Luiz Bonifácio, Arnaldo Vieira Moura, and ao Sim Adenilso da Silva. A generalized model-based test generation method. In *SEFM '08 : Proceedings of the 2008 Sixth IEEE International Conference on Software Engineering and Formal Methods*, pages 139–148, Washington, DC, USA, 2008. IEEE Computer Society.

[20] David Braginsky and Deborah Estrin. Rumor routing algorthim for sensor networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, WSNA '02, pages 22–31, New York, NY, USA, 2002. ACM.

[21] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu. Advances in network simulation. *Computer*, 33(5) :59–67, May 2000.

[22] Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner. *Model-Based Testing of Reactive Systems : Advanced Lectures (Lecture Notes in Computer Science)*. Springer-Verlag New York, Inc, Secaucus, NJ, USA, 2005.

[23] Michael Buettner, Gary V. Yee, Eric Anderson, and Richard Han. X-mac : a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 307–320, New York, NY, USA, 2006. ACM.

[24] L. Cacciari and O. Rafiq. Controllability and observability in distributed testing. *Information and Software Technology*, 41(11-12) :767–780, 9/15 1999.

[25] Rachel Cardell-oliver, Keith Smettem, Mark Kranz, and Kevin Mayer. Field testing a wireless sensor network for reactive environmental monitoring. In *In International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 14–17, 2004.

[26] Supriyo Chatterjea, Simone De Luigi, and Paul J. M. Havinga. Dirq : A directed query dissemination scheme for wireless sensor networks. In Abraham O. Fapojuwo and Bozena Kaminska, editors, *Wireless and Optical Communications*. IASTED/ACTA Press, 2006.

[27] Chee-Yee Chong Chee-Yee Chong and S P Kumar. Sensor networks : evolution, opportunities, and challenges. volume 91, pages 1247–1256. IEEE, 2003.

[28] T. S. Chow. Testing software design modeled by finite-state machines. *IEEE Trans.Softw.Eng.*, 4(3) :178–187, 1978.

[29] Kaushik R. Chowdhury, Nagesh Nandiraju, Pritam Chanda, Dharma P. Agrawal, and Qing-An Zeng. Channel allocation and medium access control for wireless sensor networks. *Ad Hoc Netw.*, 7 :307–321, March 2009.

[30] Ching chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in clustered multihop, mobile wireless networks with fading channel, 1997.

[31] Camille Constant. Génération automatique de tests pour des modèles avec variables ou récursivité., 2008-11-24 2008. ID : tel-00424546, version 1.

[32] T. van Dam and K. Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. pages 171–180, Los Angeles, CA, November 2003.

[33] I Demirkol, C Ersoy, and F Alagoz. Mac protocols for wireless sensor networks : a survey, 2006.

[34] Nathalie Dessart, Hacène Fouchal, Philippe Hunel, Harry Gros-Desormeaux, and Nicolas Vidot. Distributed decision for medical alerts using wireless sensors. In *WOWMOM*, pages 1–6. IEEE, 2009.

[35] Nathalie Dessart, Hacène Fouchal, Philippe Hunel, and Nicolas Vidot. On using a distributed approach for help in medical diagnosis with wireless sensor networks. In Gerald Eichler, Axel Küpper, Volkmar Schau, Hacène Fouchal, and Herwig Unger, editors, *IICS*, volume P-186 of *LNI*, pages 70–81. GI, 2011.

[36] S.M. Douglas, I. Bachelet, and G.M. Church. A logic-gated nanorobot for targeted transport of molecular payloads. *Science*, 335(6070) :831–4, 2012.

[37] Ian Downard and Washington Dc. Simulating sensor networks in ns-2. *Proceedings of the IEEE*, pages 1–9, 2004.

[38] E. Egea-Lopez, J. Vales-Alonso, A. S. Martinez-Sala, P. Pavon-Marino, and J. Garcia-Haro. Simulation tools for wireless sensor networks. In *Summer Simulation Multiconference - SPECTS 2005*, 2005.

[39] A. El-Hoiydi and J.-D. Decotignie. WiseMAC : An ultra low power MAC protocol for multi-hop wireless sensor networks. In *First Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS 2004), Lecture Notes in Computer Science, LNCS 3121*, pages 18–31. Springer-Verlag, July 2004.

[40] Deborah Estrin, Mark Handley, John Heidemann, Steven McCanne, Ya Xu, and Haobo Yu. Network visualization with nam, the vint network animator. *Computer*, 33(11) :63–68, November 2000.

[41] Antoine Fraboulet, Guillaume Chelius, and Eric Fleury. Worldsens : development and prototyping tools for application specific wireless sensors networks. In *Proceedings of the 6th international conference on Information processing in sensor networks*, IPSN '07, pages 176–185, New York, NY, USA, 2007. ACM.

[42] J. J. Garcia-Luna-Aceves and Marcelo Spohn. Source-tree routing in wireless networks. In *Proceedings of the Seventh Annual International Conference on Network Protocols*, ICNP '99, pages 273–, Washington, DC, USA, 1999. IEEE Computer Society.

[43] Luis Javier García Villalba, Ana Lucila Sandoval Orozco, Alicia Triviño Cabrera, and Cláudia Jacy Barenco Abbas. Routing protocols in wireless sensor networks. *Sensors (Peterboroug*, 9(11) :8399–8421, 2009.

[44] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler. The nesc language : A holistic approach to networked embedded systems. *SIGPLAN Not.*, 38(5) :1–11, May 2003.

[45] Dr Aditya Goel and Ajaii Sharma. Performance analysis of mobile ad-hoc network using aodv protocol. *International Journal of Computer Science and Security*, 2009.

[46] T.B. Gosnell, J.M. Hall, C.L. Ham, D.A. Knapp, Z.M. Koenig, S.J. Luke, B.A. Pohl, A. Schach von Wittenau, and J.K. Wolford. Gamma-ray identification of nuclear weapon materials. Lawrence Livermore National Lab., Livermore, CA (USA), 1997.

[47] Harry Gros-Desormeaux, Philippe Hunel, and Nicolas Vidot. Counting birds with wireless sensor networks. In Mohsen Guizani, Peter Müller, Klaus-Peter Fähnrich, Athanasios V. Vasilakos, Yan Zhang, and Jun Zhang, editors, *IWCMC*, pages 1163–1167. ACM, 2009.

[48] Tian He, John A. Stankovic, Chenyang Lu, and Tarek Abdelzaher. Speed : A stateless protocol for real-time communication in sensor networks. In *Proc. 23rd International Conference on Distributed Computing Systems (ICDCS'03)*, Providence, Rhode Island, May 2003.

[49] W.B. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4) :660–670, October 2002.

[50] Anders Hessel and Paul Pettersson. A test case generation algorithm for real-time systems. *Quality Software, International Conference on*, 0 :268–273, 2004.

[51] J. Hill and D. Culler. Mica : a wireless platform for deeply embedded networks. *IEEE Micro*, 22(6) :12–24, November 2002.

[52] Jason Hill, Mike Horton, Ralph Kling, and Lakshman Krishnamurthy. The platforms enabling wireless sensor networks. *Commun. ACM*, 47(6) :41–46, June 2004.

[53] William Hoarau. Injection de fautes dans les systèmes distribués, 21 mars 2008.

[54] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion : a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, MobiCom '00, pages 56–67, New York, NY, USA, 2000. ACM.

[55] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. pages 62–68, 2001.

[56] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[57] Fujiwara Bochmann Khendek, S. Fujiwara, G. V. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models. *IEEE Transactions on Software Engineering*, 17 :591–603, 1991.

[58] Ahmed Khoumsi. Test execution for distributed real time systems.

[59] Ahmed Khoumsi. Testing distributed real time systems using a distributed test architecture. *Computers and Communications, IEEE Symposium on*, 0 :0648, 2001.

[60] Mikko Kohvakka, Jukka Suhonen, Mauri Kuorilehto, Marko Hännikäinen, and D. Hämäläinen. Network signaling channel for improving zigbee performance in dynamic cluster-tree networks. *EURASIP J. Wirel. Commun. Netw.*, 2008 :27 :1–27 :15, January 2008.

[61] Samad S. Kolahi. Comparison of fdma. tdma and cdma system capacities. In *Proceedings of the 10th WSEAS international conference on Communications*, ICCOM'06, pages 333–336, Stevens Point, Wisconsin, USA, 2006. World Scientific and Engineering Academy and Society (WSEAS).

[62] Joanna Kulik, Wendi Heinzelman, and Hari Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wirel. Netw.*, 8(2/3) :169–185, March 2002.

[63] David Lee and Mihalis Yannakakis. Principles and methods of testing finite state machines — a survey.

[64] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos : An operating system for sensor networks. In Werner Weber, Jan M. Rabaey, and Emile Aarts, editors, *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005.

[65] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. Tossim : accurate and scalable simulation of entire tinyos applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pages 126–137, New York, NY, USA, 2003. ACM.

[66] Yi Li, Dongliang Xie, Jian Ma, and Canfeng Chen. A mac-layer retransmission algorithm designed for zigbee protocol. In *MSN*, pages 314–325, 2007.

[67] S. Lindsey and C. S. Raghavendra. PEGASIS : Power Efficient GAthering in Sensor Information Systems. In *Proceedings of IEEE Aerospace Conference*, 2002.

[68] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks. In *Int. Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN)*, Santa Fe, NM, April 2004.

[69] Frank Jin Ye Luo. A diagnostic method for non-deterministic finite state machines, 1996.

[70] L A N Man and Standards Committee. Ieee standard for information technology- telecommunications and information exchange between systems-local and metropolitan area networks- specific requirements–part 15.4 : Wireless mac and phy specifications for low-rate wpans. *Control*, 2006(September) :1–305.

[71] Arati Manjeshwar and Dharma P. Agrawal. Teen : A routing protocol for enhanced efficiency in wireless sensor networks. In *Proceedings of the 15th*

International Parallel & Distributed Processing Symposium, IPDPS '01, pages 189–, Washington, DC, USA, 2001. IEEE Computer Society.

[72] Mahesh K. Marina and Samir R. Das. On-demand multipath distance vector routing in ad hoc networks. In *in Proceedings of IEEE International Conference on Network Protocols (ICNP*, pages 14–23, 2001.

[73] Leslie Jill Miller. The iso reference model of open systems interconnection : A first tutorial. In *Proceedings of the ACM '81 conference*, ACM '81, pages 283–288, New York, NY, USA, 1981. ACM.

[74] Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mob. Netw. Appl.*, 1(2) :183–197, October 1996.

[75] Bilel Nefzi and Ye-Qiong Song. Performance Analysis and improvement of ZigBee routing protocol. In *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems - FeT'2007*, Toulouse France, 2007. IFAC.

[76] The network simulator ns-2. http ://www.isi.edu/nsnam/ns/, 2001.

[77] Amit N.Thakare and Mrs. M. Y. Joshi. Performance analysis of aodv & dsr routing protocol in mobile ad hoc networks. *IJCA Special Issue on MANETs*, pages 211–218, 2010. Published by Foundation of Computer Science.

[78] Mobility framework for omnet++. http ://mobility-fw.sourceforge.net.

[79] Omnet++ discrete event simulator. http ://www.omnetpp.org.

[80] Sung Park, Andreas Savvides, and Mani B. Srivastava. Sensorsim : a simulation framework for sensor networks. In *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, MSWIM '00, pages 104–111, New York, NY, USA, 2000. ACM.

[81] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing in mobile ad hoc networks. In *In ICDCS Workshop on Wireless Networks and Mobile Computing*, pages 71–78, 2000.

[82] Carlos Eduardo Pereira and Luigi Carro. Distributed real-time embedded systems : Recent advances, future trends and their impact on manufacturing plant control. *Annual Reviews in Control*, 31(1) :81–92, 2007.

[83] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. 2003.

[84] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4) :234–244, October 1994.

[85] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. pages 95–107, Baltimore, MD, November 2004.

[86] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras. Atemu : a fine-grained sensor network simulator. In *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 145–152. IEEE Press, October 2004.

[87] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5) :51–58, May 2000.

[88] C. Prehofer and C. Bettstetter. Self-organization in communication networks : Principles and design paradigms. *IEEE Commun. Mag.*, 43(7) :78–85, July 2005.

[89] C. Ramassamy, H. Fouchal, and P. Hunel. Classification of usual protocols over wireless sensor networks. In *IEEE International Conference on Communications (ICC'2012)*, June 2012.

[90] Cédric Ramassamy, Hacène Fouchal, and Philippe Hunel. Impact of application layers over wireless sensor networks. In *12th International Conference on Innovative Internet Systems*, Trondheim, Norway, June 2012. GI.

[91] Cédric Ramassamy, Hacene Fouchal, and Philippe Hunel. Impact of transmission range in 802.15.4 with usual routing protocols. In *Wireless Networking Symposium (IWCMC2012-Wireless Nets)*, Limassol, Cyprus, August 2012.

[92] Cédric Ramassamy, Hacène Fouchal, Philippe Hunel, and Nicolas Vidot. A pragmatic testing approach for wireless sensor networks. In *Proceedings of the 6th ACM workshop on QoS and security for wireless and mobile networks*, Q2SWinet '10, pages 55–61, New York, NY, USA, 2010. ACM.

[93] T.S. Rapport. *Wireless communications : principles and practice*. Prentice Hall communications engineering and emerging technologies series. Prentice Hall PTR, 1996.

[94] George F. Riley. The georgia tech network simulator. In *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, MoMeTools '03, pages 5–12, New York, NY, USA, 2003. ACM.

[95] Azat Rozyyez and Halabi Hasbullah. Comparison of routing protocols for child tracking in wsn. In *4th International Symposium on Information Technology 2010 (ITSim'10)*, pages 1238–1243, June 2010.

[96] N. Sadagopan, B. Krishnamachari, and A.Helmy. The ACQUIRE mechanism for efficient querying in sensor networks. In *Proc. 1st IEEE Intl. Workshop on Sensor Network Protocols and Applications (SNPA)*, Anchorage, AK, May 2003.

[97] Ahmad A. Saifan, Ernesto Posse, and Juergen Dingel. Run-time conformance checking of mobile and distributed systems using executable models. In *PADTAD '09 : Proceedings of the 7th Workshop on Parallel and Distributed Systems*, pages 1–11, New York, NY, USA, 2009. ACM.

[98] Mohammad Sharawi and Dennis Marko. Introduction to direct sequence spread spectrum communications. *Signals*, (November), 2000.

[99] Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults : Detection methods and prevalence in real-world datasets. *ACM Transactions on Sensor Networks (TOSN)*, 6(3) :1–39, 2010.

[100] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 188–200, New York, NY, USA, 2004. ACM.

[101] Ahmed Sobeih, Wei-Peng Chen, Jennifer C. Hou, Lu-Chuan Kung, Ning Li, Hyuk Lim, Hung-Ying Tyan, and Honghai Zhang. J-sim : A simulation environment for wireless sensor networks. In *Proceedings of the 38th annual Symposium on Simulation*, ANSS '05, pages 175–187, Washington, DC, USA, 2005. IEEE Computer Society.

[102] Katayoun Sohrabi and Gregory J. Pottie. Performance of a novel self-organization protocol for wireless ad-hoc sensor networks. In *Proc. IEEE 50th Vehicular Technology Conference (VTC)*, pages 1222–1226, 1999.

[103] Avinash Sridharan, Marco Zuniga, and Bhaskar Krishnamachari. Integrating environment simulators with network simulators, 2004.

[104] Pablo Suarez, Carl-Gustav Renmarker, Adam Dunkels, and Thiemo Voigt. Increasing zigbee network lifetime with x-mac. In *Proceedings of the workshop on Real-world wireless sensor networks*, REALWSN '08, pages 26–30, New York, NY, USA, 2008. ACM.

[105] Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora : scalable sensor network simulation with precise timing. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[106] Fouad A. Tobagi and Leonard Kleinrock. Packet switching in radio channels : Part ii–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution. *IEEE Transactions on Computing*, 23(12) :1417–1433, 1975.

[107] Wikipédia. Round-robin (informatique) — wikipédia, l'encyclopédie libre, 2012. [En ligne ; Page disponible le 11-mai-2012].

[108] Ning Xu, Sumit Rangwala, Krishna Kant Chintalapudi, Deepak Ganesan, Alan Broad, Ramesh Govindan, and Deborah Estrin. A wireless sensor network for structural monitoring. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 13–24, New York, NY, USA, 2004. ACM.

[109] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 70–84, New York, NY, USA, 2001. ACM.

[110] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3) :9–18, September 2002.

[111] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proc. INFOCOM 2002*, New York, June 2002. IEEE.

[112] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Comput. Netw.*, 52 :2292–2330, August 2008.

[113] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing : A recursive data dissemination protocol for wireless sensor networks. Technical report, University of California at Los Angeles, May 2001.

[114] Elzbieta Zielinska and Krzysztof Szczypiorski. Direct sequence spread spectrum steganographic scheme for ieee 802.15.4. *CoRR*, abs/1107.4230, 2011.

Chapitre 8

Annexe A : NS-2 : Tutoriel d'utilisation

8.1 Introduction

NS-2 comme présenté dans la thèse est un outil logiciel de simulation de réseaux informatiques à événements discrets. Il est principalement bâti avec les idées de la conception par objets, de ré-utilisabilité du code et de modularité. C'est un logiciel libre qu'on retrouve facilement sur Internet. Le logiciel est exécutable sous Linux, MAC et Windows (en utilisant CygWin). Ici nous donnerons des astuces qui ne vise pas à remplacer la documentation de NS mais plutôt à présenter cet outil au lecteur avec un éclairage différent et plus général pour en faciliter la prise en main.

NS-2 au départ a été conçu pour faciliter l'étude de l'interaction entre les protocoles et le comportement d'un réseau à différentes échelles. Pour cela, il contient des bibliothèques pour la génération de topologies réseaux, des trafics, ainsi qu'un outil de visualisation tel que l'animateur réseau NAM (network animator). Il est maintenant un outil bien adapté aux réseaux à communications de paquets et à la réalisation de simulations de petite taille. Il contient les fonctionnalités nécessaires à l'étude des algorithmes de routage unipoint ou multipoint, des protocoles de transport, de session, de réservation, des services intégrés, des protocoles d'application comme HTTP. De plus le simulateur possède déjà une palette de systèmes de transmission, d'ordonnanceurs et de politiques de gestion de files d'attente pour effectuer des études de contrôle de congestion. On retrouve dans le tableau suivant les principaux composants disponible dans NS par catégorie :

Application	Web, ftp, telnet, générateur de trafic (CBR, POISSON, ...)
Transport	TCP, UDP, RTP, SRM,...
Routage	DIFFUSION/RATE, DIFFUSION/-PROB, DSDV, DSR, FLOODING, OMNIMCAST, AODV, AOMDV, TORA, PUMA
Gestion de file d'attente	RED, DropTail, Token bucket
Discipline de service	CBQ, SFQ, DRR, Fair queueing
Système de transmission	CSMA/CD, CSMA/CA, lien point à point

En combinant tous les composants, ces capacités ouvrent le champ à l'étude de nouveaux mécanismes au niveau des différentes couches de l'architecture réseau. NS est devenu l'outil de référence pour les chercheurs du domaine.

8.2 Notions pour l'interpréteur

NS-2 est un langage écrit en C++, avec un interpréteur OTcl. Dans ce paragraphe nous allons donner les bases du langage Tcl, les principes de l'OTcl et les explications sur le mécanisme qui lie le C++ avec l'interpréteur Tcl.

8.2.1 Tcl

Tcl (Tool Command Language) est un langage de commande comme le shell UNIX mais qui sert à contrôler les applications. Il offre des structures de programmation telles que les boucles, les procédures ou les notions de variables. Il y a deux principales façons de se servir de Tcl : soit comme un langage autonome interprété ou comme une interface applicative d'un programme classique écrit en C ou C++. Toutes les applications qui utilisent Tcl créent et utilisent un interpréteur Tcl. Cet interpréteur est le point d'entrée standard de la bibliothèque. L'application *tclsh* constitue une application minimale ayant pour but de familiariser un utilisateur au langage Tcl et ne comporte que l'interpréteur Tcl. On retrouve cet interpréteur dans l'application NS. Une fois la commande "ns" tapée, l'application effectue l'initialisation des objets puis passe en mode interactif où on peut alors commencer à entrer les commandes Tcl.

Concepts

Tcl est un langage non typé où chaque commande consiste en un ou plusieurs mots séparés par des espaces ou des tabulations. Tous les mots sont des chaînes de caractères. Le premier mot de la commande est le nom de la commande, les autres mots sont les arguments passés à la commande. Chaque commande Tcl retourne le résultat sous forme d'une chaîne de caractères. Le caractère de "*retour à la ligne*" termine une commande et lance son interprétation. Le caractère de séparation de plusieurs commandes sur une même ligne est " ;".

A l'inverse du C ou C++, Tcl n'est pas un langage compilé, mais un langage interprété. Tcl évalue une commande en effectuant une analyse syntaxique et son exécution. L'analyse syntaxique consiste à identifier les mots et effectuer les substitutions. Durant cette étape, l'interpréteur ne fait que des manipulations de chaînes. Il ne traite pas la signification des mots. Pendant la phase d'exécution, l'aspect sémantique des mots est traité comme par exemple déduire du premier mot le nom de la commande, vérifier si la commande existe et appeler la procédure de cette commande avec les arguments. Le backslash (\) et les groupages permettent d'insérer des caractères spéciaux dans les mots et d'écrire des commandes sur plusieurs lignes.

Exemple

set a 12	affecte la valeur 12 à la variable a
expr 2 + 3	calcule la valeur de l'expression 2 + 3
puts Coucou	affiche Coucou sur la sortie standard

Un tutoriel détaillé sur les commandes est disponible à cette adresse http://wfr.tcl.tk/1140

8.2.2 OTcl

OTcl est une extension orientée objet de Tcl ftp://ftp.tns.lcs.mit.edu/pub/otcl/. Les commandes Tcl sont appelées pour un objet. En OTcl, les classes sont également des objets avec des possibilités d'héritage. Les correspondances avec le C++ sont :

- C++ a une unique déclaration de classe. En OTcl, les méthodes sont attachées à un objet ou à une classe.
- Les méthodes OTcl sont toujours appelées avec l'objet en préfixe.
- L'équivalent du constructeur et destructeur C++ en OTcl sont les méthodes $init\{\}destroy\{\}$
- L'identification de l'objet lui-même : *this*(C++), *$self*(OTcl). $self s'utilise à l'intérieur d'une méthode pour référencer l'objet lui-même. A la différence de C++, il faut toujours utiliser $self pour appeler une autre méthode sur le même objet. C'est à dire "*$self xyz 5*" serait "$this->xyz(5)$" ou juste "*xyz(5)*" en C++.
- L'héritage multiple est possible dans les deux langages.

8.2.3 Lien C++ et Tcl

Construire une application avec un interpréteur Tcl revient à inclure une bibliothèque Tcl qui définit les commandes de bases de Tcl dans l'application. Comme nous l'avons dit, l'interpréteur effectue l'analyse syntaxique et appelle la fonction C correspondant à la commande Tcl. Ajouter une commande Tcl consiste à établir un lien entre un mot et une fonction C. Le mot sera le nom de la commande Tcl. La fonction C est définie dans le code source de l'application. Au démarrage, l'application procède dans son *main*() aux initialisations nécessaires et passe la main à l'interpréteur. L'application passe en mode interactif : à chaque commande tapée par

l'utilisateur, la fonction C correspondante est appelée afin de réaliser la commande demandée.

8.3 Installation

L'installation de NS-2 est ici détaillé pour une version d'Ubuntu 11.04. Selon moi, une installation via *apt-get* n'est pas adaptée si on compte modifier les fichiers C++. Il est plutôt conseillé de faire une installation via **ns-allinone** que l'on trouve facilement sur Google la dernière version en date est la version **ns-allinone-2.35-RC10**. Elle contient l'essentiel pour faire fonctionner NS notamment :

- ns,
- nam (visualisation),
- otcl-1.14,
- tcl8.5.8,
- tclcl-1.20,
- tk8.5.8,
- zlib-1.2.3,
- xgraph-12.2 (pour faire des graphiques).

Avant l'installation, il est nécessaire d'installer les dépendances nécessaires en effectuant cette commande

sudo apt-get install build-essential autoconf automake libxmu-dev

Néanmoins à cause de l'évolution du compilateur gcc, plusieurs erreurs peuvent être observées pendant la compilation ou à l'exécution de NS. D'où il sera nécessaire dans certains cas d'installer une version antérieure de gcc et g++ la version 4.5.2 inclue dans la Ubuntu 11.04 fonctionne parfaitement sans erreur. N'oubliez pas d'installer g++... Si vous êtes amenés à changer la version du compilateur, il vous sera indispensable de modifier certains "***Makefile.in***" dans certains dossiers :

- otcl-1.14/ : CC = gcc-4.5
- tclcl-1.20/ : CC = gcc-4.5
- tclcl-1.20/ : CPP = g++-4.5
- ns-2.35/ : CC = gcc-4.5
- ns-2.35/ : CPP = g++-4.5
- nam-1.15/ : CC = gcc-4.5
- nam-1.15/ : CPP = g++-4.5

Il est conseillé d'installer NS dans le "home", pour cela décompressez le fichier ns-allinone-2.35-RC10 dans le home. D'autres bugs peuvent survenir pendant ou après l'installation de NS si c'est le cas il est nécessaire d'effectuer les changements pour enlever x : :X constructeur

D'abord positionnez vous dans le répertoire cd $HOME/ns-allinone-2.35-RC10 puis :

- Editez le fichier ns-2.35/tools/ranvar.cc et changez la ligne 219 :
  $return\ GammaRandomVariable :: GammaRandomVariable(1.0+alpha\_, beta\_).value()*pow\ (u, 1.0/alpha\_);$
  par
  $return\ GammaRandomVariable(1.0+alpha\_,\ beta\_).value()*pow\ (u, 1.0/alpha\_);$

➢ De même, changez les lignes 183 et 185 dans ns-2.35/mobile/nakagami.cc :
$resultPower = ErlangRandomVariable(Pr/m, int\_m).value()$;
et
$resultPower = GammaRandomVariable(m, Pr/m).value()$;

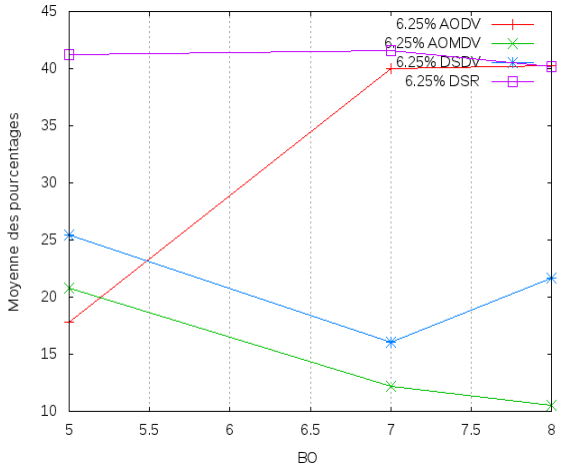
Une fois toutes ces modifications terminés, lancez l'installation en effectuant :

```
cd $HOME/ns-allinone-2.35-RC10
sudo ./install
```

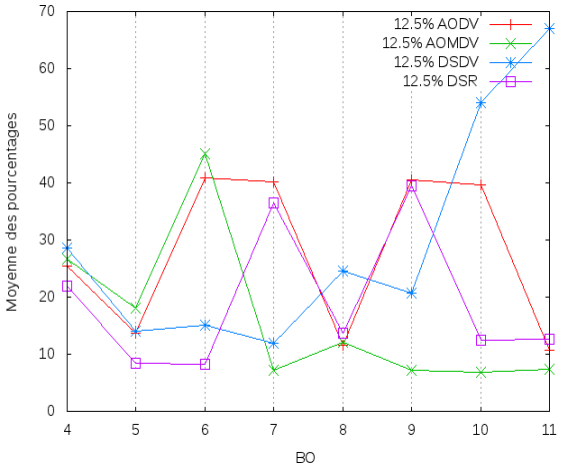
Chapitre 9

Annexe B : Graphique de la classification

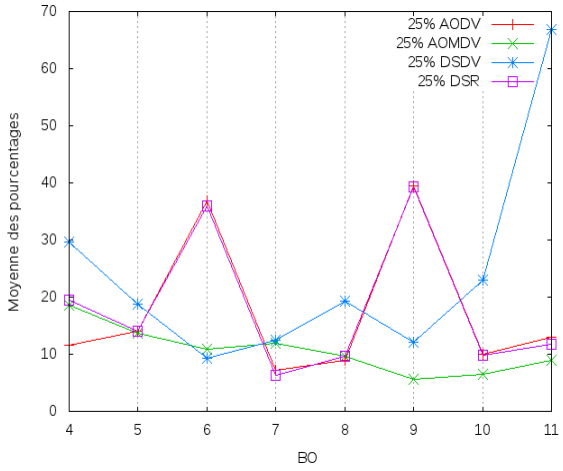
Liste des différents graphiques obtenus en effectuant la moyenne du taux de perte et des paquets perdus :



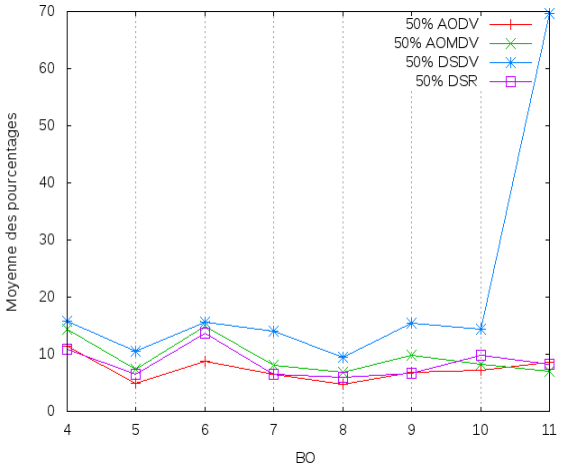
(a) 6.25% de durée active



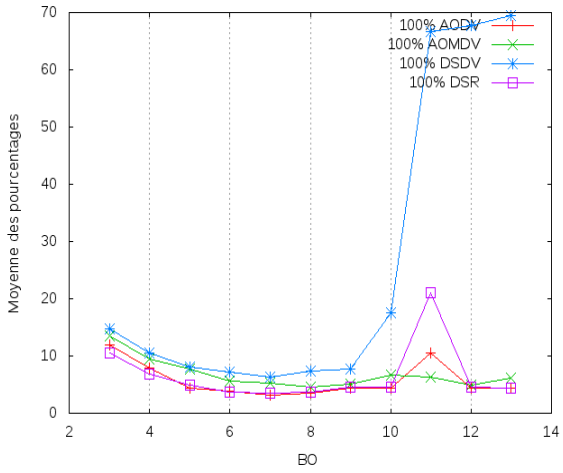
(b) 12.5% de durée active



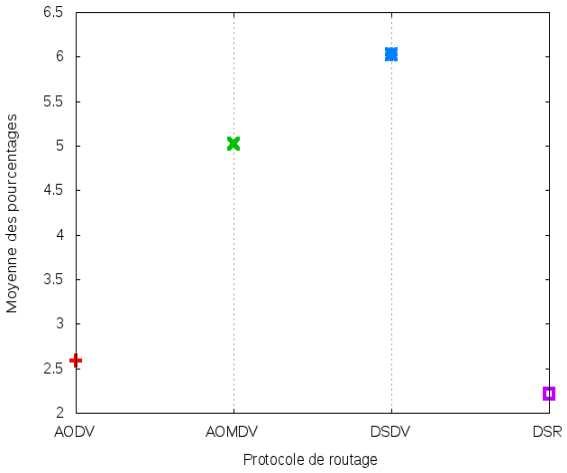
(c) 25% de durée active



(d) 50% de durée active

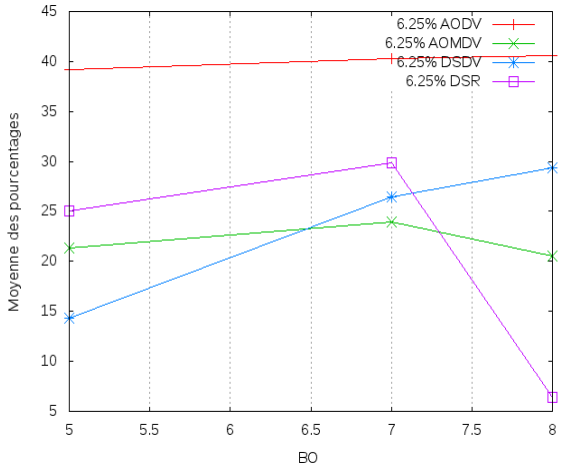


(e) 100% de durée active

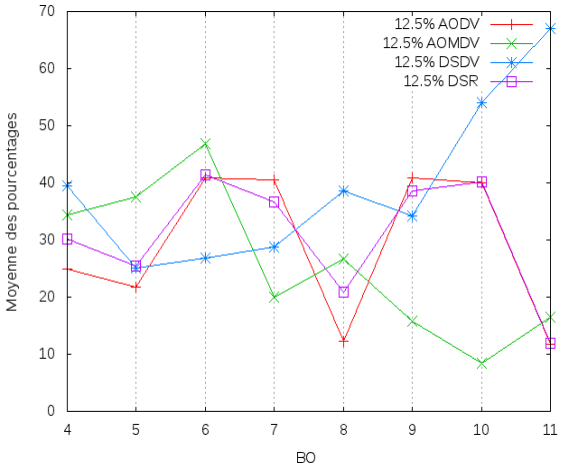


(f) Mode sans balises

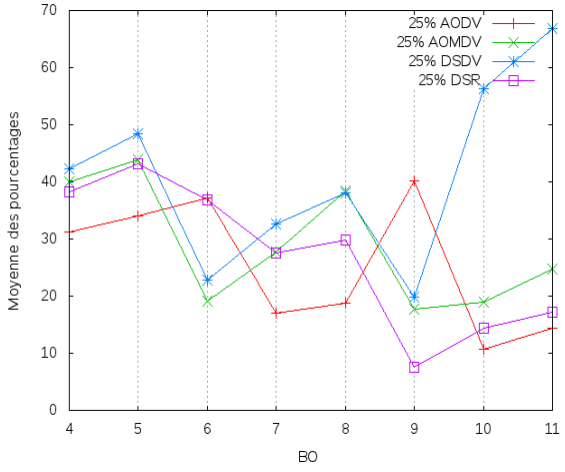
FIGURE 9.1 – Résultat pour une application régulière avec 25 noeuds



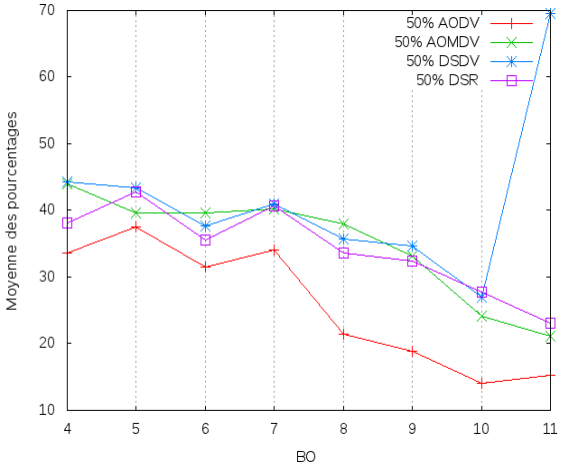
(a) 6.25% de durée active



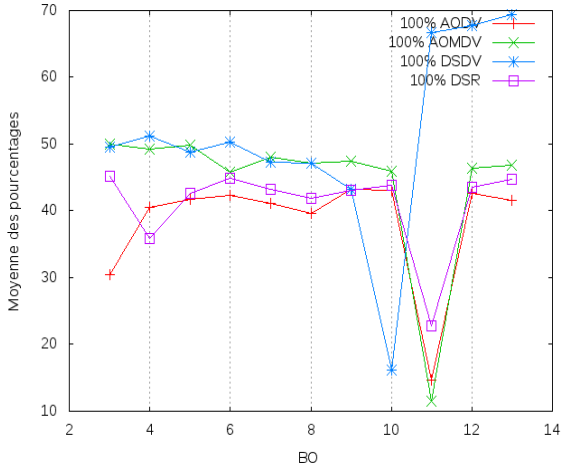
(b) 12.5% de durée active



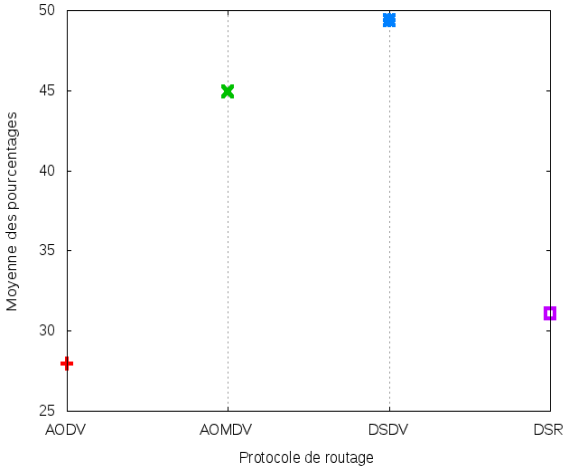
(c) 25% de durée active



(d) 50% de durée active

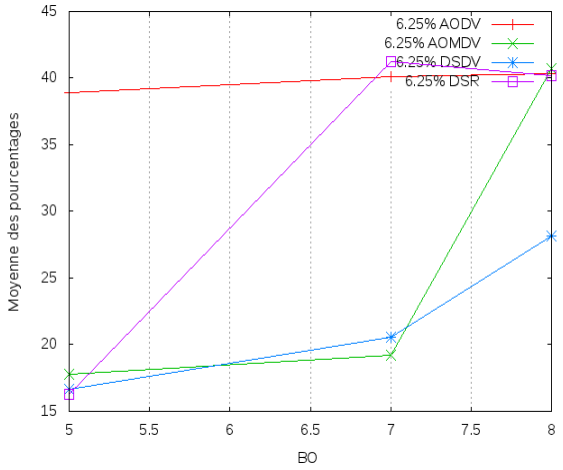


(e) 100% de durée active

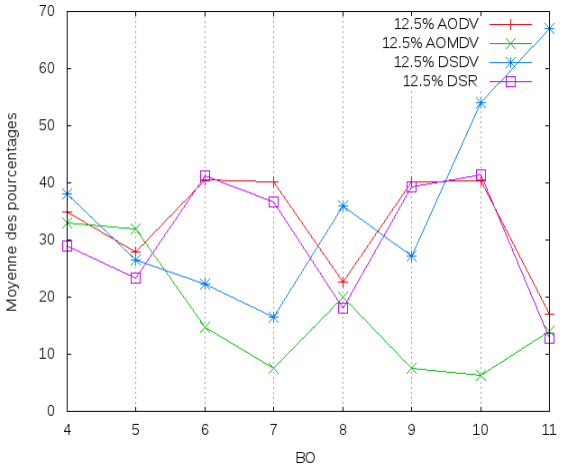


(f) Mode sans balises

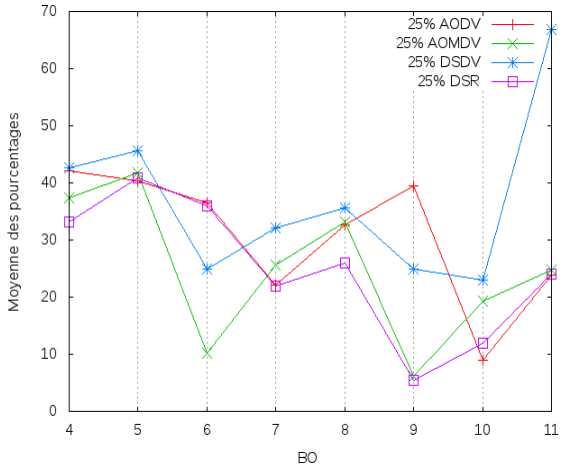
FIGURE 9.2 – Résultat pour une application avec un fort débit avec 25 noeuds



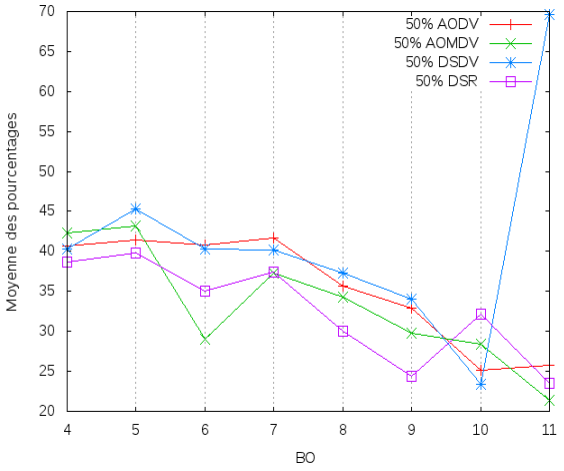
(a) 6.25% de durée active



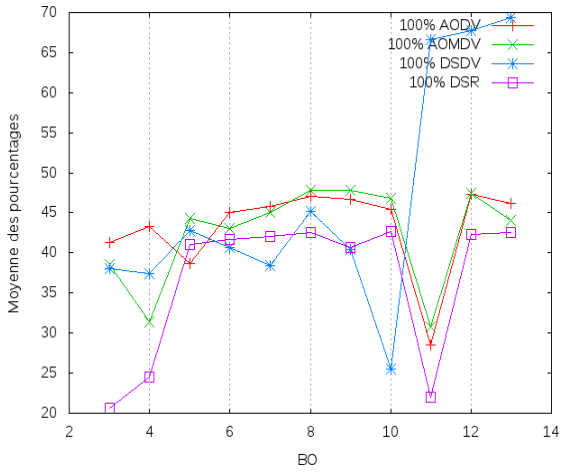
(b) 12.5% de durée active



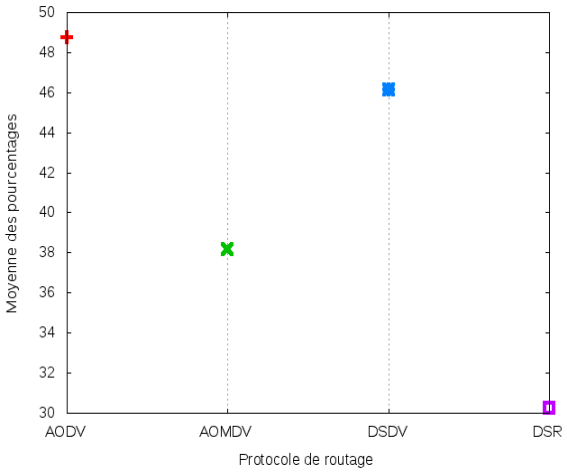
(c) 25% de durée active



(d) 50% de durée active

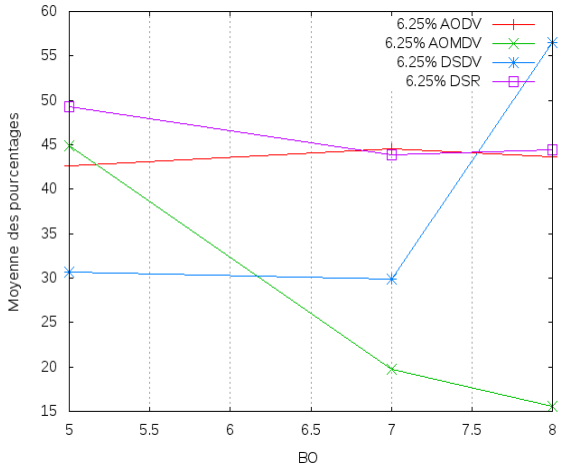


(e) 100% de durée active

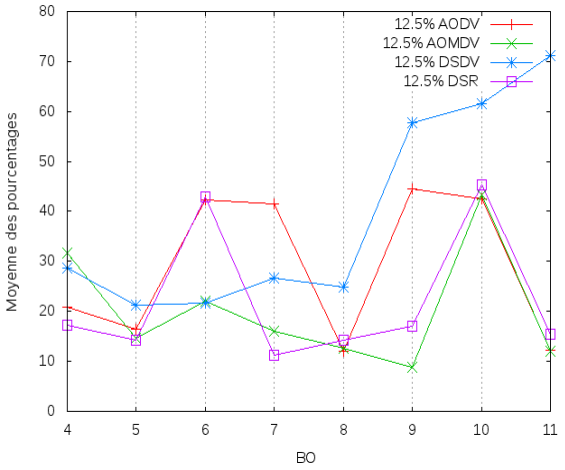


(f) Mode sans balises

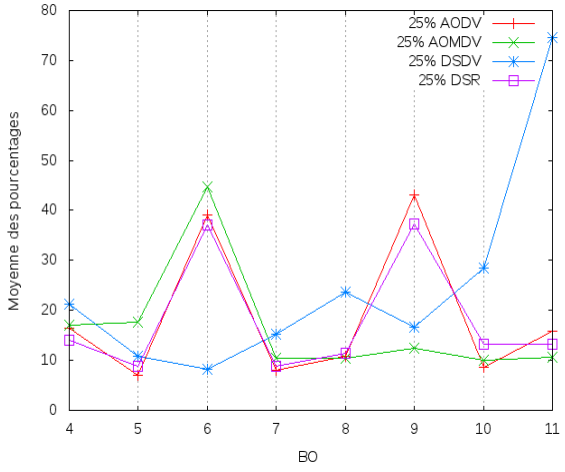
FIGURE 9.3 – Résultat pour une application par à coup avec 25 noeuds



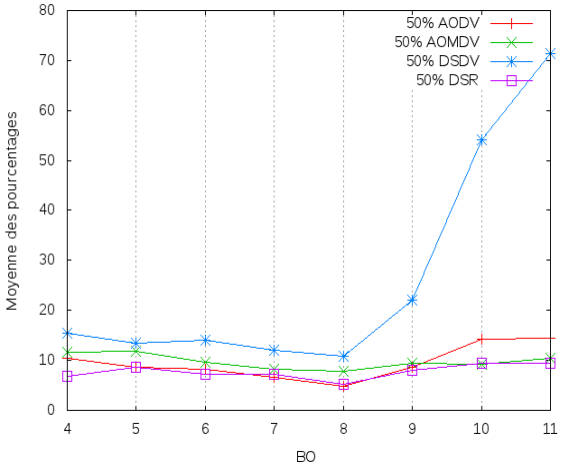
(a) 6.25% de durée active



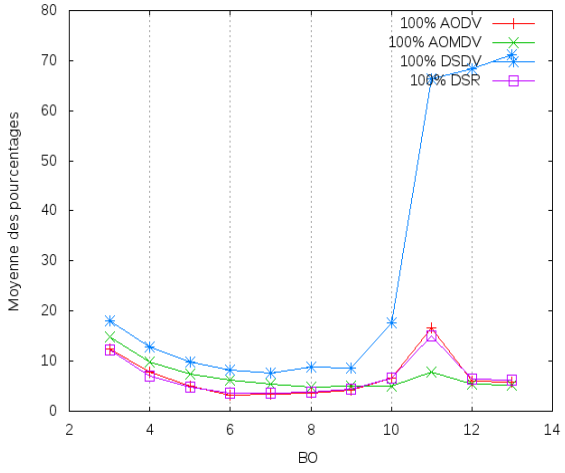
(b) 12.5% de durée active



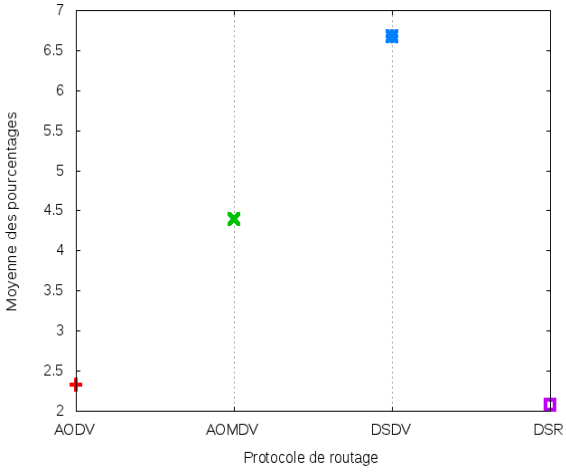
(c) 25% de durée active



(d) 50% de durée active

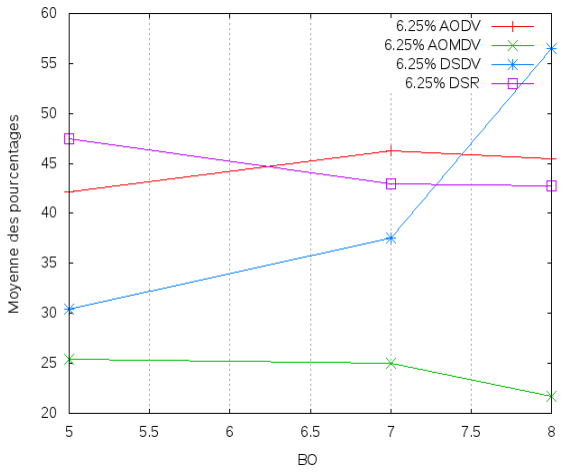


(e) 100% de durée active

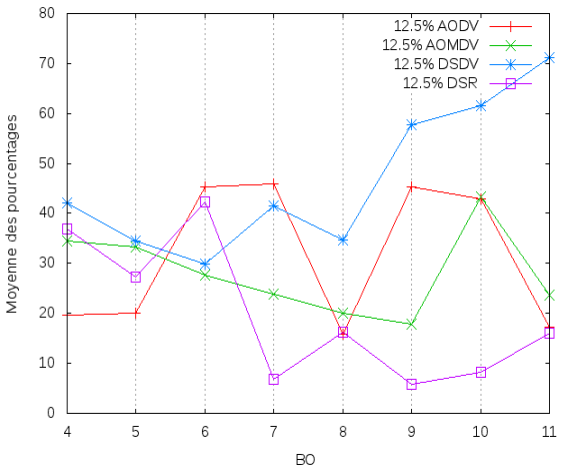


(f) Mode sans balises

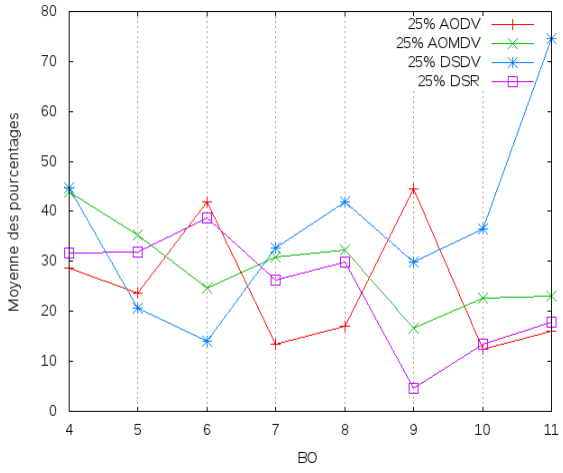
FIGURE 9.4 – Résultat pour une application régulière avec 60 noeuds



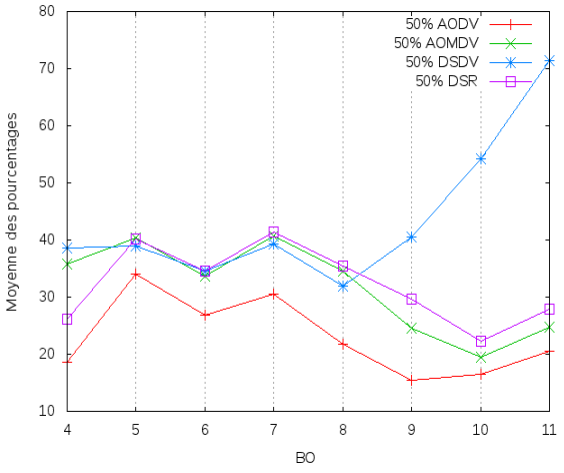
(a) 6.25% de durée active



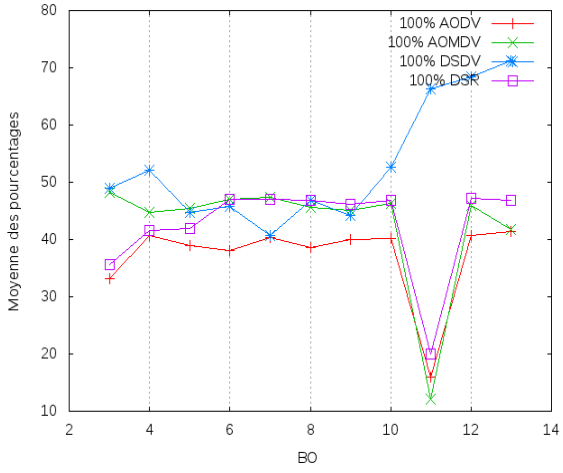
(b) 12.5% de durée active



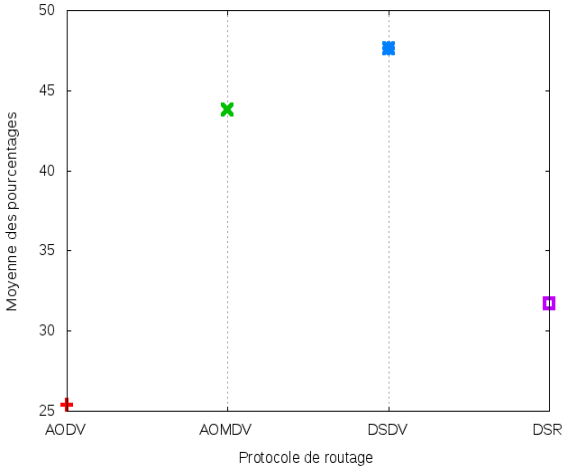
(c) 25% de durée active



(d) 50% de durée active

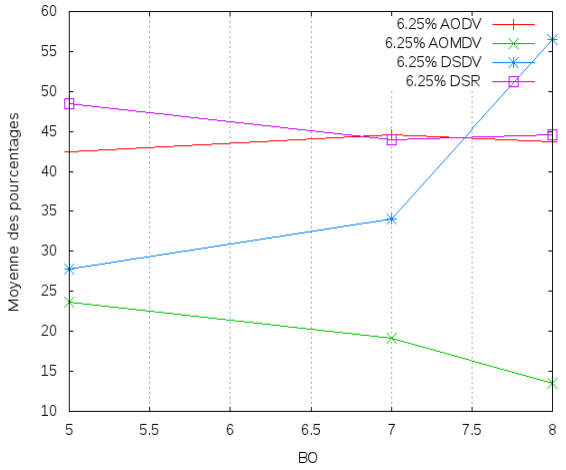


(e) 100% de durée active

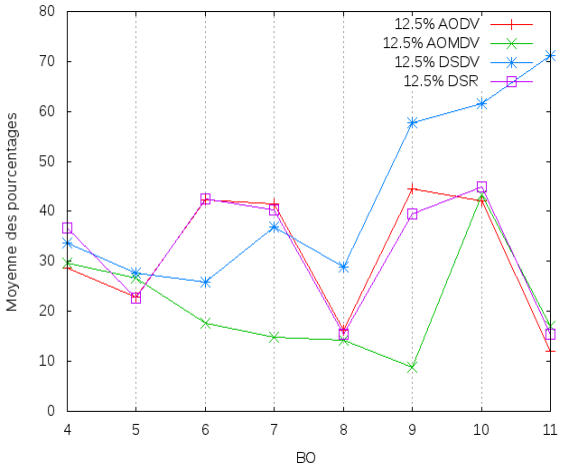


(f) Mode sans balises

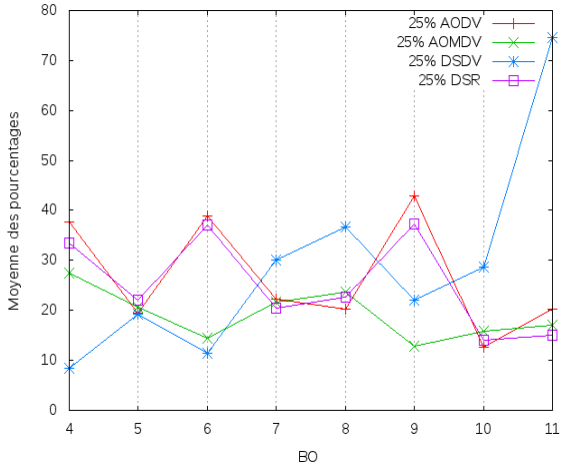
FIGURE 9.5 – Résultat pour une application avec un fort débit avec 60 noeuds



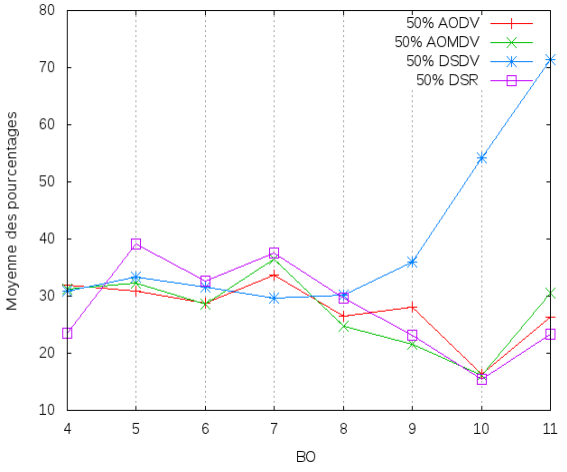
(a) 6.25% de durée active



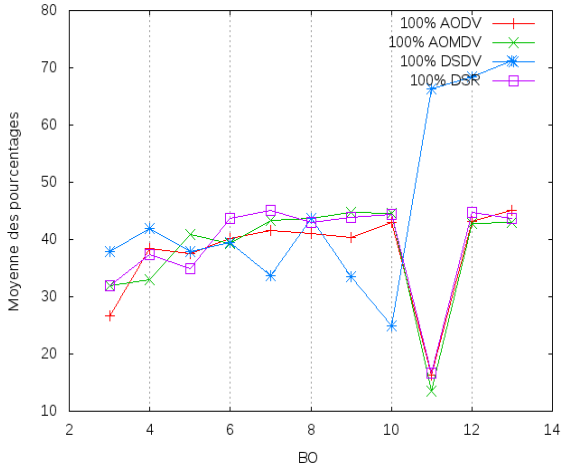
(b) 12.5% de durée active



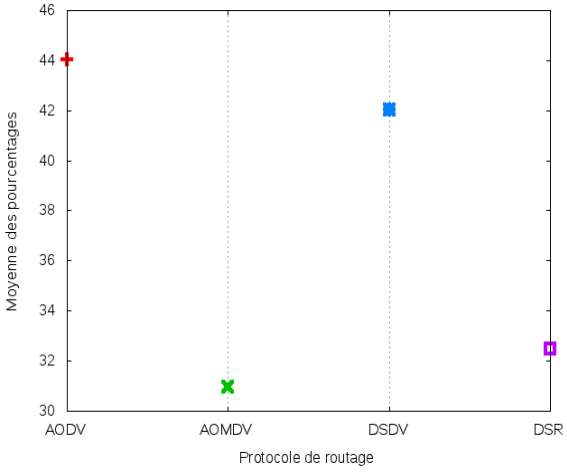
(c) 25% de durée active



(d) 50% de durée active

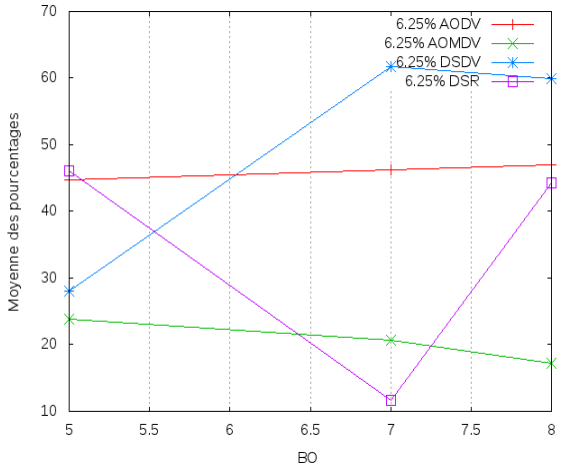


(e) 100% de durée active

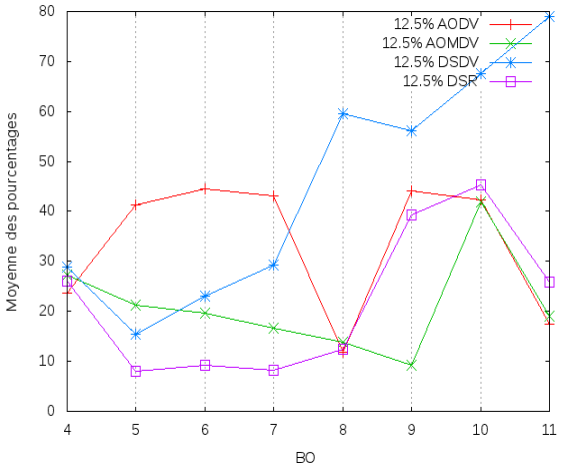


(f) Mode sans balises

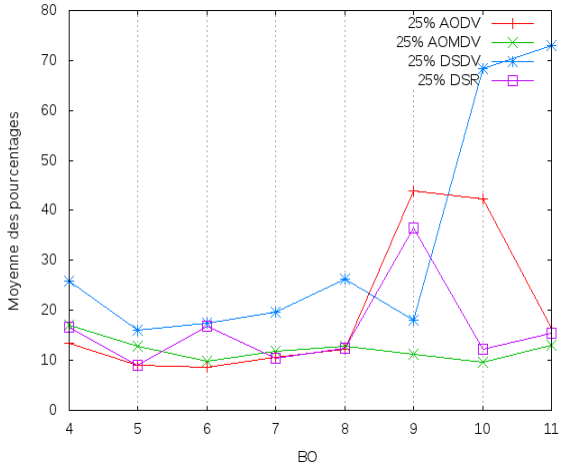
FIGURE 9.6 – Résultat pour une application par à coup avec 60 noeuds



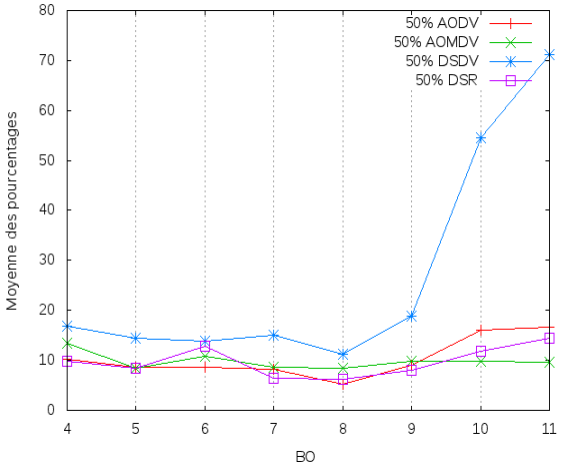
(a) 6.25% de durée active



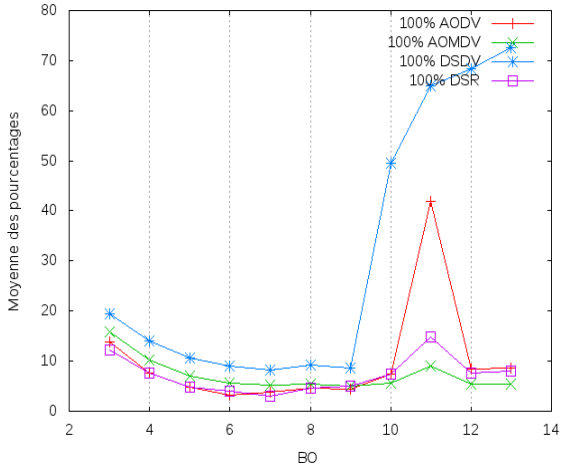
(b) 12.5% de durée active



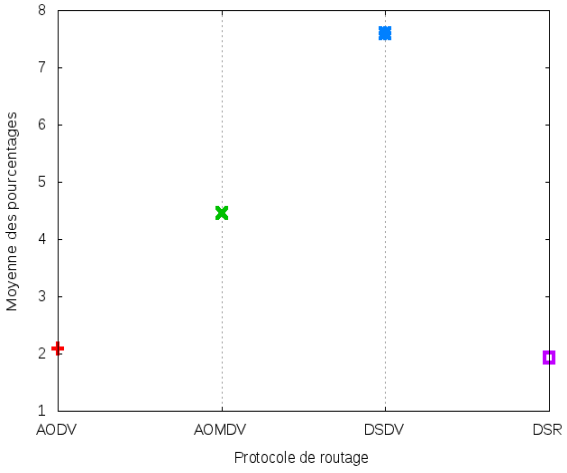
(c) 25% de durée active



(d) 50% de durée active

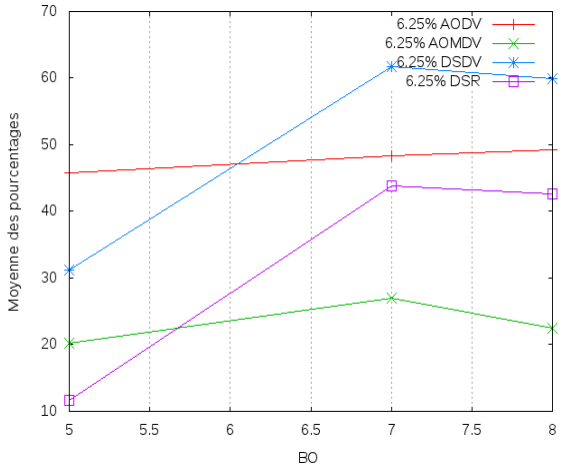


(e) 100% de durée active

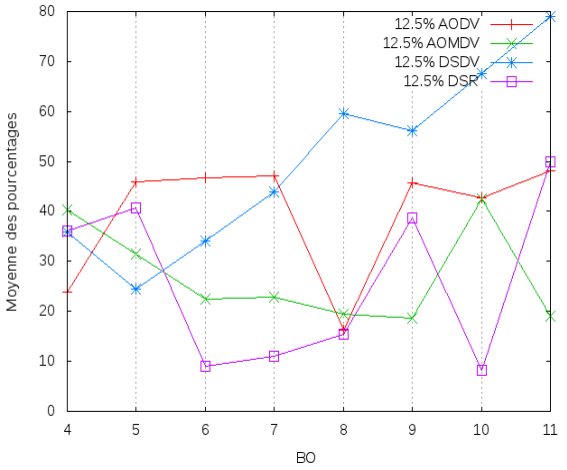


(f) Mode sans balises

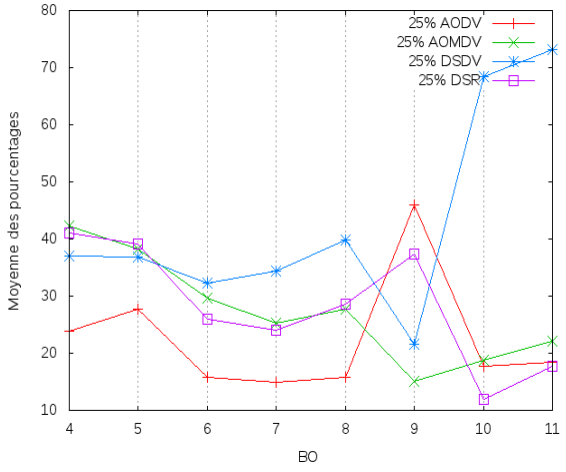
FIGURE 9.7 – Résultat pour une application régulière avec 100 noeuds



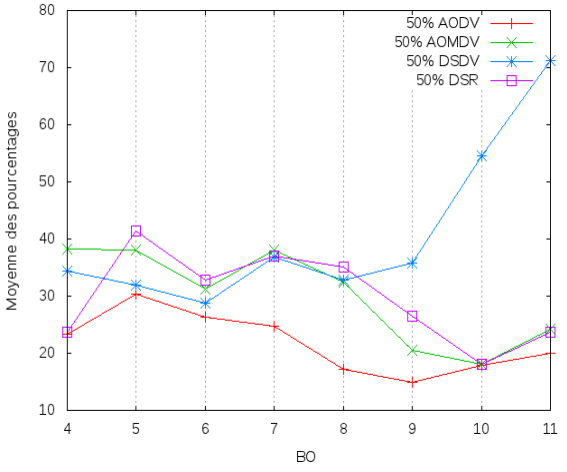
(a) 6.25% de durée active



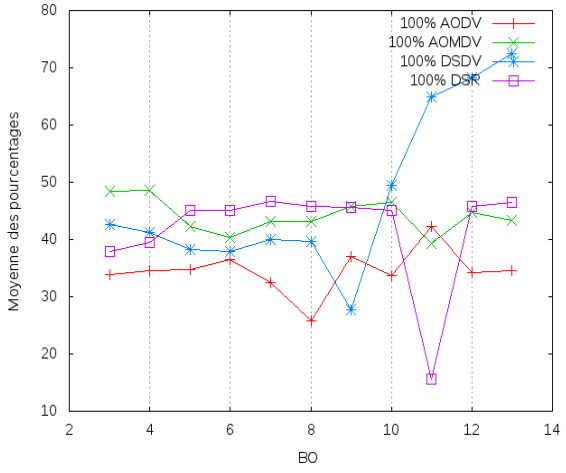
(b) 12.5% de durée active



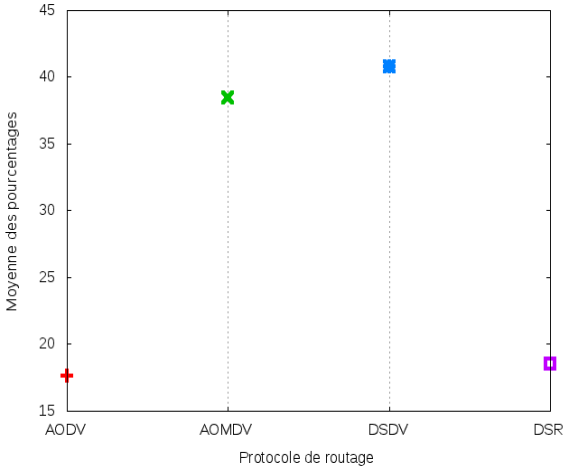
(c) 25% de durée active



(d) 50% de durée active

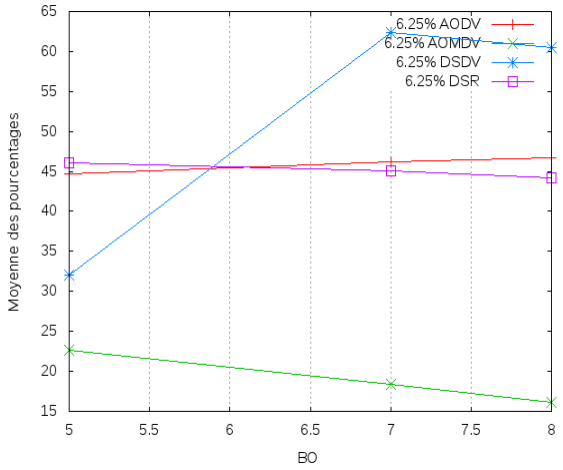


(e) 100% de durée active

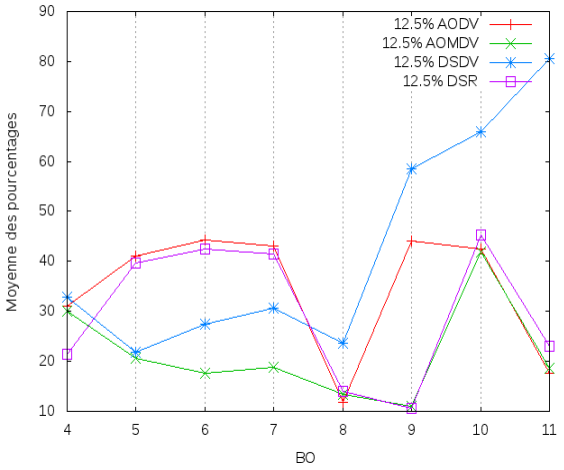


(f) Mode sans balises

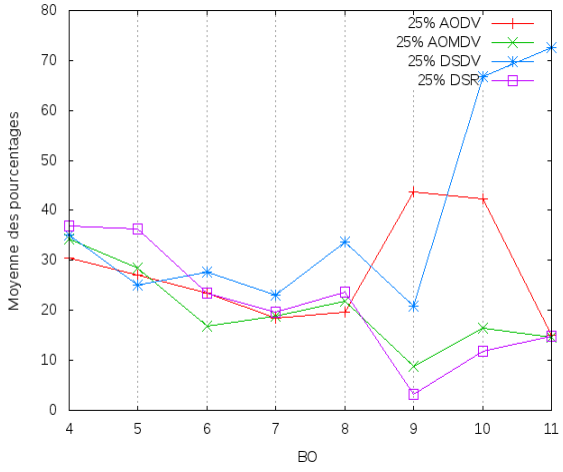
FIGURE 9.8 – Résultat pour une application avec un fort débit avec 100 noeuds



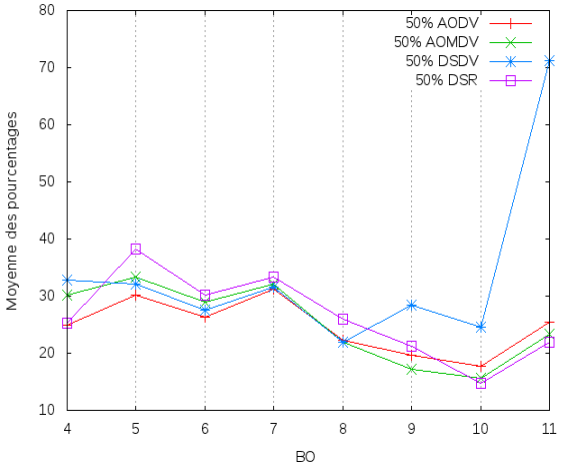
(a) 6.25% de durée active



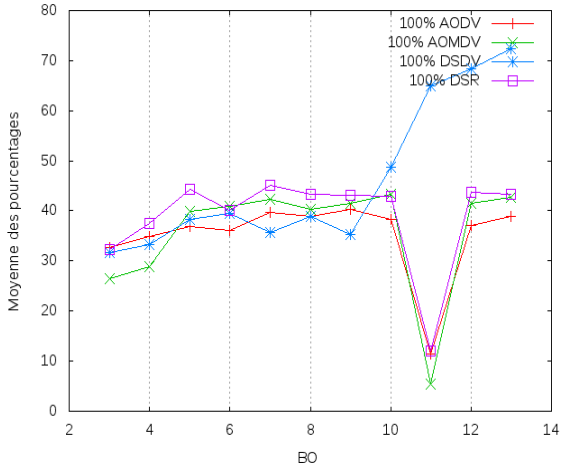
(b) 12.5% de durée active



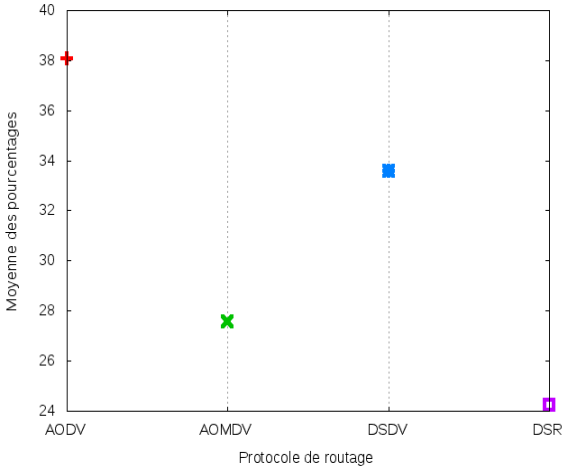
(c) 25% de durée active



(d) 50% de durée active

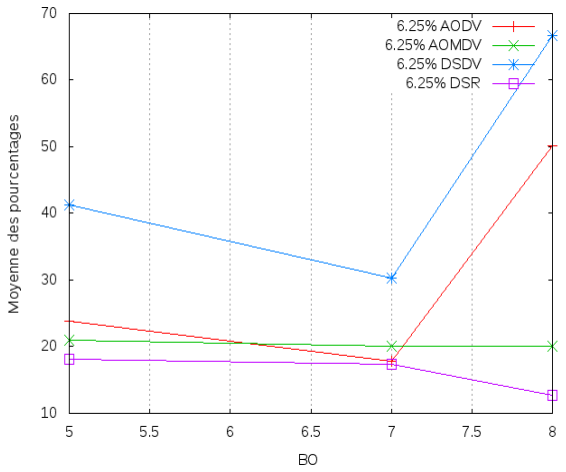


(e) 100% de durée active

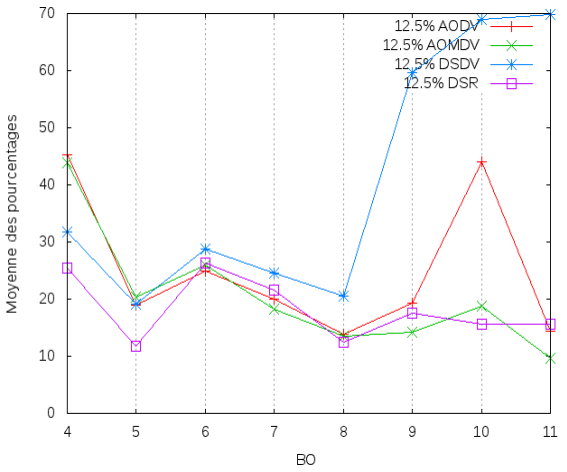


(f) Mode sans balises

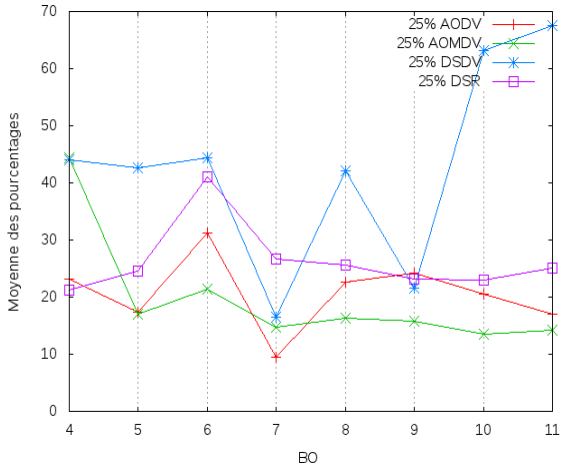
FIGURE 9.9 – Résultat pour une application par à coup avec 100 noeuds



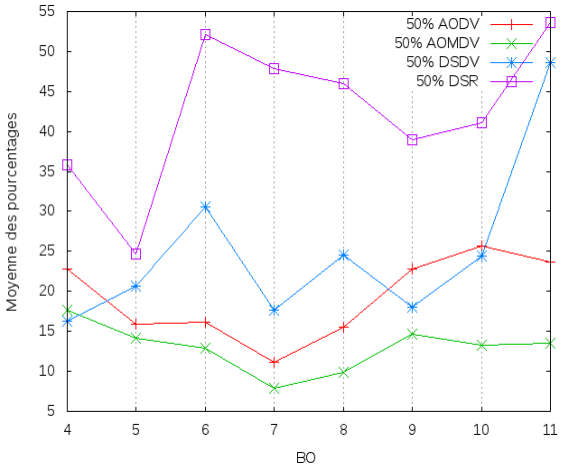
(a) 6.25% de durée active



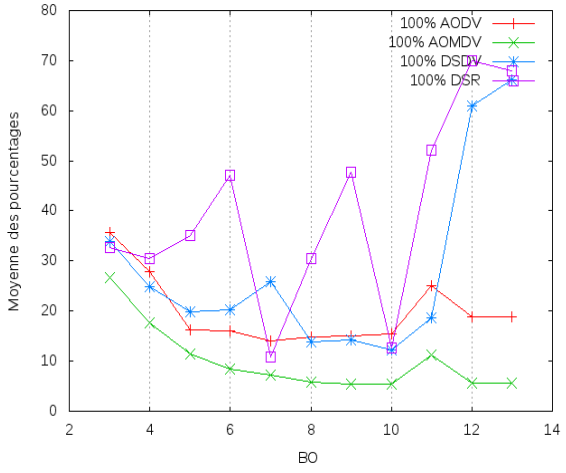
(b) 12.5% de durée active



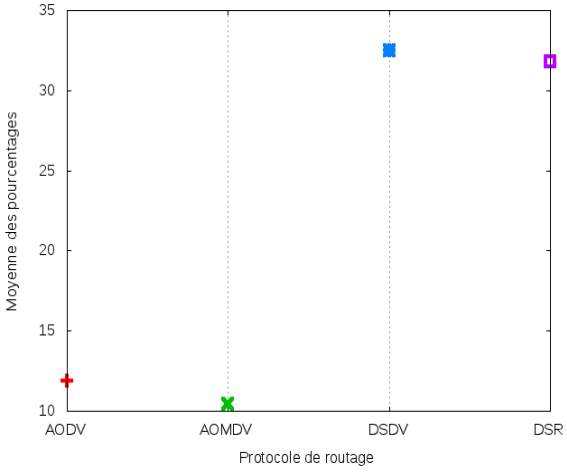
(c) 25% de durée active



(d) 50% de durée active

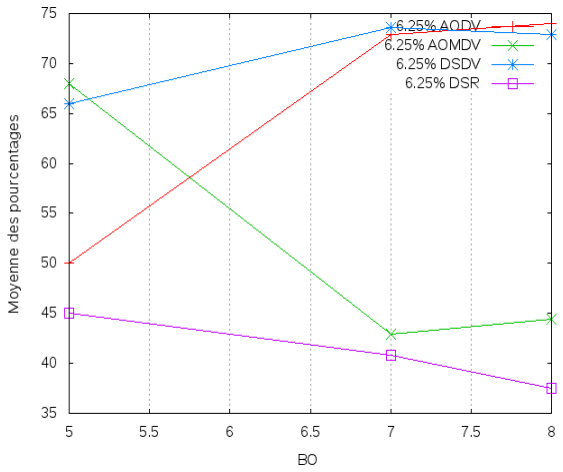


(e) 100% de durée active

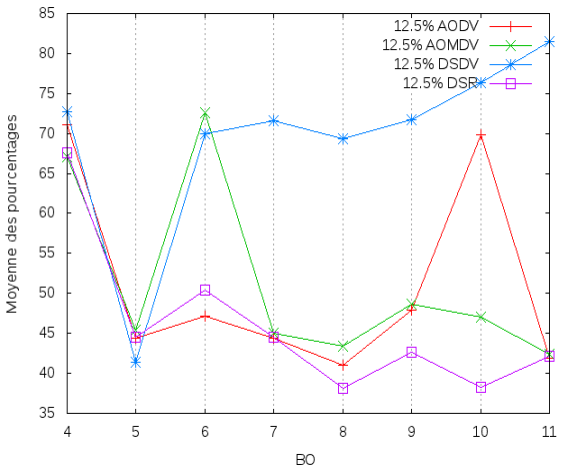


(f) Mode sans balises

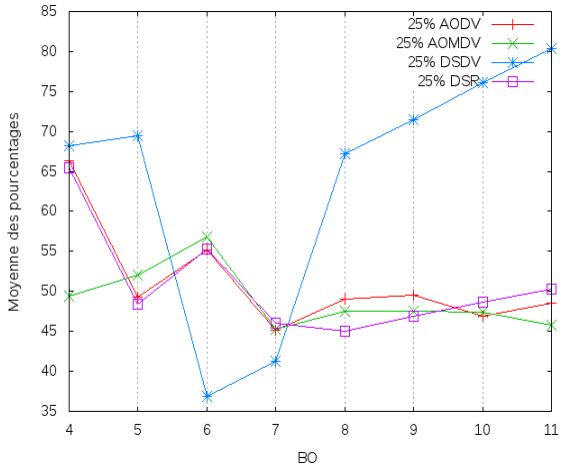
FIGURE 9.10 – Résultat pour une application régulière avec 300 noeuds



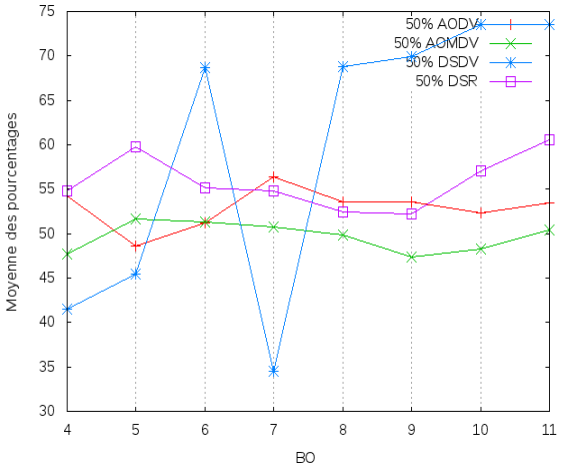
(a) 6.25% de durée active



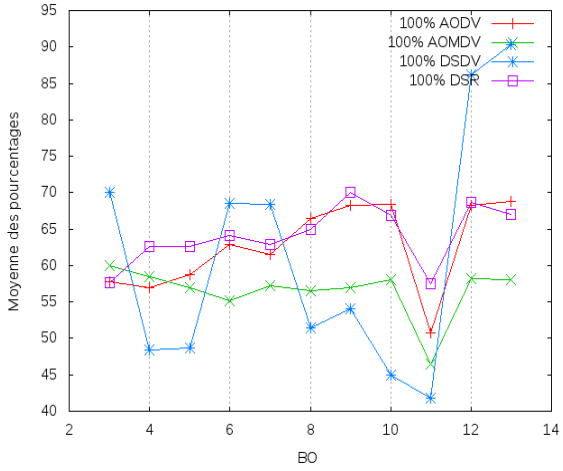
(b) 12.5% de durée active



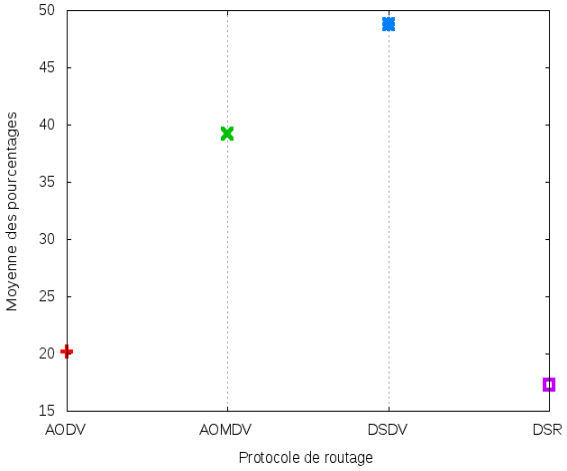
(c) 25% de durée active



(d) 50% de durée active

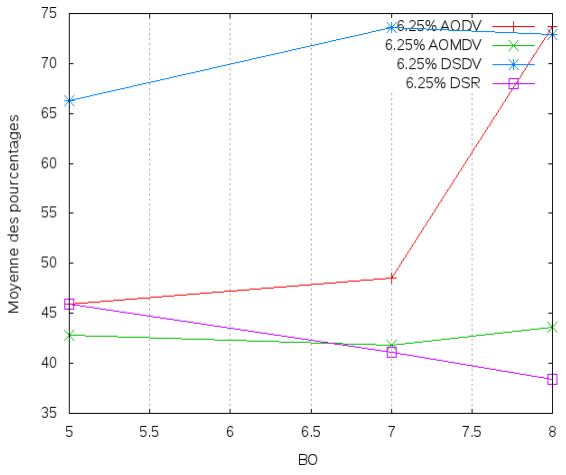


(e) 100% de durée active

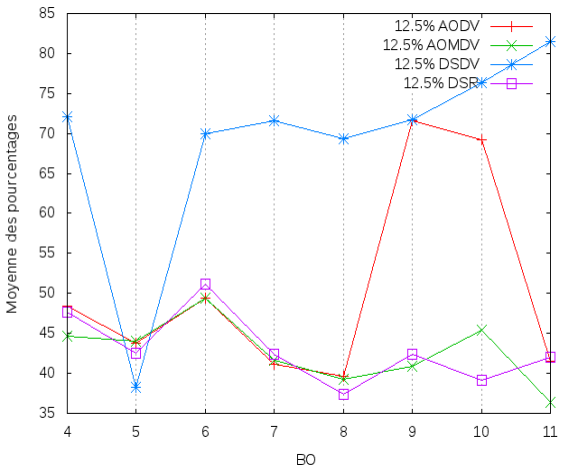


(f) Mode sans balises

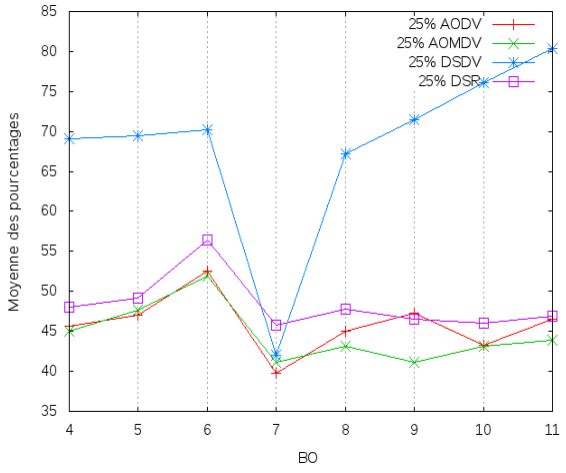
FIGURE 9.11 – Résultat pour une application avec un fort débit avec 300 noeuds



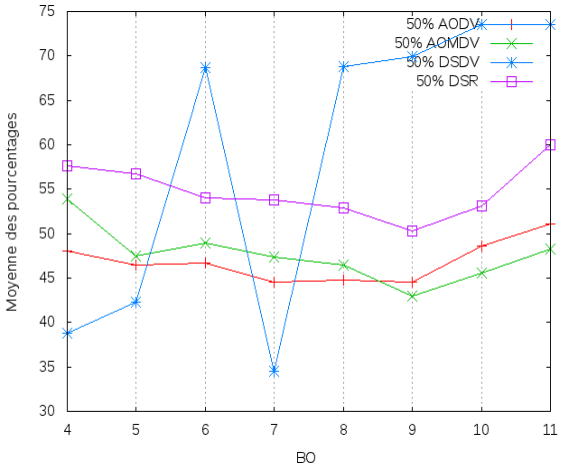
(a) 6.25% de durée active



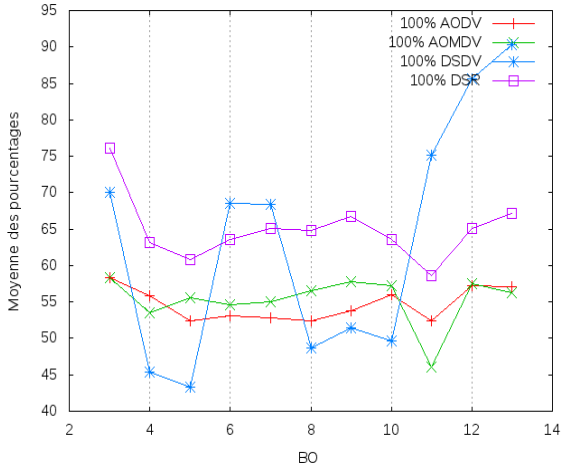
(b) 12.5% de durée active



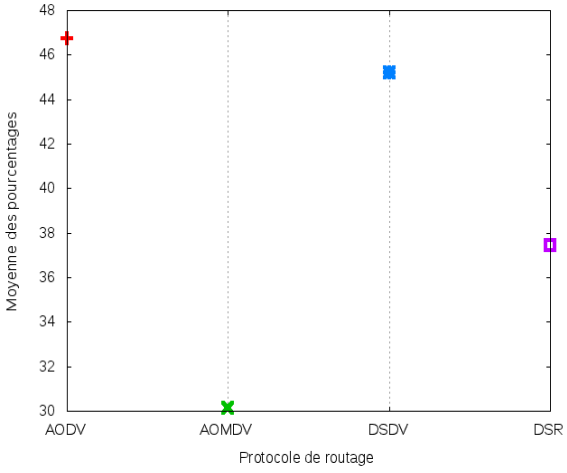
(c) 25% de durée active



(d) 50% de durée active



(e) 100% de durée active



(f) Mode sans balises

FIGURE 9.12 – Résultat pour une application par à coup avec 300 noeuds

Table des figures

Résumé

Les réseaux de capteurs constituent un axe de recherche très fertile ces dernières années. Cette technique se développe dans différents domaines comme l'environnement, l'industrie, le commerce, la médecine, l'armée, etc. Les réseaux de capteurs sont difficiles à concevoir parce qu'ils sont fortement contraints en énergie et que tous les éléments ont potentiellement une influence sur la durée de vie du système. Nous proposons un outil permettant l'aide au bon paramétrage et aux choix de paramètres optimaux pour la fiabilité des applications.

Dans cette thèse, nous nous sommes intéressés à deux problématiques : une classification des paramètres pour un outil d'aide à la décision pour la configuration d'un réseau de capteurs et la seconde, un outil de test de conformité du système dans un environnement réel. Le document est divisé en deux parties où la première partie est un état de l'art de différents protocoles existants et la deuxième partie décrit notre contribution dans ces deux problématiques.

Dans la première contribution, nous avons analysé l'impact de la couverture radio puis de la topologie sur les performances d'un réseau de capteurs. Nous étudions le taux de perte et le niveau d'énergie pour en déduire la fiabilité d'une application. Puis nous avons proposé une étude menant à une classification pour notre outil d'aide à la décision. Notre classification est basée sur une étude de divers paramètres de la couche MAC, physique, protocole de routage, nombre de noeuds et type d'application.

Dans la deuxième contribution, nous nous sommes focalisés sur une approche pragmatique permettant de tester la conformité d'un réseau de capteurs dans un environnement réel. Pour tester la conformité dans un environnement réel, nous proposons une architecture d'exécution de test sur un réseau de capteurs réel. Ceci dans un but d'assurer un niveau correct de conformité et la fiabilité de celui-ci durant son fonctionnement.

Abstract

Wireless sensor networks is one of the hotest research topic in the last few years. This technology can be applied for different fields such as environment, industry, trading, medicine, military etc. Wireless sensor networks are hard to conceive because they require a lot of energy and because each of its component may have an influence on the lifetime of the whole system. What we suggest is a tool allowing to choose the correct and optimal parameters for the reliability of the applications.

In this thesis, we focused on two major problems : firstly, a classification of the parameters for a tool allowing to make decisions about the configuration of a wireless sensors network, and secondly, a tool testing the compliance of the system with a real environment. The document is divided into two parts : the first part states the different protocols that exist, and the second part describes our contributions to those topics.

In the first contribution, we analyzed how influential the radio cover and the network topology are on the network performances. Then, we deduced from the study of the loss rate and of the level of energy, the reliability of the application. Next, we suggested a study leading to a classification for our decision making tool. For this classification, we studied various parameters related to the MAC layer, the Physical layer, the network layer, the application layer the number of nodes involved in the network.

In the second contribution, we adopted a pragmatic approach so we could test the conformity of a wireless sensors network in a real environment. In order to test its conformity in a real environment, we suggested a structured test execution on a real wireless sensor network. This task has been suggested in order to check the conformance level of the network while it was working.