Conception d'un simulateur et de protocoles cross-layer pour les nanoréseaux électromagnétiques denses

Thierry Arrabal

UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

UFC
UNIVERSITÉ
DE FRANCHE-COMTÉ

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE FRANCHE-COMTÉ

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Informatique

par

THIERRY ARRABAL

Conception d'un simulateur et de protocoles cross-layer pour les nanoréseaux électromagnétiques denses

Thèse présentée et soutenue à Montbéliard, le 25 Novembre 2019

Composition du Jury :

SONG YE-QIONG	Professeur à l'Université de Lorraine	Rapporteur
BEYLOT ANDRÉ-LUC	Professeur à ENSEEIHT	Rapporteur
GUÉRIN-LASSOUS ISABELLE	Professeur à l'Université Claude Bernard Lyon 1	Présidente du jury
DEDU EUGEN	Maître de conférences HDR à l'Université de Franche-Comté	Directeur de thèse
DHOUTAUT DOMINIQUE	Maître de conférences à l'Université de Franche-Comté	Co-encadrant de thèse

N° | X | X | X |

école doctorale **sciences pour l'ingénieur et microtechniques**

Titre : Conception d'un simulateur et de protocoles cross-layer pour les nanoréseaux électromagnétiques denses

Mots-clés : nanoréseaux électromagnétiques, couche transport, congestion, simulation

Résumé :

La miniaturisation des équipements mène à l'apparition de machines de taille nanométrique. Ces nanomachines ont des ressources très limitées. Une modulation spécifique aux antennes nanométriques a été proposée dans la littérature. Elle est basée sur des pulses très courts (100 fs) et un grand espacement entre ces pulses. Il en résulte un canal à la bande passante très élevée (plusieurs Tb/s) avec un important le multiplexage temporel.

Les travaux menés durant cette thèse ont permis le développement d'algorithmes et de protocoles adaptés à cet environnement nouveau et très différents des environnements standards. Les contributions apportées concernent plusieurs couches réseau : une méthode d'endormissement qui tient compte des spécificités des communications par pulse et des échelles de temps impliquées ; un protocole d'estimation du nombre de voisins capable de gérer un large panel de densités. Un protocole de diffusion de données counter-based qui réduit drastiquement le nombre de forwardeurs tout en conservant une très forte couverture ; enfin, un algorithme de déviation de route visant à détecter et à contourner des points de congestion.

Les nanomachines n'ont pas encore été fabriquées, et nos problématiques concernent un très grand nombre de nœuds communicants ce qui peut induire des comportements émergeant difficiles à modéliser mathématiquement. Par conséquent, nous avons choisi de nous tourner vers la simulation et nous avons développé un simulateur spécialement conçu pour les nanoréseaux.

Title: Conception d'un simulateur et de protocoles cross-layer pour les nanoréseaux électromagnétiques denses

Keywords: electromagnetic nanonetworks, transport layer, congestion, simulation

Abstract:

The miniaturization of equipments leads to the arrival of nano-sized machines. These nanomachines have very limited resources. A specific modulation for nanometric antennas has been proposed in the literature. It is based on very short pulses (100 fs) and a large spacing between these pulses. This leads to a channel with a very high bandwidth (several Tb/s) and a high time multiplexing.

The work carried out during this thesis allowed the development of algorithms and protocols adapted to this new environment and very different from standard environments. The contributions made concern several network layers. A sleep method that takes into account the specificities of pulse communications and the time scales involved; a protocol for estimating the number of neighbours, capable of managing a wide range of densities; a counter-based data dissemination protocol that significantly reduces the number of forwarders while maintaining very high coverage; finally, a path deviation algorithm to detect and bypass congestion points.

Nanomachines have not been built yet, and our problems concern a very large number of communicating nodes, which can lead to emerging behaviours that are difficult to model mathematically. For all these reasons, we have chosen to turn to simulation. To do this, we have developed a simulator specially designed for nanonetworks.

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besanon, France

CONTEXTE DE LA THÈSE

J'ai effectué ma thèse à l'institut FEMTO-ST, Université de Bourgogne Franche-Comté en France dans la ville Montbéliard. J'ai travaillé au département DISC au sein de l'équipe OMNI, et j'ai été encadré par Dr Eugen Dedu et co-encadré par Dr Dominique Dhoutaut. Cette thèse s'est déroulée du 1 octobre 2016 au 30 septembre 2019.

Nous avons également eu une démarche de promotion et de valorisation de cette jeune thématique :

- Prise en charge et collaboration avec des stagiaires, collaboration avec Florian Büther, doctorant allemand, qui a mené à la publication d'un article commun, avec qui nous avons tissé des liens sur le long terme. Cette collaboration est toujours d'actualité aujourd'hui.
- Une présentation lors des journées non-thématiques du GDR RSD (Réseaux et Systèmes Distribués) nous a permis de faire connaître cette thématique à d'autres chercheurs du domaine des réseaux.
- La participation au concours d'éloquence "Ma thèse en 180 secondes" fut une expérience enrichissante et a joué un rôle dans la diffusion de nos travaux à un public non scientifique étant donné que le concours est ouvert à tous les visiteurs, mais aussi à plusieurs classes du secondaire avec qui j'ai eu le plaisir d'interagir et d'échanger sur le thème de la recherche et des nanoréseaux.

La thématique des nanoréseaux est une thématique bien connue de l'équipe OMNI, vu que des thèses ont déjà été menées sur le sujet. Muhammad Agus Zainuddin a notamment étudié le codage de l'information ainsi que la transmission d'images et de vidéos dans les nanocommunications afin de tirer partie des spécificités de ce nouveau type de réseaux. Nicolas Boillot a développé un simulateur de nanoréseaux qui a été d'une aide précieuse à mes encadrants et à moi-même pour développer un nouveau simulateur ne souffrant pas des mêmes limites. La matière programmable ainsi que d'autres types d'applications se basant sur des nanorobots ont également été explorés.

Ma propre thèse s'inscrit donc assez naturellement dans ces thématiques existantes en considérant l'aspect réseau et télécommunication qu'offre les nanoréseaux. On notera aussi qu'une autre thèse sur les nanoréseaux est encore en cours. Lina Aliouat étudie la mise en place d'un système de hiérarchisation (de cluster) au sein de nanoréseaux. Enfin, une toute nouvelle doctorante a également été accueillie au sein de l'équipe, signe que cette thématique est encore source d'intérêt pour ses membres.

PUBLICATIONS

Les travaux menés durant cette thèse ont été publiés dans quatre conférences :

[1] Dominique Dhoutaut, Thierry Arrabal, and Eugen Dedu. BitSimulator, an electromagnetic nanonetworks simulator. In *5th ACM/IEEE International Conference on Nanoscale Computing and Communication (NanoCom)*, pages 1–6, Reykjavik, Iceland, September

2018. ACM/IEEE.

[2] Thierry Arrabal, Dominique Dhoutaut, Eugen Dedu. Efficient Density Estimation Algorithm for Ultra Dense Wireless Networks. In *27th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9, Hangzhou, China, August 2018. IEEE. CORE A.

[3] Thierry Arrabal, Dominique Dhoutaut, and Eugen Dedu. Efficient multi-hop broadcasting in dense nanonetworks. In *17th IEEE International Symposium on Network Computing and Applications (NCA)*, pages 385–393, Cambridge, MA, USA, November 2018. IEEE. CORE A.

[4] Thierry Arrabal, Florian Büther, Dominique Dhoutaut, Eugen Dedu. Congestion Control by Deviation Routing in Electromagnetic Nanonetworks. In *6th ACM/IEEE International Conference on Nanoscale Computing and Communication (NanoCom)*, pages 1–6, Dublin, Ireland, September 2019. ACM/IEEE.

En cours de soumission

Yacine Benchaïb, Thierry Arrabal, Dominique Dhoutaut, Eugen Dedu. Influence of node nanosleeping on nanonetwork connectivity. In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 1–12, Lyon, France, February 2020. ACM. CORE A.

REMERCIEMENTS

Je tiens à remercier grandement mes deux encadrants Eugen Dedu et Dominique Dhoutaut pour m'avoir formé tout au long de ces trois années de thèse. Ils m'ont tous les deux beaucoup apporté, autant sur le plan purement professionnel en me proposant un cadre de travail idéal que sur le plan personnel grâce à leur grande sympathie. Ils m'ont tous les deux permis de mener des recherches passionnantes et de m'épanouir scientifiquement.

Je souhaite également remercier Isabelle Guérin-Lassous qui m'a fait faire mes premiers pas dans le monde de la recherche scientifique via une série de stages à l'ENS Lyon.

J'adresse un grand merci à Camille Thaize, ma compagne, mais aussi à l'ensemble de ma famille dont le soutien moral m'a apporté une aide précieuse.

J'aimerais également remercier quelques amis, notamment Florian Dufour, qui m'ont donné de précieux conseils en développement et en C++.

Un merci à tous les réflecteurs qui ont pris le temps de corriger ce manuscrit.

Enfin, un remerciement tout particulier à Hugo Rios qui m'a fortement aiguillé sur certains aspects très mathématiques de ces travaux, mais qui a également fournit une relecture précieuse de ce manuscrit.

SOMMAIRE

1 Introduction **5**

1.1 Nanoéquipements et nanoréseaux 5

1.2 Contributions aux nanoréseaux 7

2 Contexte **9**

2.1 Modulation pour les nanoréseaux électromagnétiques : TS-OOK 9

2.2 Protocole SLR 11

2.2.1 Adressage 12

2.2.2 Routage 12

3 BitSimulator **15**

3.1 Introduction 15

3.2 Comparaison avec les simulateurs existants 18

3.2.1 Network Simulator (NS) et Nano-Sim 18

3.2.2 TeraSim 19

3.2.3 Vouivre 19

3.2.4 Simulateurs de physique 19

3.3 Conception de BitSimulator 20

3.3.1 Principe de fonctionnement 20

3.3.2 Fonctionnalités réseau 21

3.4 Analyse des résultats et outils de visualisation 23

3.5 Validation des collisions 25

3.6 Évaluation des performances 27

3.6.1 Petit scénario 28

3.6.2 Grand scénario 29

3.6.2.1 Avec pré-calcul des voisins 29

3.6.2.2 Sans pré-calcul des voisins 29

3.7 Collaboration 30

4 DEDeN, estimateur de densité **33**

4.1 Travaux connexes . . . 34
4.1.1 Estimation du nombre de nœuds actifs . . . 35
4.1.2 DIP . . . 36
4.1.3 Estreme . . . 36
4.1.4 Estimation du nombre de puces en RFID . . . 37
4.2 Algorithme . . . 37
4.2.1 Phase d'initialisation . . . 37
4.2.2 Phase d'estimation des voisins . . . 38
4.3 Propriétés . . . 39
4.4 Confiance en l'estimation et calcul des seuils . . . 40
4.5 Performances de l'estimation de densité . . . 43
4.5.1 Qualité et qualification de l'erreur . . . 44
4.5.2 Le coût en termes de paquets envoyés . . . 44
4.6 Applications . . . 45
4.6.1 Un simple flooding probabiliste . . . 45
4.6.2 Autres applications possibles . . . 47
4.7 Conclusion . . . 47

5 Backoff flooding 55
5.1 Travaux connexes . . . 55
5.1.1 Pure flooding . . . 56
5.1.2 Flooding probabiliste adaptatif . . . 56
5.1.3 Protocoles de type geocasting . . . 56
5.1.4 OLSR . . . 56
5.1.5 Les réseaux hiérarchiques . . . 57
5.1.6 Schéma adaptatif counter-based . . . 57
5.2 Backoff flooding . . . 58
5.2.1 Taille de la fenêtre d'attente . . . 59
5.2.2 Effet du multiplicateur k sur le nombre de forwardeurs . . . 60
5.2.3 Délai induit . . . 63
5.2.4 Coût en mémoire et temps de stockage des paquets . . . 64
5.3 Simulation . . . 65
5.3.1 Scénario et paramètres . . . 66
5.3.2 Résultats . . . 67
5.4 Conclusion . . . 70

6 Routage capable de déviation 73
6.1 Introduction . . . 73
6.2 Congestion et travaux connexes . . . 74
6.3 La congestion dans les nanoréseaux . . . 76
6.4 Détection de la congestion . . . 77
6.5 Fusion de SLR et de backoff flooding . . . 78
6.6 La déviation de paquets . . . 79
6.7 Évaluation . . . 81
6.8 Collaboration . . . 83
6.9 Conclusion . . . 83

7 Endormissement des nœuds 85
7.1 Introduction . . . 85
7.2 Travaux connexes . . . 86
7.2.1 La portée de couverture dans les réseaux de capteurs . . . 87
7.3 Le nanosleeping . . . 90
7.3.1 Schéma . . . 90
7.3.2 Propriétés . . . 91
7.4 Simulation et résultats de connectivité . . . 94
7.4.1 Scenarios . . . 94
7.4.2 Résultats . . . 94
7.5 Résultats de simulation pour la diffusion de paquet . . . 95
7.5.1 Scénarios . . . 95
7.5.2 Résultats . . . 97
7.6 Conclusion . . . 98

8 Conclusion et travaux futurs 103

1

INTRODUCTION

Les nanoréseaux, les nanorobots et les robots autoreconfigurables sont une source d'inspiration depuis des décennies dans la littérature et le cinéma. On en retrouve par exemple dans le film "Terminator" avec les replicateurs, dans le film d'animation "Les Nouveaux Héros" et dans bien d'autres œuvres de science-fiction. Les différentes avancées dans le domaine de la miniaturisation rendent ces technologies chaque jour un peu plus accessibles.

Les applications envisagées pour ce nouveau type de réseaux ne sont pas moins impressionnantes que celles présentes dans l'imaginaire collectif, et s'en rapprochent même beaucoup. Dans les paragraphes suivants je vais faire une liste non-exhaustive des applications possibles pour les nanoréseaux.

Les nanoéquipements peuvent également embarquer une nanocaméra. Cette caméra de taille minuscule a des capacités extrêmement limitées et ne peut fournir que des images de taille très réduite. Cependant, un équipement macro équipé d'un réseau de nanocaméras très dense est capable de reconstituer une image dont chaque pixel est fourni par un nanoéquipement différent. Cette technologie permet alors d'obtenir des images d'une immense résolution grâce à la taille minuscule des capteurs.

Les réseaux de nanoéquipements sont aussi très intéressants pour la manipulation de produits chimiques. En effet, un réseau de nanocapteurs est capable, grâce à la taille minuscule de chaque capteur, de détecter des concentrations de produit chimique très faibles. De même, cet atout est utile pour déployer des nanoréseaux à l'intérieur du corps humain. Ce type de réseaux peut servir à détecter des maladies, des cellules endommagées etc. Ces réseaux sont également capables d'effectuer des livraisons de médicaments au plus prêt de l'endroit où ils seront utiles à l'organisme [39].

De nombreuses perspectives offertes par ce nouveau type de réseaux motive la recherche dans le domaine des nanoréseaux.

1.1/ NANOÉQUIPEMENTS ET NANORÉSEAUX

Ce nouveau type de réseaux est composé de nanoéquipements. Intuitivement, ces nanoéquipements sont des équipements de taille nanométrique, comme le montre la Figure 1.1. Ces nanoéquipements ont des capacités et des ressources très limitées. Ces limitations conduisent à la construction de réseaux de nanoéquipements potentiellement très denses afin de fournir des services et applications utiles comme mentionné au début

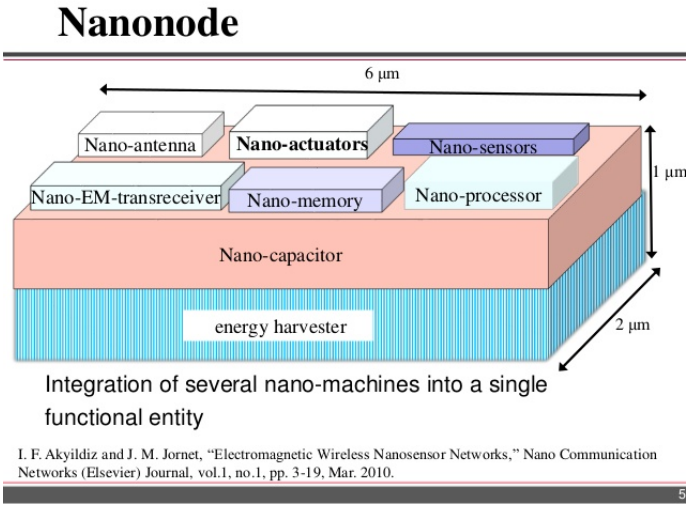


FIGURE 1.1 – Description d'un nanoéquipement ou nœud

de la section 1. [2] donne également un bref aperçu des possibilités offertes par les nanoréseaux. La taille extrêmement réduite de ces nanoéquipements leur permet cependant de récolter de l'énergie dans leur environnement, notamment grâce aux vibrations (à échelle macro) du milieu dans lequel ils sont placés [30]. Les vibrations du sang dans le corps, les ondes acoustiques d'une pièce peuvent par exemple représenter des sources d'énergie pour ces équipements.

Ces équipements ont des portées de communication très courtes, allant au plus jusqu'à quelques centimètres seulement. Cette contrainte oblige les nanocommunications à se faire de manière multi-sauts si l'on veut que le réseaux ait une taille raisonnable (quelques dizaines de centimètres par exemple pour un objet manipulable ou un réseau à l'intérieur du corps humain). Cette thèse s'intéresse avant tout aux aspects "réseau" de la couche protocolaire au sens TCP/IP. Cependant, il est important de noter que les nanoéquipements sont tellement simples que les approches et solutions considérées dans cette thèse sont principalement "cross-layer" et impactent donc des aspects habituellement considéré comme appartenant à la couche MAC ou encore à la couche transport.

On notera qu'il existe dans la littérature deux types de nanoréseaux. Les nanoréseaux moléculaires [9] et les nanoréseaux électromagnétiques. Dans les réseaux moléculaires, les communications se font via échange de molécules. Pour communiquer, les nœuds relâchent sur le canal une certaine quantité d'une molécule donnée. Les messages peuvent être différenciés grâce à différentes concentrations de la dite molécule dans le canal. Les communications moléculaires sont dites bio-compatibles. Des messages codés via des molécules sont monnaie courante dans le corps humain, il est donc facilement envisageable d'utiliser les communications moléculaires pour des applications médicales. Ce type de communication présente des caractéristiques très différentes des communications électromagnétiques habituellement utilisées dans les réseaux sans-fils. Les nanoréseaux électromagnétiques utilisent des ondes électromagnétiques pour communiquer, et ont donc une approche commune de la communication sans-fil. L'ensemble de ces travaux de thèse sont concentrés sur les nanoréseaux électromagnétiques. Cependant, les communications électromagnétiques dans les nanoréseaux sont très différentes des communications électromagnétiques que l'on rencontre dans les réseaux

macros. Les spécificités des techniques de modulation utilisées dans les nanoréseaux sont présentées en section 2.1.

1.2/ CONTRIBUTIONS AUX NANORÉSEAUX

Mes travaux s'inscrivent donc dans un domaine scientifique encore très jeune. Les systèmes nanométriques n'ont que très peu été étudiés d'un point de vue réseau, télécommunications multi-sauts, et couches transport. Nous avons essayé, durant cette thèse, de poser des bases, de mener des études préliminaires et de développer des premiers outils. Ces bases serviront (on l'espère) à des études plus poussées sur les nanoréseaux. La constitution d'un socle de connaissances et d'outils pertinents à l'étude des nanoréseaux a été un véritable fil rouge tout au long de ces travaux de thèse.

Les travaux connexes de chacune de mes contributions seront présentés en début de chaque chapitre afin que chaque chapitre puisse se lire de façon indépendante.

Dans la suite de ce manuscrit, je présenterai les contributions apportées par les travaux menés durant cette thèse :

- BitSimulator : Un simulateur réseau dédié aux nanoréseaux. Dans le chapitre 3 j'explique en quoi la simulation est nécessaire dans notre cas et en quoi les simulateurs existants ne sont pas utilisables pour nos travaux. J'y détaille également les principales fonctionnalités de notre simulateur et en propose une brève analyse.
- DEDeN : Un estimateur de densité (nombre de voisins des nœuds) prenant en compte les spécificités des nanoréseaux. Pour de nombreux algorithmes et protocoles fonctionnant dans les réseaux, des informations concernant les voisins de chaque nœud sont nécessaires. Or dans les nanoréseaux faire une liste, pour chaque nœud, de tous ces voisins est impossible. De plus les estimateurs de densité qui fonctionnent dans les réseaux macro ne fonctionnent pas dans les nanoréseaux. Le chapitre 4 explique en détail un nouvel algorithme d'estimation de densité compatible avec les contraintes des nanoréseaux et des réseaux très denses.
- Backoff flooding : Un schéma de transmission counter-based pour la dissémination de données. Les techniques de diffusion se basant sur un flooding pur sont souvent très inefficaces ; elles consomment énormément de ressources. Dans les nanoréseaux, il est impossible de choisir précisément un forwardeur optimal pour la diffusion d'un message. C'est pourquoi nous proposons, dans le chapitre 5, une méthode de diffusion optimisée permettant de limiter fortement le nombre de forwardeurs lors d'une diffusion.
- Déviation : Le chapitre 6 présente un protocole de routage. Ce nouveau protocole repose sur SLR, un protocole de routage adapté aux nanoréseaux et est lui aussi adapté à ce type de réseaux. Il est capable de détecter des zones de congestion en utilisant des caractéristiques propres aux nanocommunications et de dévier une route pour les éviter évitant ainsi la perte de paquets.
- Sleeping : un cycle d'endormissement/éveil des nœuds est proposé dans le chapitre 7. Il permet l'économie de ressources tout en prenant en compte les spécificités des nanoréseaux. En effet, les spécificités des communications dans les nanoréseaux permettent aux nœuds de s'endormir et de se réveiller plusieurs fois au cours de la réception d'un même paquet. Cependant, les techniques d'endormissement peuvent altérer la couverture ou la connectivité d'un réseau.

Cette étude vise à trouver un compromis entre sauvegarde de la connectivité et économie de ressources.

Dans l'ensemble de ce manuscrit, on parlera de message ou d'information pour désigner une donnée à transmettre sur le réseau. Si un nœud transmet une information et que tous les nœuds répètent cette information, on dit alors que le message (ou l'information) se diffuse dans le réseau. De même on parlera de paquet pour désigner la suite de bits codant ce message (auquel on ajoute tous les en-têtes appropriés). Ainsi, dans le cas où une diffusion multi-sauts est nécessaire les nœuds ont besoin de forwarder le paquet ; c'est le paquet qui est répété et non le message. On ne fait donc pas de distinction entre ce qu'on appelle habituellement trame, paquet ou datagramme. Nous avons adopté une approche cross-layer dans tous nos travaux, c'est pourquoi cette distinction n'a pas lieu d'être dans cette thèse.

2

CONTEXTE

2.1/ MODULATION POUR LES NANORÉSEAUX ÉLECTROMAGNÉTIQUES : TS-OOK

Les limitations en terme d'énergie et de puissance ainsi que la taille nanométrique des antennes ne permettent pas l'utilisation des modulations habituelles reposant sur des signaux porteurs (comme celles utilisées dans 802.11 par exemple). Par conséquent, un schéma très simple a été proposé, TS-OOK [34]. TS-OOK utilise des pulses d'énergie très courts (100 femtosecondes) pour coder les symboles. Le codage utilisé est on ne peut plus simple : à la réception, un pulse est interprété comme étant un "1" binaire et l'absence de pulse (un silence) comme un "0". Pour nos études, seul un petit nombre de paramètres est intéressant à considérer : la durée d'un pulse T_p, un seuil de puissance au-dessus duquel un "1" est reçu correctement et enfin le temps inter-symbole T_s (le temps séparant deux symboles). Le ratio d'étalement $\beta = T_s/T_p$ influence directement la vitesse de transmission. La transmission de quatre bits consécutifs est donnée en exemple dans la Figure 2.1. On notera que l'échelle et la forme des pulses ont été modifiées pour des raisons de lisibilité. Les pulses sont représentés comme de simples "pics" et le délai entre deux symboles est généralement bien plus grand que celui montré ici (β peut varier habituellement entre quelques centaines à quelques milliers). Cette modulation engendre des caractéristiques très différentes des modulations à porteuse.

Quand $T_s \gg T_p$, cette modulation permet un multiplexage temporel des paquets. Cette idée est illustrée dans la Figure 2.2, où les pulses de plusieurs paquets sont entrelacés. En effet, un nœud capable d'envoyer des pulses très courts ne pourra pas forcément envoyer un grand nombre de bits dans un laps de temps court (principalement à cause de la limitation d'énergie et de puissance de calcul). Par conséquent, un paquet donné ne peut pas être envoyé extrêmement rapidement. Mais, dans un environnement dense,

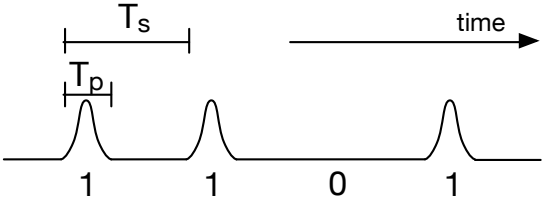


FIGURE 2.1 – Modulation des "0" et des "1" dans TS-OOK.

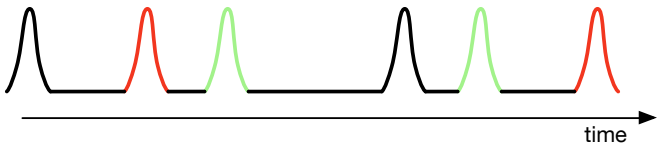


FIGURE 2.2 – Multiplexage temporel des paquets in TS-OOK.

le débit agrégé de plusieurs paquets multiplexés peut atteindre des valeurs très élevées. Cette capacité de multiplexage est très différente des réseaux traditionnels où les paquets sont envoyés de manière séquentielle sur le canal. Ici, avec de grandes valeurs de $beta$, des centaines de trames peuvent se trouver sur le canal en même temps. Ce type de multiplexage est différent du multiplexage que l'on retrouve dans d'autres types de communication. Intuitivement, plusieurs paquets peuvent coexister sur le canal en même temps sans que ces derniers ne subissent d'altérations. Les collisions sont un concept très différent dans les réseaux utilisant TS-OOK et dans les réseaux habituels, comme on le verra plus tard.

La très courte durée des pulses apporte une autre particularité : le délai de propagation radio (la vitesse de la lumière) n'est plus négligeable, même sur des distances aussi courtes que quelques millimètres, car ce délai peut être bien plus grand que la durée d'un pulse. Prenons un exemple : la vitesse de propagation est de 300 nm/fs (vitesse de la lumière), une portée de communications de 10 mm, une durée de pulse $T_p = 100$ fs et $\beta = 10$. Le temps que met un pulse pour voyager le long de la portée de communication est d'environ 33333 fs, ce qui correspond à environ à $33T_p$. Autrement dit, il y a le temps d'envoyer 33 pulses sur le canal avant qu'un nœud, situé à 10 mm de distance, ne reçoive le premier pulse.

La Figure 2.3 montre une topologie où S1 et S2 génèrent chacun un paquet à des moments différents, mais ces paquets arrivent à R1 au même moment. Particulièrement dans des réseaux denses avec de nombreux transmetteurs à portée de communication mais situés à des distances variées, cela signifie que les nœuds reçoivent les bits dans un ordre différent. Par exemple, le nœud R1 peut recevoir un bit du nœud S1 puis un bit du nœud S2, alors qu'un autre récepteur (disons R2) reçoit un bit du nœud S2 puis un bit du nœud S1. La position des nœuds les uns par rapport aux autres peut causer à certains bits d'être reçus en même temps sur certains nœuds et à des moment différents sur d'autres, et ce, dans un même voisinage.

Encoder les bits avec des pulses implique des spécificités concernant les collisions. On considérera, pour la suite, un modèle simplifié pour les pulses (où on ne tient pas compte de la vraie forme du signal électromagnétique). Dans les réseaux utilisant TS-OOK, les "collisions" surviennent lorsque deux bits arrivent au même moment sur un récepteur. On dit que les bits se chevauchent. Cependant, deux bits qui se chevauchent ne provoquent pas nécessairement d'erreur. Dans le cas où le bit en train d'être reçu est un bit "1", et qu'un autre "1" d'un paquet différent arrive sur le récepteur au même moment, cela ne cause pas d'erreur (de collisions) car la puissance reçue sur le canal dépasse le seuil de détection et donc le "1" est détecté correctement. De même les "0" ne provoquent pas de collisions car ce sont des silences. Pour conclure, deux bits reçus en même temps provoquent une erreur si un "0" devait être reçu mais qu'il est masqué par un "1" d'un autre paquet. Le phénomène est illustré dans la Figure 2.4, où les paquets transmis par les émetteurs S3 et S4 sont en collision au niveau du récepteur R2 : seuls les bits "0" de

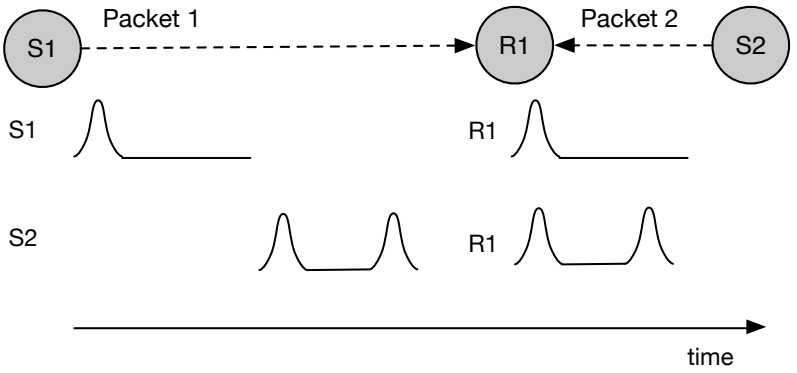


FIGURE 2.3 – Dans TS-OOK, le délai de propagation influence l'ordre d'arrivée des paquets et les collisions.

chaque trame sont altérés.

On notera que, dans un même voisinage à cause des échelles de temps et d'espace impliquées, les collisions ne touchent pas tous les nœuds de la même manière.

2.2/ PROTOCOLE SLR

La plupart des protocoles de routage connus dans les réseaux sans fil macro ne conviennent pas aux nanoréseaux. On peut dégager trois raisons principales à cela.

La première est la faible quantité de mémoire disponible sur chacun des nœuds. On n'attend pas des nanoéquipements d'être capables de maintenir une liste complète de leurs voisins. D'une part car la taille des équipements ne permet de stocker que très peu d'informations, mais aussi et surtout car nous nous intéressons à des réseaux particulièrement denses. Nous avons choisi de travailler avec des densités pouvant aller jusqu'à plusieurs milliers de voisins, voire théoriquement plus. Ces densités extrêmement élevées peuvent facilement être rencontrées dans le cadre de SDM (software defined materials) ou matière programmable. Ces deux paramètres assez extrêmes ne permettent pas un passage à l'échelle des protocoles habituels.

Le second point qui empêche l'utilisation de protocoles classiques est le fait que les nanoéquipements ne présentent que très peu de capacité de calcul. Il est donc souvent inenvisageable de faire des calculs complexes pour calculer la couverture optimale de chacun des voisins par exemple. L'algorithme d'adressage et de routage SLR [55] (Stateless Linear Routing) n'utilise que des nombres entiers et est sans état. Cela signifie qu'il est très léger et est donc adapté aux nanoréseaux. On notera que l'ensemble des solutions proposées dans cette thèse ne repose que sur de l'arithmétique entière ou à virgule fixe (qui est assimilable à de l'arithmétique entière).

Enfin, à ces échelles-là (des équipements de quelques μm), aucun système de positionnement macro, comme GPS ou Galileo, n'est encore disponible. Aucun protocole reposant sur un positionnement précis des nœuds n'est disponible. De plus les systèmes actuels ne permettent pas une localisation suffisamment précise, dans nos échelles de temps, pour pouvoir être utilisés dans les nanoréseaux [53]. Dans ce type de réseaux, quelques mm représentent déjà plusieurs portées de communication. Le manque de pro-

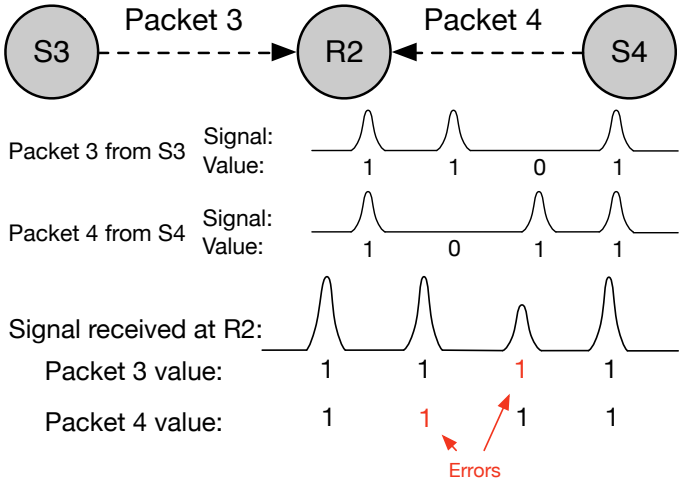


FIGURE 2.4 – Erreur de reception avec TS-OOK.

tocoles adaptés aux nanoréseaux a conduit à la conception d'un protocole de routage spécialement adapté à ces derniers. SLR est à la fois un protocole d'adressage et de routage de données. Ces deux aspects se déroulent dans deux phases bien distinctes et sont approfondis dans la suite.

2.2.1/ ADRESSAGE

La phase d'adressage est nécessaire pour pouvoir router des données dans le réseau. La phase d'adressage suppose l'existence de nœuds "particuliers" appelés ancres disposés à des endroits précis dans le réseau. On va s'intéresser à des réseaux de forme rectangulaire (ou carrée) ou à des parallélépipèdes rectangles (ou des cubes). Ces ancres doivent être au minimum au nombre de 2 pour des réseaux en 2D et au minimum au nombre de 3 pour des réseaux en 3D. Elles doivent se situer aux extrémités (ou angles) du réseau et sur une même face (donc pas placées aux angles opposés dans un réseau 2D).

Lors du déploiement du réseau, ces ancres diffusent chacune un beacon. Ce beacon est répété par tous les nœuds et se diffuse donc à travers tout le réseau. À chaque transmission du beacon, un champs dans celui-ci est incrémenté. Ce champ représente le nombre de sauts entre le nœud recevant le beacon et l'ancre qui l'a envoyé. Ce champ est similaire au TTL d'IPv4 mais en s'incrémentant. Avec suffisamment d'ancres et donc de beacons (2 en 2D et 3 en 3D), on crée un système de coordonnées. On notera que ce système de coordonnées et d'adressage n'assigne pas de coordonnée (ou d'adresse) unique à chaque nœud. Il est question ici de créer des zones contenant plusieurs nœuds et d'adresser ensuite chacune de ces zones.

2.2.2/ ROUTAGE

Dans la phase de routage, un nœud a besoin de transmettre des données à une zone de destination. Lorsque le nœud initial envoie son paquet, ce paquet est reçu par des nœuds des zones adjacentes. Ce paquet contient, en plus du reste, les adresses des

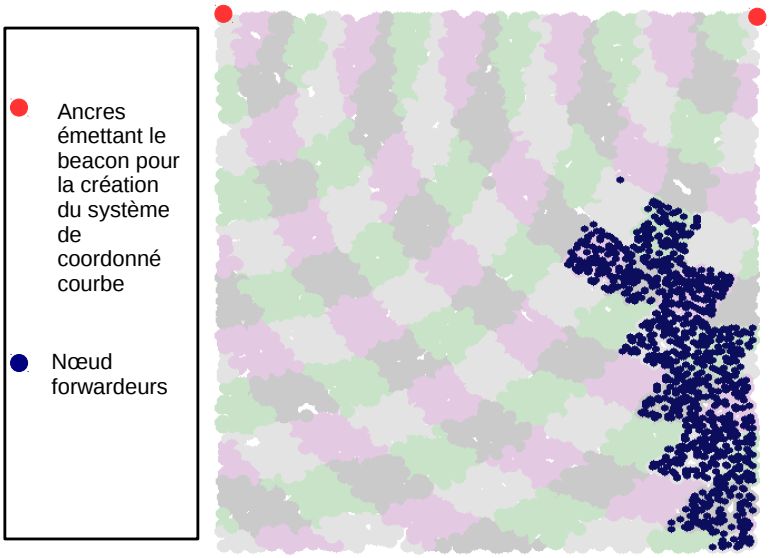


FIGURE 2.5 – Exemple de réseau SLR

zones source et destination. Grâce à celles-ci, les nœuds recevant le message peuvent comparer l'adresse source et destination avec leur propre adresse de zone SLR. Chaque nœud ayant reçu le paquet vérifie alors s'il se trouve dans une zone (car il connaît la zone à laquelle il appartient) sur le "segment" ayant la zone source et la zone destination pour extrémités. Ces calculs peuvent se faire entièrement en n'utilisant que de l'arithmétique entière, ce qui est particulièrement adapté aux faibles capacités de calcul disponibles sur les nœuds.

On notera que les "droites" et "segments" dans ce système de coordonnées ne sont pas des droites "standard". Le système de coordonnées créé de cette façon est "courbé". Le routage est illustré dans la Figure 2.5. On peut y avoir les coordonnées courbes, les nœuds de différente zone forwarder le message en bleu et les ancre mis en évidences en rouge.

3

BitSimulator

BitSimulator est un outils sur lequel repose la plupart des études présentées dans ce manuscrit. C'est un simulateur spécialement dédié aux nanoréseaux que j'ai co-développé tout au long de cette thèse. Ce chapitre explique en quoi les outils disponibles ne correspondent pas à nos besoins, présente les fonctionnalités de BitSimulator et fait une évaluation du simulateur.

BitSimulator a été un véritable fil rouge tout au long de cette thèse. Son développement a commencé relativement tôt et des ajouts y sont encore faits aujourd'hui. Pour chaque chapitre, il sera donné le numéro de la version avec laquelle les résultats de simulation ont été générés. Cependant, la plupart des résultats sont dépendant de nombres aléatoires cela signifie qu'il peut être difficile de les reproduire à l'identique même en utilisant les mêmes graines pour la génération des nombres aléatoires différent systèmes peuvent générer des séries de nombres pseudo-aléatoires différents pour des graines identiques. En revanche, les comportements généraux, les tendances et les ordres de grandeurs des résultats sont reproductibles.

Notre simulateur est disponible sous licence GNU. La dernière version disponible est la 0.9.4 parue en avril 2019. BitSimulator dispose d'une page web contenant son code source, de la documentation sur fonctionnement général et le code reproduisant certains articles. Un tutoriel pour l'utilisation de base du simulateur est également disponible sur cette page [1]. L'installation et la configuration est grandement facilité grâce à l'outils autotools.

3.1/ Introduction

Les nanoréseaux à communication électromagnétique sont un sujet de recherche encore très jeune, où des milliers voire des millions d'appareils très simples et très petits communiquent et interagissent entre eux. De nombreux défis (verrous scientifiques) sont encore à résoudre, notamment à cause des restrictions imposées par la taille nanométrique des équipements. Pour faire face aux spécificités physiques et environnementales des nanoréseaux (CPU, mémoire et énergie extrêmement limités), une totale révision de la pile protocolaire est nécessaire. Toutes les couches sont concernées, aussi bien l'accès au canal, le codage, le routage et la couche transport que les applications elles-mêmes.

De nombreux progrès ont déjà été faits concernant la modélisation du canal

1. http://eugen.dedu.free.fr/bitsimulator

électromagnétique ainsi que l'accès au canal. TS-OOK (Time Spread On-Off keying) a été proposé dans l'article [34]. Ce schéma de communication utilise des pulses électromagnétiques très courts pour communiquer. Ces pulses sont suffisamment courts pour être générés par des équipements minuscules et être détectés par ces mêmes équipements. Cependant, il reste encore beaucoup à faire en ce qui concerne les autres couches protocolaires.

Aucune implémentation (physique) de nanomachines communicantes sans-fil n'est aujourd'hui disponible. Les travaux portant sur les protocoles réseau, le codage ou encore sur les applications doivent être menés soit de façon analytique (mathématique) soit via des simulations. Les méthodes purement analytiques sont très complexes à mettre en œuvre, car le domaine des communications dans les nanoréseaux est encore immature. De plus, des comportements émergeants et donc très difficiles à prendre en compte dans des formules prédictives sont attendus dans ce domaine, car le nombre de nœuds communiquant impliqués est potentiellement très élevé. De plus, chacun des protocoles et des comportements impliqués dans un réseau peut être complexe en soi. Modéliser mathématiquement des agents complexes ayant des interactions complexes peut facilement mener à des erreurs. Une autre solution serait d'utiliser des modèles simplifiés ; cependant, ce type de modèles masque bon nombre de phénomènes importants, en particulier certains comportements émergeant difficiles à prévoir par pure expérience de pensée ou modélisation mathématique. C'est donc cet ensemble de raisons qui nous a poussés à nous tourner vers l'utilisation de simulations.

Les chercheurs utilisent souvent des simulateurs développés en interne, sous forme de projet secondaire. Cependant, ces outils ont souvent des défauts étant donné le peu de temps investi dans ces derniers. Il est très difficile, lors de la conception de ces outils, de trouver le bon équilibre entre complexité, précision et ressources nécessaires (CPU et mémoire par exemple). La valeur scientifique des résultats de simulations est affectée par ces considérations.

Deux approches sont alors possibles. D'un coté, l'utilisation de simulateurs existants permet une mise en route des études plus rapide. Le simulateur étant pré-existant, la charge de travail que représente le développement d'un nouveau simulateur n'est pas à fournir. De plus, cela permet une centralisation des travaux, ce qui influe sur la reproductibilité des résultats : utiliser un simulateur connu, et donc plus accessible, facilite la réutilisation des travaux. D'un autre coté, un simulateur conçu en interne permet de faire face à des problématiques très spécifiques et de manière extrêmement contrôlée étant donné que tous les aspects du code (les possibilités et limites) du simulateur sont connus. Il sera montré dans la section 3.2 qu'aucun des simulateurs existants ne répond à nos attentes. Pour nos besoins, le simulateur doit proposer les fonctionnalités suivantes.

Des nœuds et des applications individuels. Comme mentionné dans l'introduction, nous avons besoin d'un outil pour aider la conception, simuler et valider des protocoles et applications réseaux. Une étude analytique du réseau dans sa globalité est souvent très complexe (quand la pile réseau ou le scénario est trop complexe) ou pas suffisamment poussée (pour prendre en compte des subtilités et des cas particuliers). Une implémentation de toute la pile protocolaire est souvent nécessaire, avec une instanciation individuelle de chaque élément. Chaque nœud et chaque fragment de code (application sur les nœuds par exemple) sont traités séparément. Cette implémentation individuelle de chaque entité (nœuds et protocoles) permet de mettre en avant et de faire ressortir leurs interactions complexes de manière très fidèle.

Les transmissions de chaque bit et le calcul des erreurs. Comme présenté dans la section précédente, le contenu des paquets (le payload) et le codage (nombre de ”1”et de ”0” dans les paquets) ont un effet majeur sur le taux de collision. Ces erreurs (collisions) ont besoin d'être simulées de façons très précises, surtout dans les études traitant du codage. Par exemple, des travaux précédents de l'équipe sur le codage [62] [63] [64] ont été validés uniquement mathématiquement et par analyse numérique faute de simulateur. Comme mentionné auparavant, les collisions dans TS-OOK sont très différentes des collisions classique.

Délai et propagation du signal. De petits changements dans le positionnement de nœud ou le timing dans l'envoi des paquets affectent significativement les bits reçus correctement ou en collision. Des protocoles d'accès au canal comme [60] utilisent des préambules physiques spécifiques et calculent un temps inter-bit optimal. Ce type de protocole réduit significativement le risque de collision, mais ne peut pas les empêcher totalement, surtout dans des scénarios très denses. La simulation correcte de chaque bit contenue dans les paquets (cf. point précédent), ainsi que le moment précis de chaque événement (y compris le délai de propagation) ne peuvent pas être négligés à ces échelles de temps et d'espace.

De “nombreux” paquets entrelacés sur le canal. Ce point est crucial et définit la nature même des nanocommunications sans-fil. De nombreux paquets (possiblement des centaines voire plus) peuvent être entrelacés sur le canal. Cela implique que les nœuds puissent recevoir et décoder plusieurs trames en parallèle. Cela est techniquement possible, mais le nombre de paquets pouvant être décodés en parallèle doit être limité pour prendre en compte les limitations des nœuds, que ce soit d'un point de vue matériel et/ou logiciel.

Possibilité de gérer un grand nombre de nœuds. Les applications envisagées pour les nanoréseaux comprennent notamment la matière programmable et les réseaux de capteurs. Ces applications peuvent impliquer de nombreux nœuds, possiblement des millions. Le simulateur doit pouvoir être capable de gérer ce type de réseaux afin de tester et de valider des algorithmes et protocoles dans ce genre de scénario. Il doit aussi être capable de gérer des réseaux où la densité locale (c'est-à-dire le nombre de voisins de chaque nœud) est potentiellement très élevée.

Vitesse de simulation. Pour augmenter la fiabilité des simulations et diminuer l'influence des événements statistiquement rares ou des cas particuliers, et ce, même s'il est important de ne pas les négliger, la pratique communément admise consiste à lancer de nombreuses fois un même scénario en changeant les graines des générateurs de nombres aléatoires. Il est alors d'autant plus important que le simulateur ait un code optimisé car, pour chaque scénario à tester, il est courant de lancer au moins une dizaine de simulations. Une parallélisation du simulateur est difficile, compte tenu des interdépendances (notamment spatiale) entre les événements simulés. Dans le cadre d'une approche statistique des résultats (exécuter plusieurs fois le même scénario en changeant les graines), il est par contre possible de lancer en parallèle plusieurs instances du simulateur, pour un speed-up optimal.

Ce chapitre présente BitSimulator, un simulateur dédié aux nanoréseaux électromagnétiques qui a été développé tout au long de ces trois années de thèse. Cet outil est conçu pour aider les chercheurs à manipuler des nanoréseaux et donc faciliter l'appréhension et la compréhension des comportements inhérents à ce type de réseau. Ce simulateur permet également le développement et l'étude de nouveaux

protocoles. On notera que notre simulateur considère des antennes (et donc des communications) omnidirectionnelle. Cependant, des travaux menés par Julien Bourgeois, Hakim Mabed et Lina Aliouat sont actuellement en cours pour considérer des antennes unidirectionnelles dans le simulateur.

Le simulateur est capable de simuler aussi bien des environnement en 2 dimensions (2D) que des environnement en 3 dimensions (3D).

Les contributions présentées de se chapitre reposent principalement sur l'article [21].

3.2/ COMPARAISON AVEC LES SIMULATEURS EXISTANTS

Plusieurs simulateurs réseaux ont déjà été développés. En développer un nouveau est un effort conséquent, cette décision ne doit pas être prise avant d'avoir considéré l'usage d'outils existants. Nous ne parlerons ici que des simulateurs capable de simuler des nanoréseaux.

3.2.1/ NETWORK SIMULATOR (NS) ET NANO-SIM

Network simulator (NS) est utilisé depuis environ trente ans. NS-2 et NS-3 sont largement utilisés dans la communauté réseau. NS-3 [2] est un outil très versatile et peut simuler de nombreux types de réseaux (satellites, Wi-Fi, Zigbee, réseaux filaires etc.). NS-3 implémente toute la pile protocolaire avec plusieurs couches physiques, liaisons, routages, transports et applications.

NS-3 comporte une couche dédiée aux nanoréseaux, Nano-Sim [40], qui implémente le modèle TS-OOK mentionné plus tôt. NS-3, paraissait initialement être une très bonne option pour nos travaux. Cependant, plus nos recherches avançaient sur l'accès au canal et le contrôle de congestion sur des réseaux mono-saut mais surtout multi-sauts, plus le besoin d'avoir un outil spécialisé se faisait ressentir. Un article préliminaire dans notre équipe avait par ailleurs trouvé certaine limite à l'usage de Nano-Sim [20]. De plus, un de nos objectifs était d'étudier et de comprendre comment les erreurs et les redondances affectent les performances. Nous avions donc besoin de travailler sur les bits eux-mêmes. Nano-sim permet l'entrelacement des paquets sur le canal (plusieurs paquets sur le canal en même temps). Cependant, le modèle implémenté est trop simple :

- Le délai de propagation n'est pas pris en compte : quand un paquet est envoyé, tous les nœuds à portée le reçoivent en même temps.
- La détection de collision considère que tous les paquets utilisent la même valeur pour le paramètre β. C'est un calcul "tout-ou-rien", c'est-à-dire que si un bit d'un paquet entre en collision avec un bit d'un autre paquet, tous les bits des deux paquets sont considérés comment étant en collision.
- Les bits arrivant au même moment sur un récepteur provoquent toujours une collision quelle que soit leur valeur ("0" ou "1").
- Les paquets sont réordonnés sans raison apparente, et la gigue ne varie jamais, ce qui n'est pas réaliste [20].

De plus, la forte versatilité de NS-3 implique un surcoût en terme de CPU et de mémoire qui est non négligeable, réduisant drastiquement le nombre de nœuds pouvant être si-

2. https://www.nsnam.org/

mulés.

3.2.2/ TERASIM

TeraSim [61] est un simulateur de nanoréseaux qui se base également sur NS3 et qui est paru en 2018. Il est donc paru après que nous ayons utilisé notre simulateur. Les auteurs ont mis l'accent sur la modélisation du canal THz et sont ainsi capables de prendre en compte des phénomènes que BitSimulator ne peut pas encore gérer. Par exemple, dans TeraSim, il est possible que le bruit accumulé de plusieurs bits éloignés puisse générer des interférences et donc des collisions, même si aucun des bits éloignés (pris individuellement) n'aurait eu aucun impact sur les communications considérées.

Cependant, BitSimulator présente plusieurs avantages sur TeraSim. Premièrement, TeraSim considère que toutes les communications se font avec le même paramètre β, ce qui ne permet pas des études poussées concernant ce paramètre important. De plus, TeraSim ne prend pas en compte le payload des paquets, dès que deux paquets se chevauchent, ne serait-ce que d'un bit, il y a collision et perte complète des deux paquets. Cette façon de faire accélère les calculs, mais ne permet pas d'étude sur le codage ou sur les qualités vidéo (PSNR, SSIM) par exemple. Enfin, reposant sur la lourde infrastructure imposée par NS3, TeraSim n'est pas capable de gérer les réseaux comprenant un grand nombre de nœuds ; après discussion avec les auteurs, un millier de nœuds semble être une limite optimiste.

On souhaite faire de BitSimulator un outil souple proposant divers types d'options pouvant ainsi modifier le ratio vitesse d'exécution / complexité et précision.

3.2.3/ VOUIVRE

Vouivre [13] est une bibliothèque de simulation de nanoréseaux développée dans notre équipe et paru en 2015. Elle peut également être utilisée en "standalone", c'est-à-dire comme un simulateur standard. À cause de l'absorption moléculaire, le canal produit du bruit quand il est excité, c'est-à-dire quand des bits "1" sont envoyés [31]. Vouivre est capable de simuler des réseaux extrêmement denses comprenant plusieurs millions de nœuds. Il utilise le modèle statistique réaliste [33] pour le calcul des erreurs, il supporte de très grandes densités (des dizaines de milliers de voisins) tout en restant assez rapide. Cependant, l'usage d'un modèle statistique empêche l'étude précise (au niveau des bits) des effets liés à l'exact contenu des paquets. Il empêche également l'étude de positionnement très précis des nœuds (au nm prêt dans notre cas) et de cas particuliers.

A noter, que mon co-encadrant, Dominique Dhoutaut a été co-développeur de Vouivre.

3.2.4/ SIMULATEURS DE PHYSIQUE

Les approches très bas niveau utilisent des modèles physiques opérant sur des simulateurs génériques. Dans [47], COMSOL Multiphysics [3] est utilisé pour simuler les comportements d'une nanoantenne conçue à partir de graphène. [54] utilise AnyLogic [4] qui

3. https://www.comsol.com
4. https://www.anylogic.com

implémente une technologie basée sur le tracé de rayons (raytracing).

Ce type d'outils produit des résultats particulièrement précis, mais ont deux inconvénients majeurs dans le cadre de nos recherches. Ces outils ont besoin d'une configuration complexe où l'environnement doit être décrit très précisément. De plus, en fonction des compromis que l'on souhaite faire (typiquement, si le contenu des paquets est effectivement simulé ou non), les calculs peuvent être très coûteux et ne peuvent être utilisés que pour des réseaux ne comprenant que peu de nœuds.

3.3/ CONCEPTION DE BITSIMULATOR

Les limitations imposées par les outils existants nous ont amenés à développer un simulateur entièrement dédié aux nanoréseaux. BitSimulator est le compromis entre le niveau de détails, la précision et la vitesse d'exécution dont j'ai eu besoin durant mon doctorat. Le simulateur ne gère qu'un seul type de réseau : les nanoréseaux sans-fil basés sur TS-OOK ; cela nous a permis de nombreuses simplifications lors de la conception et de se prémunir de surcoûts inutiles (comme ceux présents dans NS-3). Ces simplifications permettent un développement, un traitement des résultats et une visualisation relativement simples.

Cette partie du rapport donne de plus amples explications à propos des choix de design faits dans le développement de BitSimulator.

3.3.1/ PRINCIPE DE FONCTIONNEMENT

Comme bien d'autres simulateurs gérant des applications et des protocoles de routage, BitSimulator est basé sur des événements. En son cœur réside un modèle à événement discret où les actions sont ordonnancées dans une liste triée selon la date à laquelle l'événement doit survenir. Les événements dans cette liste sont traités dans l'ordre, et peuvent à leur tour déclencher la création (et donc l'insertion) de nouveaux événements dans cette même liste (mais en aucun cas modifier des éventements déjà présents dans la liste). La simulation s'arrête quand il ne reste plus aucun événement dans cette liste ou que le temps de simulation maximal a été dépassé (ce temps est une durée arbitraire modifiable dans le code).

Le simulateur est amené à gérer des périodes très courtes (comme la durée d'un pulse unique ou encore le délai de propagation du signal sur quelques millimètres) ainsi que des périodes potentiellement bien plus longues (comme le temps entre la génération de deux paquets par l'application). Pour permettre cela, des entiers sur 64 bits sont utilisés pour gérer le temps en femtosecondes. Cette très haute résolution permet à la fois de gérer des pulses de 100 fs et des temps de simulation relativement longs ; de plusieurs secondes par exemple. Avec des entiers codés sur 64 bits, il est possible de coder l'équivalent d'un peu plus de 2 heures et demi à la femtoseconde près, sachant que quelques secondes représentent déjà une durée relativement longue au regard des durées impliquées dans les nanocommunications. Les nœuds sont disposés dans un monde en 2D ou 3D. Pour être cohérent dans la très grande résolution temporelle, la résolution spatiale est tout autant élevée. Le positionnement et les distances sont exprimés en nanomètres.

Dans le simulateur,les nœuds sont théoriquement capables de se déplacer à l'intérieur du monde simulé. Un mode de calcul dynamique des voisins est disponible et peut donc gérer une possible mobilité. Cependant, aucune fonction de déplacement ni aucun modèle de mobilité n'a été implémenté pour le moment. Dans le cadre de cette thèse, on a considéré uniquement des réseaux statiques, l'étude de réseaux dynamiques sera faite dans des travaux ultérieurs.

La reproductibilité des résultats est assurée par l'utilisation de générateurs de nombre aléatoires reproductibles. Plusieurs graines sont utilisées ce qui permet de tester des phénomènes très précis. Par exemple, il est possible d'effectuer deux simulations où les nœuds sont disposés de la même manière (en fixant la graine utilisée pour le positionnement) mais d'avoir différents tirages aléatoires pour divers paramètres des protocoles simulés (les durées de backoff, moment d'éveil ou d'endormissement, probabilité de forwarder, etc).

Chaque scénario de simulation est décrit dans un fichier XML (dont un exemple est donné dans la page web [5]), et de nombreux paramètres peuvent également être définis ou remplacés en ligne de commande (au moment de l'exécution du simulateur). Cela facilite l'exécution de simulations dans des scripts shell par exemple.

Le passage à l'échelle concernant le nombre de nœuds simulés est un des buts principaux du simulateur. Cela est possible grâce à un équilibre entre simplicité et souplesse du code. Un nombre relativement petit de classes C++ sont fournies pour encapsuler toutes les fonctionnalités principales, comme expliqué dans la suite.

3.3.2/ FONCTIONNALITÉS RÉSEAU

Afin de garder le simulateur simple et rapide tout en permettant aux chercheurs de contrôler les applications et les protocoles réseaux, une architecture basée sur trois couches réseaux est proposée. Cependant, les interactions entre ces couches sont très faciles. Chaque "couche" à facilement accès aux informations inhérentes aux autres couches, pour respecter la philosophie cross-layer avec laquelle a été pensé notre approche des nanoréseaux.

Partie physique et de contrôle d'accès au canal. Cette partie gère le délai de propagation du signal, gère les réceptions et les collisions. Les équipements simulés possèdent un transmetteur radio unique dont la portée peut être configurée. Cette couche implémente par défaut le modèle TS-OOK avec des pulses des 100 fs et un β configurable pour chaque paquet. Cependant, il est relativement facile d'altérer son comportement pour implémenter n'importe quel autre modèle basé sur des communications par pulse.

De multiples paquets peuvent être entrelacés sur le canal au même moment, c'est pourquoi les nœuds doivent traquer celui (ceux) par le(s)quelle(s) ils sont intéressés. Le matériel et/ou logiciel utilisé sur ces nœuds risque d'imposer un nombre maximal de paquets qui peuvent être suivis simultanément sur chacun des nœuds. Ce paramètre peut être configuré via le paramètre `maxConcurrentReceptions`.

Lors d'une réception correcte, les paquets sont pris en charge par un agent de routage. Il est possible de configurer dans le code un seuil d'erreur. Ce seuil détermine le nombre de bits qui peuvent être altérés sans rendre le paquet impossible à décoder. Grâce à

5. http://eugen.dedu.free.fr/bitsimulator

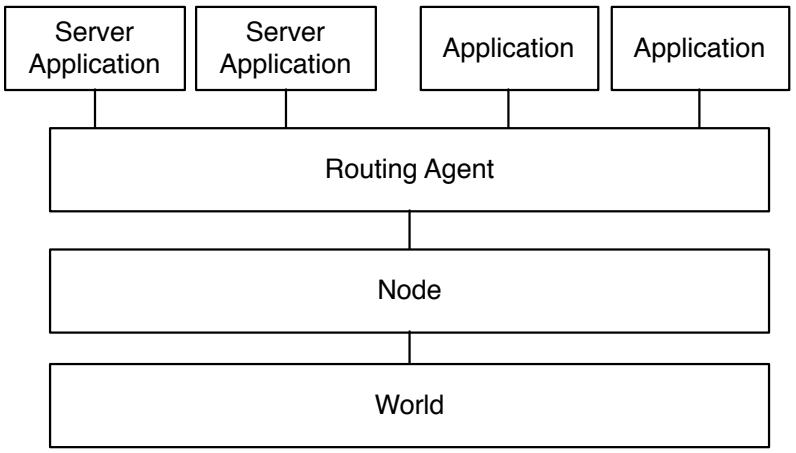


FIGURE 3.1 – Interactions entre les classes C++ principales.

ce seuil, il est facile de mimer le comportement d'un code correcteur d'erreurs et de mener des études préliminaires sur ce sujet. De plus, il est toujours possible de choisir de faire remonter les paquets "trop endommagés" aux couches supérieures, pour étudier les effets de l'altération des paquets sur l'interprétation des données, par exemple.

Cette couche est principalement implémentée par la classe C++ `Node` (voir la Figure 3.1). Certaines interactions et les structures de données concernant le positionnement sont implémentées dans la classe `World`.

Routage. À cause de la faible énergie disponible, la portée de communication des nanoéquipements sera probablement très courte. Les réseaux multi-sauts et ad hoc seront donc monnaie courante dans les nanoréseaux. C'est avec cela en tête que nous avons proposé trois options principales : pas de routage, une simple inondation (que l'on appellera pure flooding tout au long de ce rapport) et SLR [55]. Une version backoff flooding est également disponible pour l'inondation et SLR, ainsi qu'une version probabiliste de l'inondation, ces points-là sont développés dans des chapitres dédiés. D'autres agents peuvent être facilement ajoutés, un tutoriel à cet effet est disponible sur la page web de BitSimulator.

Application. BitSimulator permet l'instanciation de deux types d'applications. Application, qui ne peuvent qu'envoyer des paquets, et ServerApplications, qui peuvent envoyer et recevoir des paquets. Un nœud peut héberger autant d'applications que nécessaire. Les applications Server sont liées à un port logiciel qui permet le démultiplexage et la distribution des paquets à l'application serveur concernée. Pour créer une application (simple ou serveur), il faut hériter de la classe appropriée, et l'attacher aux nœuds concernés.

Les données évoluant dans le réseau (ainsi que dans les nœuds) sont modélisées via la classe `Packet`. Les paquets contiennent une charge binaire (payload), qui peut être définie par l'application, de manière statique ou aléatoire. Ils contiennent également diverses méta-données qui aident à visualiser et à comprendre les protocoles impliqués, notamment la source, la destination et des identifiants de flux. Il est facile d'ajouter tous les champs nécessaires à d'éventuels nouveaux usages.

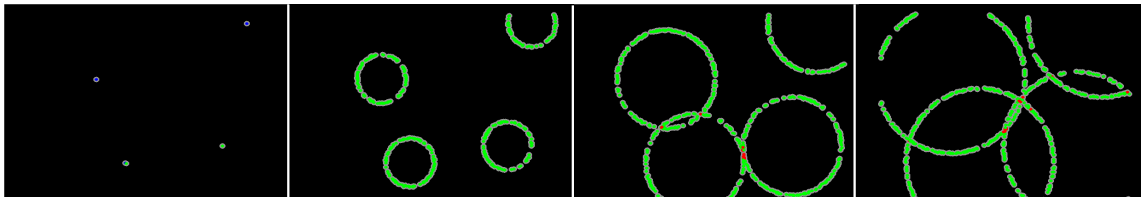


FIGURE 3.2 – 30 000 voisins recevant quelques paquets à différents moments (4 étapes très courtes).

3.4/ ANALYSE DES RÉSULTATS ET OUTILS DE VISUALISATION

BitSimulator fournis certaine informations (comme la vitesse de simulations, le temps simulé, le nombre d'événements total...) sur la sortie standard. Les résultats de simulation sont eux enregistré dans divers fichiers. La positions de chaque nœuds est par exemple stocké dans un fichier. Un autre fichier contient l'ensemble des événements qui au lieux au cours de la simulation. L'analyse des fichiers de traces produits par BitSimulator doit rester le plus simple possible. Certaines traces sont suffisamment simples pour être directement importées dans un tableur ou dans l'outil gnuplot. Mais il est préférable d'utiliser un outil dédié à la visualisation pour des traces plus volumineuses. Nous avons donc développé en parallèle VisualTracer. Cet outil importe les fichiers de traces produits par le simulateur et en produit une visualisation. Cela permet une appréhension intuitive du comportement global d'un protocole et de la propagation des messages dans le réseau, et ainsi faciliter le repérage d'éventuels problèmes dans sa conception. Bien que BitSimulator puisse fonctionner en 3D, VisualTracer lui en est incapable. Cette fonctionnalité est aujourd'hui en cours de développement.

VisualTracer gère actuellement deux modes d'affichage : le mode global et le mode individuel. Le mode global montre l'ensemble des nœuds et permet de visualiser divers événements comme les transmissions, les réceptions ou encore les collisions. De plus, il est possible d'interagir avec l'affichage afin de masquer/afficher certains types d'événements (et donc de n'afficher que les collisions par exemple). L'utilisateur peut choisir un intervalle de temps à afficher puis naviguer de manière interactive entre les différentes étapes de la simulation.

L'intégralité de l'interface de VisualTracer en mode global est montrée dans les Figures 3.3 et 3.4. Le même scénario qu'à la Figure 3.2 est présenté, mais l'intervalle de temps affiché est bien plus grand dans la Figure 3.3. Dans cette Figure, dans la zone située à gauche, tous les nœuds du réseau sont affichés. La couleur des noeuds indique leur état :

- Gris : Couleur par défaut
- Bleu : Couleur des nœuds venant de transmettre un paquet
- Vert : Couleur des nœuds venant de correctement recevoir un paquet
- Rouge : Couleur des nœuds venant de recevoir un paquet en collisions
- Jaune : Couleur des nœuds recevant recevoir un paquet qu'ils ne peuvent pas traiter

On notera que lors d'un même intervalle de temps (surtout si celui ci est grand) un même nœud peut vouloir afficher plusieurs événement de type différent (par exemple un nœuds recevant deux paquets, un reçu correctement et un reçu en collision). Pour gérer ce genre de situation, il est possible d'afficher/masquer certains type d'événements. Les

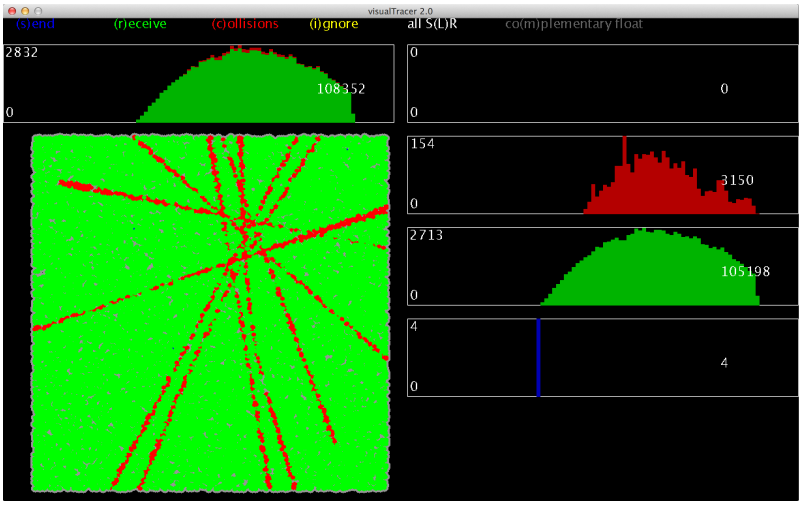


FIGURE 3.3 – Réceptions différées causées par le délai de propagation montrée dans VisualTracer : Grand intervalle de temps.

nœuds se recolorent alors en fonctions de type d'événement affichés/masqués. On notera également il existe des couleurs dédiées à l'affichage des zones SLR. Les nœuds recevant des paquets contenant des erreurs peuvent facilement être identifiés grâce à ce code couleur (ligne en pointillé rouge). Sur la partie droite, les graphiques montrent l'activité des nœuds durant la période sélectionnée, le code couleurs est le même que celui mentionné plus haut. Ces histogrammes donnent une autre vision de ce qui se passe durant l'intervalle de temps en train d'être montré dans la partie principale de l'interface en découpant cet intervalle en sous-intervalles 100 fois plus petits. Il est donc possible d'observer la répartition des différents événements au sein d'un intervalle donné. Par exemple dans la Figure 3.3 on voit la répartition temporelle des événements de l'ensemble de la simulation alors que dans la Figure 3.4 on devine aisément que des événements ont eu lieu avant et auront lieu après. D'autres histogrammes sont affichés au-dessus de la zone principale. Ils sont une vision condensée des 4 séries d'histogrammes mentionnés précédemment, cela permet également de rapidement comparer les nombres d'événements de chaque type entre eux.

La Figure 3.4 montre une fois de plus le même scénario, mais avec un intervalle de temps bien plus petit, facilitant ainsi la compréhension de l'apparition des collisions et leurs emplacements (quels nœuds, précisément, reçoivent un paquet erroné). On y voit quatre paquets en train de se propager, et les autres nœuds les reçoivent graduellement en fonction de leur distance par rapport aux sources.

La Figure 3.2 présente un partie de la fenêtre de VisualTracer pour un scénario avec 30 000 nœuds (tous dans le même rayon de communication) dont 4 émetteurs. Ces 4 émetteurs commencent leur transmission plus ou moins au même moment. La Figure 3.2 montre la partie de l'interface où les nœuds sont affichés, et ce, à quatre moments différents. On peut voir la progression du message (et donc des paquets) d'image en image, mais également que certains nœuds reçoivent des paquets en collision (intersection de cercles).

Le second mode de visualisation est le mode individuel, qui présente le point de vue d'un nœud en particulier sous la forme de chronogramme. Ce mode affiche les bits et les paquets sur une “timeline”, ce qui est particulièrement utile pour détecter des répétitions

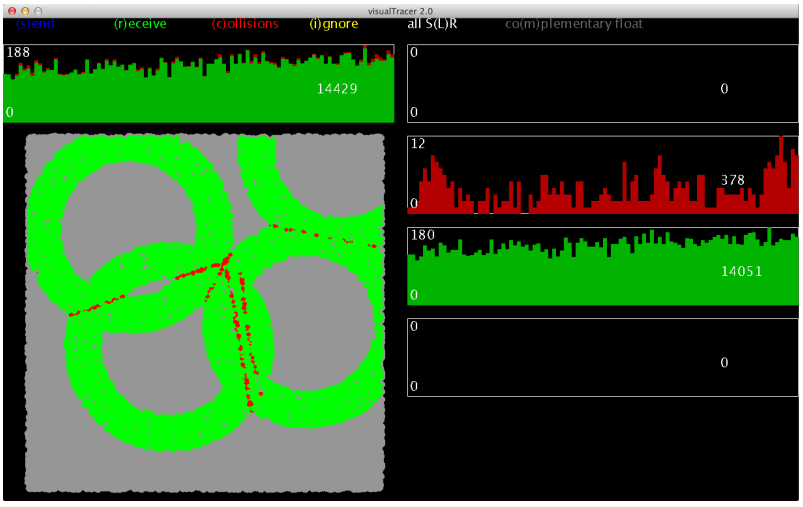


FIGURE 3.4 – Réceptions différées causées par le délai de propagation montrée dans VisualTracer : Intervalle de temps moyen.

ou des explosions de collisions. La Figure 3.5 montre les collisions qui ont lieu dans le scénario à quatre flux. Dans cette situation, seulement deux des quatre flux sont affectés par des collisions (les deux en bas). Les Figure 3.6,3.7 et 3.8 montrent différents niveaux de zoom (interactif) dans un autre scénario avec quatre émetteurs actifs transmettant des paquets de 10 bits et quelques collisions peuvent être observées.

3.5/ VALIDATION DES COLLISIONS

Les simulations présentées dans cette section, sont des simulations de validation. Elles ont été conduites juste avant la première véritable release (à savoir la 0.9.2 parue en Juin 2018) afin de s'assurer du bon fonctionnement du simulateur.

Afin d'étudier la capacité du simulateur à gérer correctement les collisions, nous avons conçu un scénario comprenant 5000 nœuds tous à portée les uns des autres. Parmi eux, quelques dizaines de nœuds transmettent à un rythme constant des flux composés de paquets longs de 8000 bits. Le temps entre la génération de deux paquets est choisi de telle sorte qu'un nouveau paquet soit envoyé juste après que le paquet précédent ait fini d'être transmis si on considère que le β choisi est de 1000. Si le β utilisé est plus petit, cela signifie que le paquet prendra moins de temps à être transmis, et le nœud devra alors attendre avant de générer un nouveau paquet. De plus, tous les nœuds sont configurés pour utiliser le même β (50, 500 ou 1000 dans les trois cas décrit plus bas). La Figure 3.9 montre comment ces flux occupent le canal d'un point de vue temporel. On peux voir sur la première ligne les paquets utilisant un $\beta = 1000$ occupant le canal longtemps, si bien qu'il sont envoyés sans espacement entre deux paquets. Sur la deuxième ligne, les paquets sont envoyé au même rythme (les début de paquet sont synchronisés) mais avec un $\beta = 500$. On peut voir que dans ce cas là, chaque paquet dure deux fois moins longtemps et qu'il y a de l'espace entre deux paquets consécutifs. Enfin, sur la dernières ligne, les paquets sont envoyé avec un $\beta = 50$.

Pour empêcher les collisions dues au fait que les émetteurs commencent à transmettre en même temps, un backoff aléatoire est utilisé, avec une valeur tirée dans l'intervalle $[0; T_s]$

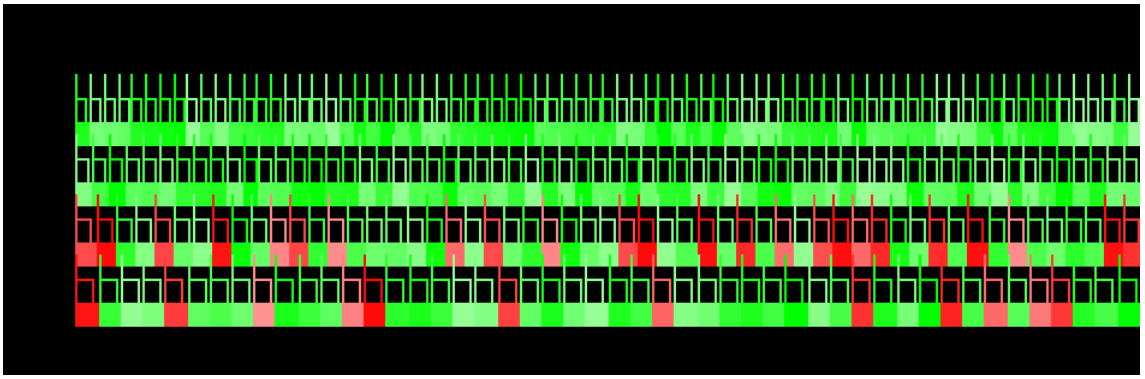


FIGURE 3.5 – Mode chronogramme : multiples collisions entre 2 flux dans un scénario de 4 flux.

(i.e. la durée d'un pulse multiplié par β). Les bits contenus dans les paquets sont choisis aléatoirement et uniformément (autant de "0" que de "1") et chaque paquet transmis est unique.

Ce scénario peut être considéré comme "tout-ou-rien" en ce qui concerne les collisions. Entre deux paquets reçus en parallèle, les bits sont soit tous alignés, et donc reçus en même temps (à cause de leurs backoff respectifs et du délai de propagation), soit aucun ne l'est. Mais si tous les bits d'un paquet sont alignés avec ceux d'un autre flux, cela ne signifie pas que tous sont altérés (cf. Figure 2.4). En effet, si deux paquets de 8000 bits entrent en collision, seul les bits "0" peuvent être écrasés par les bits "1" de l'autre paquet. Avec une charge aléatoire et uniforme, cela arrive en moyenne un bit sur quatre. C'est exactement ce qui est montré dans la Figure 3.10. Cette figure montre le nombre de paquets ayant un certain nombre de bits altérés. La plupart des paquets observés dans ce scénario présentent un nombre d'erreurs avoisinant les 2000 bits sur les 8000 bits au total.

Cependant, avec des valeurs plus petites de β, d'autres groupements de paquets sont plus communs, avec un nombre d'erreurs autour de 3000 et de 3750 (et une plus large distribution). Cela s'explique par une plus grande contention et par des collisions impliquant plus de deux paquets simultanément. Les valeurs observées peuvent également être obtenues de manière analytique.

La Figure 3.10 montre que le nombre de bits en collision n'est pas homogène dans un réseau mono-saut. Quand les paquets entrent en collision, le nombre de bits altérés est soit autour de 2000, soit autour de 3000, soit autour de 3500. L'explication intuitive est la suivante : 2000 bits erronés signifient que deux paquets sont entrés en collision. Comme les bits ont autant de chance d'être un "0" ou un "1", la probabilité d'altérer un bit lors de la collision entre deux bits et de $\frac{1}{4}$, ce qui donne en effet $8000 \times \frac{1}{4} = 2000$ bits en collision. 3000 bits en collisions signifient que trois paquets sont en collisions, et pour 3500 bits quatre paquets, etc.

On peut généraliser cette intuition par induction ce qui donne l'équation suivante :

$$c = b \times \frac{1}{2} - \frac{1}{2^n} \tag{3.1}$$

où c est le nombre de bits altérés d'un paquet donné, b est le nombre total de bits des paquets impliqués, et n le nombre de paquets impliqués dans la collision.

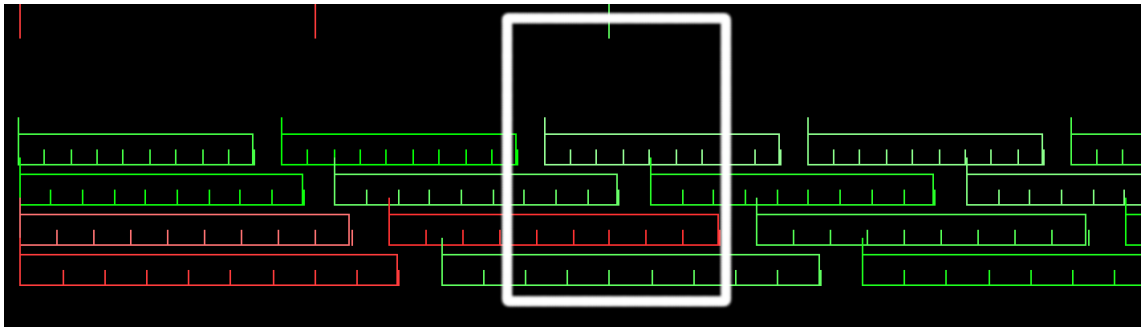


FIGURE 3.6 – Mode chronogramme : paquets de 10 bits en concurrence.

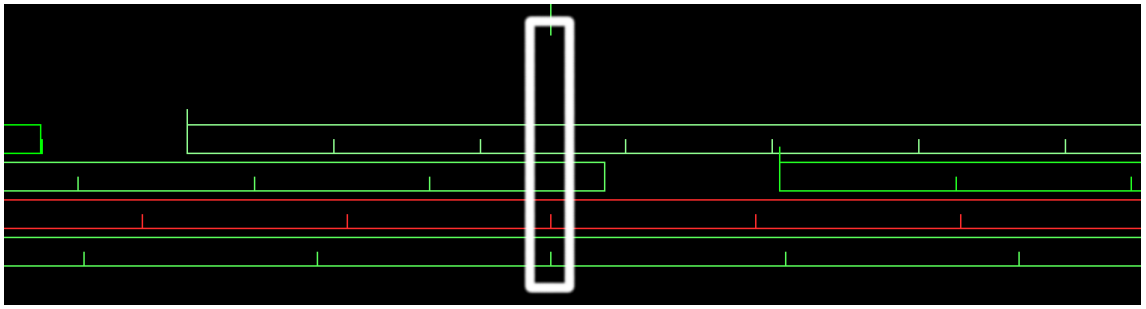


FIGURE 3.7 – Mode chronogramme : zoom sur des paquets en collisions.

Évidement ce phénomène étant grandement aléatoire, il est normal d'observer une certaine variabilité dans les valeurs évoquées plus haut, ce qui explique l'étalement dans la visualisation des données de collisions. Cependant, les simulations arrivent effectivement à des valeurs très proches ce qu'on calcule avec l'équation [3.1], ce qui renforce notre confiance dans les résultats produits par le simulateur.

3.6/ ÉVALUATION DES PERFORMANCES

Dans cette section, nous allons rapidement examiner les capacités et performances de BitSimulator via deux exemples. Le premier montrera la rapidité du simulateur sur un cas de faible envergure. Le second démontrera la capacité du simulateur à gérer des très larges réseaux denses. Pour les deux scénarios, on s'intéressera au temps d'exécution, à la mémoire consommée et au nombre d'événements traités dans la simulation. Les simulations de cette section ont été réalisé en utilisant la dernière release en date lors de la rédaction de ce manuscrit, à savoir la release numéro 0.9.4 sur un simple PC portable milieu de gamme.

BitSimulator présente deux modes de fonctionnement. Par défaut, le simulateur pré-calcule les voisins de chaque nœud (étant donné qu'on s'intéresse principalement à des réseaux statiques). Ainsi, à chaque fois qu'un nœud transmet quelque chose, le simulateur peut rapidement déclencher les événements de réception sur tous les nœuds concernés. Cependant cette méthode a un coût en mémoire élevé. L'autre mode de fonctionnement ne pré-calcule pas les listes de voisins. Ceci permet une forte économie de mémoire dans certains scénarios ; l'économie faite dépend de la densité du réseau simulé et des communications. Cependant, à chaque transmission, le simulateur doit recalculer

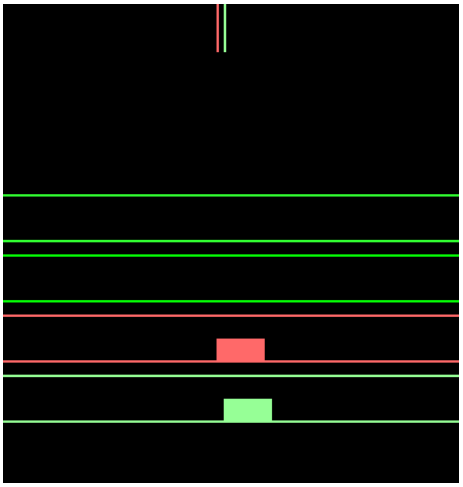


FIGURE 3.8 – Mode chronogramme : zoom sur les bits en collisions.

les voisins du nœud émetteur, ce qui ralentit la simulation. Cette fonctionnalité permet également de gérer la mobilité, étant donné qu'on ne considère pas de liste de voisins fixes (cependant, la mobilité des nœuds n'a pas encore été implémentée, cette fonction est avant tout utilisée pour gérer des problèmes de mémoire). Pour le petit scénario, on utilisera le mode sans pré-calcul des voisins, et pour le grand scénario, on montre les résultats pour les deux modes de calcul des voisins.

Les informations concernant la mémoire utilisée et le temps d'exécution sont issues de la commande :

```
/usr/bin/time -f "\t%E real,\t%U user,\t%S sys \t%M memory" \
./bitsimulator -D directoryContainingMyScenario
```

Quant au nombre d'événements et le nombre d'événements par seconde (events/s), ils sont affichés dans la console par défaut par le simulateur. On rappelle que la durée "real" désigne le temps effectif d'exécution, la durée "user" désigne le temps passé dans du code utilisateur et que la durée "sys" désigne le temps passé dans du code du noyau du systeme d'exploitation.

3.6.1/ PETIT SCÉNARIO

Pour ce petit scénario, nous avons simulé le réseau suivant :

- Réseau carré de 6 000 000 nm de coté
- 5 004 nœuds
- rayons de communications de 500 000 nm
- densité moyenne de 101 voisins/nœud

Dans le réseau décrit ci-dessus, les nœuds commencent par exécuter la phase d'adressage de SLR, ensuite un nœud envoie une série de 10 messages de 1000 bits à un autre nœud situé à l'autre bout du réseau (la route reliant les nœuds sources et destinations traverse toute la longueur du réseau). De plus, le réseau simulé est relativement grand (au vu des échelles considérées) : la diagonale du réseau représente $\frac{6000000\times\sqrt{2}}{500000} \approx 17$ rayons de communications.

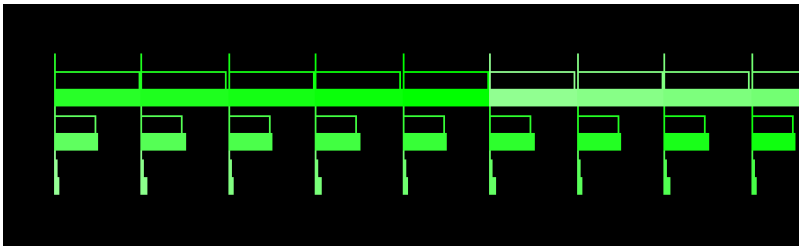


FIGURE 3.9 – Différents espacements temporels des paquets pour les ratios $\beta = 1000$ (en haut), $\beta = 500$ (au milieu) and $\beta = 50$ (en bas).

- Mémoire utilisée : 44 Mo
- Temps d'exécution :
  - *real :* 14s 64
  - *user :* 14s 02
  - *sys :* 0s 28
  - Nombre d'événements total : 2 288 077 soit 159 748 events/s

On notera que ce petit scénario implique déjà une densité de 100 voisins, ce qui représente un réseau très dense si on se compare aux autres simulateurs de réseaux ainsi que deux flooding complets du réseau (la phase d'adressage de SLR) en plus des 10 messages envoyés.

3.6.2/ GRAND SCÉNARIO

Pour ce grand scénario, nous avons simulé le réseau suivant :
- Réseau carré de 9 000 000 nm de coté
- 85 000 nœuds
- rayons de communication de 500 000 mm
- densité moyenne de 786 voisins/nœud

Dans le réseau décrit ci-dessus, un nœud inonde le réseau avec un message de 1000 bits. Le réseau simulé est très grand (au vu des échelles considérées) : la diagonale du réseau représente plus de 25 rayons de communication et présente une densité et un nombre de nœuds qu'aucun autre simulateur n'est capable de gérer (à notre connaissance).

3.6.2.1/ AVEC PRÉ-CALCUL DES VOISINS

- Mémoire utilisée : 9979 Mo
- Temps d'exécution :
  - *real :* 24m 30s 83
  - *user :* 1449s 24
  - *sys :* 9s 74
- Nombre d'événements total : 133 887 998 soit 93 351.4 events/s

3.6.2.2/ SANS PRÉ-CALCUL DES VOISINS

- Mémoire utilisée : 5810 Mo

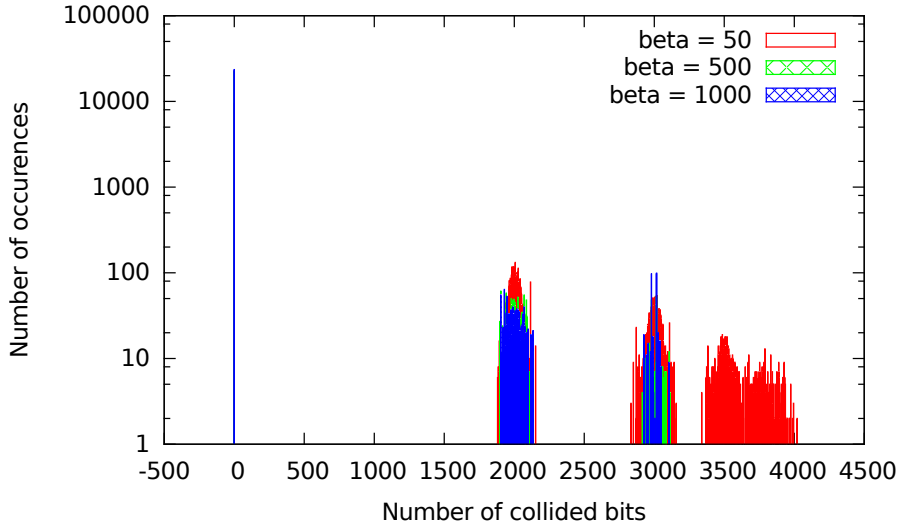


FIGURE 3.10 – Le nombre de bits en collision est soit d'environ 2000, quand deux paquets entrent en collision, soit d'environ 3000, quand trois paquets entrent en collision, soit d'environ 3750, quand quatre paquets entrent en collision.

- Temps d'exécution :
  - *real :* 29m 24s 78
  - *user :* 1744s 82
  - *sys :* 11s 87
- Nombre d'événements total : 133887998 soit 76 379.4 events/s

Ces informations sont données seulement à titre indicatif ; elles sont issues d'une unique exécution du simulateur. Le temps d'exécution dépend évidemment de la machine sur laquelle est lancé le simulateur. Cependant, ces résultats montrent que le simulateur est capable de gérer des scénarios de grande envergure sur un simple PC portable milieu de gamme.

On notera que pour les deux instances du grand scénario, le nombre d'événements simulés est exactement le même. Cela met en avant le fait que les simulations sont équivalentes et que le mode de calcul des voisins n'influe pas le déroulement de la simulation. Cependant, les fichiers de logs peuvent ne pas être exactement les mêmes et il est possible que dans certains cas les résultats soient légèrement différents en fonction du mode de calcul de voisin utilisé. En effet, l'ordre dans lequel les voisins sont considérés peut être différent d'un mode à l'autre, le comportement du simulateur peut être très légèrement différent.

3.7/ COLLABORATION

Le simulateur a été principalement développé au début de ces travaux de thèse étant donné que le besoin d'un tel outil s'est fait très vite ressentir. L'expérience de mon co-encadrant Dominique Dhoutaut dans la conception de simulateurs de ce genre a été particulièrement utile lors de la conception et du développement de BitSimulator, notamment pour la conception du moteur à événements discrets. Différentes fonctionnalités

et algorithmes ont été ajoutés tout au long de la durée de cette thèse en fonction des besoins rencontrés.

BitSimulator a déjà été utilisé par quelques personnes en dehors des encadrants de cette thèse et de moi-même, ce qui nous a permis d'avoir de premiers retours quant à la prise en main du simulateur.

Contributeurs et développeurs

— Lina Aliouat, doctorante de la même équipe de recherche et travaillant également sur les nanoréseaux.
— Florian Büther, doctorant allemand qui à implémenté un certain nombre de fonctionnalités dans le simulateur et à participer au développement global. Son expertise en C++ nous a été précieuse. Les contributions contenues dans le chapitre 6 ont été faites en collaboration avec lui.

Utilisateurs

— Yacine Benchaïb qui a participé aux études concernant l'endormissement dans le chapitre 7
— Différents stagiaires et étudiants en projet, notamment un étudiant de l'université de Virginie aux États-Unis Tucker Wilson en stage durant 2 mois et Mohammed Rahmany en stage durant 1 mois.

4

DEDeN, ESTIMATEUR DE DENSITÉ

Les communications dans les réseaux très denses ont certaines particularités. La méthode naïve du pure flooding ne fonctionne pas (ou très mal) à cause d'une redondance très forte, mais provoque également un usage désastreux du canal et des ressources des nœuds. Cependant, les protocoles plus élaborés se heurtent souvent aux capacités limitées de chaque nœud. De plus, dans les nanoréseaux les ressources ne sont disponibles qu'en très faible quantité. Ainsi, le manque de mémoire, d'énergie, de puissance de calcul (ou de ressource physique en général) couplé à la très grande densité potentielle peuvent empêcher un nœud de maintenir une liste exhaustive de ses voisins. La connaissance du nombre de voisin permet une meilleure configuration des algorithmes et protocoles réseau. L'utilité de cette information sera pleinement éprouvée dans le chapitre 5 traitant du backoff flooding et quelques exemples d'usages seront donnés dans ce chapitre en section 4.6.

Dans cette partie, je présente DEDeN (Density Estimator for Dense Networks), un algorithme conçu pour les réseaux (très) denses, en particulier dans les nanoréseaux, et qui permet aux nœuds d'estimer leur nombre de voisins (on parle parfois de degré d'un nœud, de la densité de nœud, de densité du voisinage, de densité de voisin, de densité locale ou simplement de densité). DEDeN a la propriété remarquable d'être configurable ; c'est-à-dire que la confiance et la marge d'erreur de l'estimation peuvent être ajustées en fonction des besoins de l'utilisateur, et ce, avec un coût en nombre de paquets échangés prédictible. En fonction de la façon dont il est initié, il permet à un unique nœud d'estimer son nombre de voisins, mais DEDeN peut également être exécuté sur l'entièreté du réseau. Cette dernière façon de faire est plus efficace car elle permet à tous les nœuds d'obtenir une estimation de la densité locale en une seule exécution de l'algorithme. Aussi, l'algorithme peut être exécuté à chaque fois qu'une estimation est nécessaire, et en particulier durant la phase d'initialisation du réseau (avant de commencer à réellement l'utiliser). DEDeN n'utilise que très peu de mémoire : un simple compteur est nécessaire à son fonctionnement. Les simulations montrent qu'il est capable de donner des estimations de très bonne qualité avec très peu de paquets échangés, bien moins qu'en utilisant une simple méthode basée sur l'envoi de paquets HELLO. Il est capable de fonctionner sur des réseaux à faible densité, mais c'est, à notre connaissance, le seul estimateur proposé pour les réseaux très denses (des milliers de voisins directs).

Comme expliqué plus tôt, à cause du manque de ressources disponibles sur les nœuds, les algorithmes de routage élaborés qui essaient de trouver les retransmetteurs optimaux ne fonctionnent pas correctement, voire pas du tout. D'un autre coté, des inondations et des géocasts sont simples à mettre en place, mais ne sont pas efficaces dans leur forme naïve si la densité est trop haute. Heureusement, ils peuvent être optimisés en utilisant

la connaissance de la densité locale pour limiter le nombre de retransmissions. (Des applications seront données plus tard.)

Les simulations utilisant BitSimulator qui sont présentées dans ce chapitre ont été réalisées avec la version 0.9.2 du simulateur. Les contributions présentées de se chapitre reposent principalement sur l'article [6].

4.1/ TRAVAUX CONNEXES

Les réseaux sans-fil multi-sauts ont largement été étudiés durant ces 20 dernières années et de nombreux protocoles ont été proposés pour les réseaux ad hoc, de capteurs, et véhiculaires ainsi que pour de multiples autres types de réseaux qui reposent sur une infrastructure. Les nanoréseaux sont différents de ces derniers de part les points suivants :

- La très faible capacité de calcul des nœuds et le peu de mémoire disponible.
- Chaque nœud peut avoir un très grand nombre de voisins (des milliers voire des millions peuvent être envisagés).
- La capacité d'avoir plusieurs paquets sur le canal en parallèle (sans causer de perte).
- Aucun mécanisme de positionnement global n'est disponible.
- La capacité des nœuds à récolter de l'énergie de l'environnement et donc de pallier la consommation d'énergie liée aux communications, si ces dernières peuvent être suffisamment étalées dans le temps.

Les applications opérant sur un réseau sans fil multi-sauts peuvent avoir des schémas de communication très différents, par exemple de la collecte de données, de la diffusion, un ou plusieurs flux unicast, principalement des échanges locaux, etc. L'inondation est le schéma de retransmission le plus basique, les nœuds retransmettent (une seule fois) tous les paquets qu'ils reçoivent. Cette méthode est en général très inefficace, des protocoles de routage plus avancés sont en général utilisés pour gérer la transmission de paquets. L'inondation dans un réseau très dense (milliers de voisins) est extrêmement inefficace étant donné que la plupart des retransmissions ne propagent le paquet qu'à très peu de nouveaux nœuds (voire aucun). De plus, des contre-mesures doivent être prises pour éviter que la contention du canal et les collisions deviennent dramatiques.

Les protocoles de routage multi-sauts utilisent en général des méthodes proactives (comme OLSR [19]) ou réactives (comme DSR [29]), et parfois un mélange de deux. Les routages réactifs reposent sur une inondation pour découvrir de nouvelles routes, et donc ne sont pas utilisés dans des réseaux très denses. Les méthodes proactives essaient de construire et de maintenir des routes, et ont en général besoin d'avoir une connaissance détaillée de (au moins) leurs voisins directs. Par conséquent, la mémoire requise dans un réseau très dense est trop élevée (d'autant plus dans les nanoréseaux).

Les protocoles de routage sans état sont de meilleurs candidats pour les nanoréseaux très denses dans la mesure où ils ne nécessitent pas de grande table de routage ni d'inondation régulière du réseau. Dans ce type de routage, les nœuds forwardent le message s'ils se considèrent être sur le "chemin" entre la source et la destination, on appelle ce type de routage le geocasting ou routage géographique. Ce calcul peut être fait si des informations de positionnement sont connues, comme avec le GPS dans les systèmes

macros. Les différences entre ces différents type de routage sont expliqué dans [22]. Autrement, des coordonnées peuvent être obtenues relativement à des nœuds spécifiques, SLR (Stateless Linear Routing, routage linéaire sans état) pour les réseaux 3D [55] est un bon exemple de type de routage. Comme on le verra plus tard, SLR utilise tout de même une forme de flooding et peut donc souffrir d'une trop grande densité.

Dans le monde macro, les nœuds accèdent au canal et le capture les uns après les autres pour envoyer un paquet (cf. Figure 4.1). Une des solutions communes aux problèmes causés par l'inondation est de faire en sorte que les nœuds retirent le paquet de leurs buffers (ils droppent le paquet) de transmission dès qu'ils ont observé un certain nombre de copies de ce paquet (cf. Figure 4.2). Avec TS-OOK, les (re) transmetteurs ne sont pas forcés d'accéder au canal de manière séquentielle, et de nombreuses copies du paquet peuvent être transmises en parallèle (cf. Figure 4.3). Avec une implémentation naïve, les nœuds ne détecteraient les copies que bien trop tard et auraient déjà commencé à transmettre leur propre copie. Une solution serait, pour chaque retransmetteur, d'attendre un temps aléatoire avant de commencer à transmettre. Ce temps dépend de la densité locale du réseau. Durant ce temps aléatoire, le nœud peut compter les copies qui ont été envoyées dans son voisinage. Dans cette idée est exploitée dans le chapitre backoff flooding 5 des réseaux comprenant de nombreuses communications, de nombreux paquets différents peuvent coexister en même temps, cela requiert alors des buffers de grande taille.

Il existe un moyen distribué très simple, mais très efficace (comme nous le verrons ensuite) d'adapter ce genre de protocoles à des réseaux denses : la retransmission aléatoire. Habituellement, les nœuds transmettent le paquet avec une probabilité inversement proportionnelle à leur nombre de voisins. En fonction de la densité, la probabilité de transmissions peut être choisie de façon à contrôler le nombre de retransmissions et donc de gérer la contention du canal.

L'algorithme que nous proposons permet aux nœuds de connaître leur densité locale (i.e. leur nombre de voisins) tout en tenant compte des contraintes des nanoréseaux. En particulier, un algorithme distribué dans les nanoréseaux ne devrait pas construire une liste (potentiellement très longue) de voisins afin de les compter, ni envoyer de nombreux messages pouvant consommer le peu d'énergie disponible et rapidement saturer le canal. Dans les réseaux macros, plusieurs solutions existent, mais ne sont pas suffisamment adaptées aux nanoréseaux. Pour une présentation des algorithmes non-utilisables dans notre contexte, comme les téléphones mobiles et les réseaux de capteurs (des réseaux à faible densité en comparaison), on peut se référer à la section Related Work de [18].

4.1.1/ ESTIMATION DU NOMBRE DE NŒUDS ACTIFS

Bianchi et al. [12] proposent une méthode pour estimer le nombre de nœuds en compétition pour l'accès au canal dans les réseaux 802.11 ; c'est-à-dire le nombre de nœuds essayant de transmettre dans un certain intervalle de temps. Cette estimation se base sur l'observation de la proportion de temps où le canal est occupé et sur les moments où les collisions ont lieu. En utilisant des outils analytiques incluant le taux de collisions (déduit de l'observation) et des filtres de Kalman, les auteurs sont capables d'estimer le nombre de nœuds en compétition pour l'accès au canal. Cette méthode est utilisée durant le fonctionnement normal du réseau et fournit une estimation en temps réel.

On notera que cette méthode estime le nombre de nœuds actifs (qui transmettent des données). Dans les nanoréseaux, le nombre de nœuds inactifs peut être très grand, ce qui peut être trompeur pour ce genre de protocole d'estimation. Nous avons besoin d'une méthode d'estimation qui fonctionne même si le réseau n'est pas actif, de ce fait, cette méthode n'est pas appropriée pour les études menées durant cette thèse.

4.1.2/ DIP

DIP (Density Inference Protocol, Protocole d'inférence de densité) [44] est un protocole de la couche MAC qui estime le nombre de voisins dans un réseau de capteurs macros.

DIP utilise la contention du canal pour inférer la densité locale. Les nœuds estiment la densité durant des intervalles de temps (que l'on appelle slot) divisés en durées égales. Chaque nœud envoie un paquet durant un slot choisi aléatoirement et uniformément où chaque slot a la même probabilité d'être choisi. Pour éviter les biais causés par les interférences et les collisions (plusieurs paquets envoyés en même temps), DIP base son estimation sur le nombre de slots où le canal est libre (sans communication). DIP se passe en plusieurs étapes (ou itérations), c'est-à-dire plusieurs intervalles de temps. Chaque nœud calcule son nombre de voisins n de façon statistique en utilisant la formule suivante [44] :

$$n = \log \frac{E(0,n)}{m} / \log\left(1 - \frac{1}{m}\right) \tag{4.1}$$

où $E(0,n)$ est le nombre de slots libres perçus par le nœud, et m le nombre total de slots.

DIP n'est pas approprié pour nos travaux pour plusieurs raisons. Dans les nanorésaux qui utilisent TS-OOK, la définition d'un canal libre est fondamentalement différente de la définition dans des réseaux standards (802.11 par exemple) : plusieurs paquets peuvent être en cours de transmission en même temps (cf. explication TS-OOK en section 2.1), et donc il n'y a pas de concept tel que "canal libre". Si on s'intéresse au coût de l'algorithme, le nombre de messages échangés dans DIP est le nombre total de nœuds multiplié par le nombre d'itérations. Le coût de la méthode d'estimation que nous proposons peut être inférieur au nombre de nœuds si le réseau est suffisamment dense. Étant donné que DIP utilise la contention du canal pour faire son estimation, tout le trafic (autre que celui généré par DIP) doit être arrêté durant son exécution pour avoir une estimation correcte. La formule présentée plus haut demande la manipulation de nombres à virgule flottante alors que notre méthode n'utilise que des nombres entiers et des nombres à virgule fixe, ce qui est pertinent lorsque que l'algorithme doit être exécuté sur des équipements aux capacités particulièrement réduites.

4.1.3/ ESTREME

Estreme [18] est une méthode statistique pour estimer le nombre de voisins. Elle opère dans les réseaux où tous les nœuds sont actifs, effectuant une transmission, de façon périodique mais aléatoire au cours d'une certaine durée, comme cela peut être le cas dans les réseaux de capteurs. Dans Estreme, le temps est divisé en périodes. Lors d'une période, chaque nœud doit être actif à un moment aléatoire. Cette activité doit pouvoir être perçue par tous les voisins. Cet événement peut être l'envoi d'un probe, d'un beacon ou autre. Lorsqu'un nœud souhaite estimer la densité de son voisinage, il attend le prochain événement sur le canal. Ce temps d'attente est utilisé pour calculer une estimation

de la densité. Il vise des réseaux avec “une centaine de voisins” (traduit de’ [18]). Tous les nœuds estiment la densité de manière passive, le surcoût est donc très faible.

Tel quel, Estreme ne peut pas être utilisé dans notre contexte, car il n’y a pas d’événement périodique dans le réseau. Cependant, il peut être implémenté d’une façon similaire à notre propre estimateur, dans une phase dédiée durant laquelle chaque nœud estime son voisinage. Dans ce cas, pour éviter les collisions, les périodes ont besoin d’être relativement longues, étant donné que tous les nœuds envoient un paquet. De plus, la probabilité utilisée dans Estreme n’est pas adaptative (elle est en fait inférée par le réseau lui-même, ainsi que par son activité), et la variété de densités supportées par le protocole (autour d’une centaine de voisins) est bien plus petite que la variété de densités supportées par notre estimateur.

4.1.4/ ESTIMATION DU NOMBRE DE PUCES EN RFID

Zero-One Estimator Protocol (ZOE) [65], est un protocole probabiliste qui estime le nombre de puces d’un grand système RFID composé d’un lecteur et de nombreuses puces RFID. Dans ce protocole, le lecteur utilise plusieurs ”rounds”, dans chacun de ces rounds, il envoie une requête à toutes les puces. Les puces répondent au lecteur avec une probabilité fixe. Le lecteur peut ensuite estimer le nombre de puces en utilisant le nombre de “rounds vides” (sans aucune réponse).

Bloom Filter based Cardinality Estimator (BFCE) [36] fonctionne de façon similaire, mais utilise une phase supplémentaire pour avoir une estimation grossière dans un premier temps, afin de fixer correctement la probabilité de réponse à utiliser ensuite et obtenir une estimation bien plus fine.

Les méthodes utilisées dans les systèmes RFID ne sont pas adaptées à notre problème. En effet, un système RFID est un cas particulier de réseau à un saut. DEDeN, quant à lui, fonctionne sur les réseaux multi-sauts. De plus, la probabilité utilisée par ZOE est fixée à l’avance et les formules utilisées pour la probabilité calculée est complexe et donc inappropriée aux contraintes des nanoréseaux. DEDeN, en changeant la probabilité d’émission de façon dynamique est capable de gérer un large panel de densité. Et finalement, dans un système RFID, les équipements ne sont pas homogènes étant donné que ce genre de système comporte un lecteur de puce RFID. Le lecteur peut synchroniser toutes les puces étant donné qu’il peut toutes les joindre directement. La synchronisation peut être compliquée dans des nanoréseaux vastes (en terme de nombre de sauts).

4.2/ ALGORITHME

L’algorithme se divise en deux phases, qui utilisent respectivement des paquets `Initialization` et des paquets `DensityProbe`.

4.2.1/ PHASE D’INITIALISATION

Un beacon `Initialization` est diffusé dans le réseau par un nœud initiateur (ou qui souhaite obtenir une estimation). La réception de ce message déclenche la deuxième phase, l’estimation du nombre de voisins.

Il est important de noter que le temps que peut mettre le paquet d'initialisation à traverser le réseau (de façon multi-sauts) et atteindre tous les nœuds n'a pas d'importance majeure. Par nature, un nœud et tous ses voisins directs vont recevoir ce message d'initialisation dans un intervalle de temps extrêmement court. Cet intervalle étant de plusieurs ordres de grandeurs inférieur à la durée d'un round, il s'avère négligeable.

L'algorithme n'a pas de problème particulier si plusieurs nœuds tentent de l'initier en parallèle. Un nœud ignore simplement tout nouveau beacon reçu pendant qu'il exécute déjà le protocole. De plus, le nœud initiateur n'a pas d'importance dans le protocole, donc avoir deux nœuds (ou plus) qui initient l'algorithme (plus ou moins) en même temps aura juste pour effet de faire commencer certains nœuds plus tôt dans certaines parties du réseau, cela peut donc même avoir un effet positif.

La synchronisation entre les nœuds voisins est cruciale dans notre protocole, car les rounds de la phase d'estimation doivent commencer et se terminer (plus ou moins) en même temps. Le délai de propagation du message initial (indiquant le début du protocole pour tous les nœuds) peut être relativement grand dans un réseau multi-sauts, ce qui signifie que tous les nœuds ne commenceront pas à exécuter le protocole au même moment. Cependant, notre algorithme n'a pas besoin d'une synchronisation globale de tous les nœuds, mais d'une synchronisation locale (entre les nœuds voisins). Or les nœuds d'un même voisinage reçoivent le message d'initialisation dans un intervalle de temps très court [1].

L'algorithme peut être exécuté lors du déploiement du réseau ou à des moments prédéfinis à l'avance, si les nœuds possèdent des horloges suffisamment précises. Sinon, il serait aussi possible de déclencher l'algorithme à la réception d'un signal direct provenant d'un équipement macro, si un tel équipement est disponible. Et enfin, il peut être exécuté dès que des nœuds ont besoin de connaître leur nombre de voisins, en fonction de la dynamique de l'application par exemple, du déplacement des nœuds, de nœuds qui entrent ou sortent du réseau, etc. Dans ce dernier cas (mais pas uniquement) l'algorithme répand un message (beacon) dans le réseau, ce dernier indique aux nœuds de commencer l'algorithme. Le message peut être envoyé via le pure flooding, mais aussi en utilisant un protocole plus optimisé, tel que le backoff flooding (présenté plus tard) ou un flooding probabiliste, avec des paramètres "prudents" (conservateurs) afin d'atteindre tous les nœuds.

4.2.2/ PHASE D'ESTIMATION DES VOISINS

Après l'initialisation, l'estimation sera produite dans un intervalle de temps dont la durée maximale est connue d'avance. Cependant, dans la majorité des cas, l'estimation sera fournie avant cette borne maximale. L'exécution de DEDeN est découpée en plusieurs étapes que l'on appelle "rounds" R_i, de durées égales. Durant chaque round, chaque nœud est susceptible d'envoyer un unique paquet `DensityProbe` avec une probabilité p_i qui dépend du round actuel.

Cette probabilité p_i commence avec une valeur proche de 0 et augmente à chaque round R_i. Durant chaque R_i, le nœud compte le nombre de paquets reçus de la part de ses voisins. En connaissant la probabilité de transmission (qui est la même pour tous

1. Un bit parcourt la distance d'une portée de communication (de 0.5 mm) en approximativement 167 fs, ce qui est négligeable en comparaison de la durée de chaque round ($10\,\mu$s).

durant un R_i donné), le nœud estime le nombre de voisins dans sa portée de communication. La probabilité augmentant à chaque round (multipliée par un paramètre growthRate à chaque round), le nombre moyen de paquets envoyés durant un round augmente également. (La durée d'un round est choisie en considérant une densité maximale possible afin d'éviter le plus possible les collisions durant l'exécution du protocole). Commencer avec une probabilité p_1 très basse assure que même si le réseau est très dense, très peu de probes seront envoyés, gardant ainsi le risque de collision et l'utilisation du canal très bas. Plus la probabilité initiale est basse, plus la densité maximale gérée par DEDeN est élevée.

La durée des rounds peut être connue par chaque nœud à l'avance ou être spécifiée dans le message d'initialisation. Pour éviter que tous les nœuds n'envoient leur paquet en début de round, provoquant de nombreuses collisions qui pourraient fausser l'estimation, les nœuds attendent un temps aléatoire compris entre 0 et la durée totale d'un round avant de transmettre leur probe. Les collisions sont rendues d'autant moins probables en choisissant une durée de round adéquat. Un bit (un pulse) a une durée de 100 fs, la transmission d'un probe de 10 bits représente une activité sur le canal de 1 ps. Dans nos simulations où nous considérons une densité maximale possible de 1 000 000 voisins pour un nœud, nous avons choisi une durée de round de 10 μs qui rend les collisions très peu probables même si 1 000 probes sont envoyés durant un seul round. On notera que le nombre maximum de probes attendus dépend des paramètres voulus par l'utilisateur. L'intervalle de confiance désiré de l'estimation, ou la densité maximale que l'on veut pouvoir supporter, par exemple.

L'algorithme obtient son estimation finale quand l'une de ces deux conditions est remplie :

— La probabilité de transmission p_i atteint 1. Dans ce cas, à la fin du dernier round, le nombre de probes reçus indique directement le nombre de voisins. Ce cas peut survenir quand le nombre de voisins est très bas.
— Le nombre de probes reçus durant le round en cours dépasse le seuil th_i. Les valeurs utilisées pour un seuil th_i de chaque R_i peuvent être pré-calculées (et stockées dans une table sur chaque nœud) avant le déploiement du réseau ; ces seuils correspondent à un degré de confiance lors de l'estimation du nombre de voisins. Cet aspect est détaillé dans la section 4.4. De plus, les nœuds continueront d'envoyer des probes avec une probabilité croissante pour quelques rounds supplémentaires. Dans les environnements non homogènes, cela permet aux nœuds n'ayant pas encore une estimation satisfaisante de continuer l'algorithme sans être perturbés par d'éventuels voisins ayant, eux, déjà terminé.

Un résumé du fonctionnement de DEDeN est donné dans l'algorithme 1.

4.3/ PROPRIÉTÉS

Nous proposons un algorithme distribué qui estime la densité locale en particulier pour les réseaux très denses, comme peuvent l'être les nanoréseaux

DEDeN, notre estimateur, fonctionne également en présence de trafic concurrent. Dans ce cas, il faut que le taux de collisions soit acceptable (que le canal ne soit pas trop chargé) et il faut être capable de discerner les paquets de DEDeN des paquets des autres communications, par exemple en utilisant un simple identifiant dans l'en-tête des

Algorithm 1 Algorithme d'estimation de densité exécuté par les nœuds.

Input : p_1, growthRate > 1, th_i for each round
Output : estimation

$i = 1$
while TRUE **do**
    send a probe with probability p_i using a backoff
    wait for the round R_i to end
    k = number of received probes in this round
    if $k >$ threshold th_i **then**
        estimation = k/p_i
        end algorithm
    end if
    if $p_i \geq 1$ **then**
        estimation = k
        end algorithm
    end if
    p_{i+1} = $p_i \times$ growthRate
    i++
end while

paquets. Lorsque DEDeN est utilisé sans trafic concurrent, des paquets de 1 bit sont suffisants puisque les paquets ne comportent absolument aucune donnée. (Par simplicité, nous nous sommes préoccupés du cas sans trafic concurrent. Nous prévoyons de nous occuper du cas avec dans des études futures.)

L'algorithme a un comportement statistique : il ne donnera pas (en général) le nombre exact de voisins, mais plutôt une bonne estimation de ce dernier. Il est capable de donner un résultat fiable sur une grande plage de densité, allant de quelques nœuds à des millions de voisins. En revanche, il n'est pas du tout optimal (en terme de nombre de messages échangés) s'il est utilisé sur des réseaux très peu denses. Son efficacité augmente rapidement avec la densité. Il est capable de gérer des réseaux dont la densité est hétérogène (les différentes parties du réseau sont plus ou moins denses). Il profite du possible multiplexage des trames sur le canal pour obtenir son estimation en très peu de temps, mais il peut également être utilisé sur n'importe quel type de réseau. Quelle que soit la densité du réseau, il ne sature jamais le canal (par construction) et il ne nécessite que très peu de mémoire et de puissance de calcul pour fonctionner correctement. Il utilise seulement un compteur afin de garder une trace du nombre de paquets reçus à chaque round (le même compteur peu être écrasé à chaque round). Et enfin, il n'utilise que des nombres entiers et des nombres à virgule fixe.

4.4/ CONFIANCE EN L'ESTIMATION ET CALCUL DES SEUILS

Du point du vue d'un nœud, la réception de probe peut se modéliser en utilisant une loi de distribution binomiale avec les paramètres suivants :

— k (le nombre de succès), le nombre de probes reçus durant le round
— p^{trans} (la probabilité de succès d'une épreuve de Bernoulli), la probabilité de transmission

— n^{real} (le nombre d'épreuves), le nombre réel de voisins

L'approche naturelle pour calculer le nombre de voisins $n^{estimated}$ est de considérer la moyenne de la distribution binomiale :

$$n^{estimated} = \frac{k}{p^{trans}} \tag{4.2}$$

Cette valeur représente le nombre de voisins le plus probable. Mais la valeur k est issue d'un processus aléatoire. Cette valeur est susceptible de changer entre chaque instance de l'expérience aléatoire, causant des variations indésirables sur l'estimation.

La confiance que l'on peut avoir dans cette estimation dépend de la répartition (de l'étalement) de la distribution binomiale associée. Cependant, n^{real} n'est pas connue, et donc la répartition réelle de la distribution binomiale non plus.

L'équation suivante donne la probabilité de recevoir k probes pour un nombre de voisins n et une probabilité de transmission p donnés :

$$Pr(k, n, p) = Pr(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{4.3}$$

On peut illustrer ce mécanisme en considérant tout d'abord le cas où la probabilité d'envoi d'un probe (p) est de 0.1 alors que le nombre de voisins (n_{real}) est de 100. Dans ce cas, la loi de probabilité est montrée dans la Figure 4.4. Observer 10 probes est le cas le plus probable, mais il est en réalité possible d'en observer entre 0 et 100, avec des probabilité très différentes et potentiellement proches de 0.

Dans un deuxième temps, nous pouvons considérer le cas où p vaut toujours 0,1 mais où n_{real} vaut maintenant 200. La probabilité d'observer un certain nombre de probes change et est également affiché dans la Figure 4.4. 20 est maintenant le nombre de probes observé le plus probable. Mais il est néanmoins possible (avec une probabilité moindre) de n'en observer que 10.

En considérant ces deux exemples, si 10 probes ont été effectivement observés, on se rend compte qu'il peut y avoir en réalité 100 ou 200 voisins, mais il est plus probable qu'il y en ait 100 que 200. À partir de cette base, nous avons écrit un programme qui construit de manière itérative une courbe représentant pour chaque n^{real} potentiel la probabilité de recevoir un certain nombre de probe. Ceci permet notamment la construction des seuils de confiance.

Il est possible de choisir un intervalle qui est le plus probable de contenir la valeur réelle de n ; cette probabilité est généralement fixée à 95%. Pour un certain nombre de voisins attendus, on peut alors choisir un seuil d'erreur e^{max}. L'équation 4.2 devient alors :

$$n^{real}_{min} = \frac{k}{p^{trans}}(1 - e^{max}) \tag{4.4a}$$

$$n^{real}_{max} = \frac{k}{p^{trans}}(1 + e^{max}) \tag{4.4b}$$

La confiance P que l'on peut avoir dans l'estimation la probabilité que l'intervalle $[n^{real}_{min}, n^{real}_{max}]$ contienne n^{real} :

$$P = \frac{\sum_{n=n_{min}^{real}}^{n=n_{max}^{real}} Pr(k, n, p^{trans})}{\sum_{n=0}^{n=\infty} Pr(k, n, p^{trans})} \tag{4.5}$$

La Figure 4.5 illustre l'équation (4.5) pour différentes valeurs de p et de k. Dans cet exemple, les courbes montrent que $n^{estimated} = 600$ est le nombre de voisins le plus probable, ce qui peut aussi être calculé grâce à l'équation 4.2. Les deux lignes verticales délimitent l'intervalle d'erreur requise $I = [n_{min}^{real} = 570, n_{max}^{real} = 630]$, soit un $e^{max} = 0.05$. Ici, la confiance P a pour valeur 0.36, 0.52 et 0.78 respectivement pour des valeurs de (p, k) de (0.125,75), (0.25,150) et (0.5,300). Cela signifie que 36% de la surface sous la courbe "$p = 0,125$; observé = 75" de la Figure 4.5 se trouve dans l'intervalle $[570; 630]$. Formulé plus simplement, si l'on observe 75 probes envoyés avec une probabilité de 0.125, il n'y que 36% de chance que le nombre réel de noeuds soit de 600 avec une marge d'erreur de 5% maximum. Par contre, oberserver 300 probes envoyés avec une probabilité de 0.5 fait monter cette confiance à 78%.

Comme le montre la Figure 4.6, pour une probabilité de transmission p^{trans} donnée, la confiance P augmente avec le nombre de probes reçus. Pour un p^{trans} donné, le seuil th représente le nombre de probes qui doivent être reçus pour que P atteigne la confiance requise (ici, une probabilité de 95% que l'intervalle I contienne n^{real}).

La confiance requise, ici, 95% de chance que I contienne n^{real}, peut être atteinte quand P concorde avec le e^{max} requis.

Pour une probabilité p^{trans} et un nombre de voisins n^{real} donnés, le nombre de probes k reçus peut ne pas atteindre le seuil de confiance requis. Dans ce cas, l'algorithme exécutera le prochain round en multipliant p^{trans} par le facteur growthRate. De ce fait,dans ce round suivant, étant donné que p^{trans} a augmenté, plus de probes seront envoyés.

Ce processus peut être répété jusqu'à ce que la confiance requise soit atteinte.

À noter que le choix de l'intervalle de confiance impacte fortement le coût (en nombre de paquets) de l'algorithme. Les petites valeurs de e^{max} (donc un intervalle de confiance plus fiable) requiert un seuil th bien plus élevé. Intuitivement, plus la précision de l'estimation est élévée, plus le coût sera élevé. Cela est montré dans la Figure 4.6 où les courbes utilisent une erreur maximale de $e^{max} = 10\%$ et où bien moins de probes sont nécessaires pour atteindre la confiance requise (ici 95%).

Le coût théorique de l'algorithme est représenté dans la Figure 4.7. Ce coût est exprimé en nombre de paquets reçus par chaque nœud. On peut remarquer que ce coût n'augmente plus quand le nombre de voisins devient très grand. Cela signifie que pour les hautes densités, seulement une petite proportion de nœuds ont besoin d'envoyer un probe afin d'obtenir une estimation de bonne qualité.

La Figure 4.7 montre aussi la présence de dents de scie. Ces pics apparaissent car chaque round utilise une valeur de p^{trans} adaptée pour une certaine plage de densité, et le nombre de probes reçus est plus bas quand le nombre réel de voisins est plus proche du début de plage que de sa fin. Le paramètre growthRate influe sur les variations du coût de l'algorithme. Les petites valeurs de growthRate limitent ces variations au prix d'un plus grand nombre de rounds (pour une même p^{trans} de départ) et donc d'une durée d'exécution plus longue.

La valeur de growthRate et de la probabilité de départ p_1 (et donc implicitement le nombre maximal de round), ont de forts impacts sur l'exécution de l'algorithme. Si p_1 est trop

élevée, alors les nœuds (ou une majorité) vont recevoir suffisamment de probes pour atteindre le seuil th_1 durant le premier round. Dans ce cas, le nombre de paquets envoyés est directement proportionnel à la densité. De plus, les risques de collisions augmentent également avec la densité, ce qui peut altérer la qualité de l'estimation. Un autre problème peut survenir dans ce cas-là : si trop de probes sont reçus simultanément sur un nœud, ce dernier ne sera pas capable de les recevoir correctement (cf. section 6.3), ce qui peut également dégrader le résultat obtenu étant donné que certains probes seront envoyés, mais pas reçus. Ces paramètres sont donc à choisir avec précaution en fonction des conditions de déploiement du réseau, mais aussi des besoins de l'application.

Intuitivement, plus la probabilité de départ est faible, plus la densité maximale que DEDeN peut gérer est élevée. Quant à growthRate, plus il est faible plus le nombre de rounds maximal est élevé (étant donné que la probabilité maximale est toujours de 1), ce qui force donc le protocole à prendre plus de temps à s'exécuter. Cependant, un nombre de rounds faibles implique un risque de "débordement". Autrement dit, si la probabilité à un moment donné est trop élevée, trop de nœuds vont envoyer leur probe durant un round ce qui peut saturer le réseau (le canal ou les nœuds) et donc nuire à la qualité de l'estimation.

4.5/ PERFORMANCES DE L'ESTIMATION DE DENSITÉ

Les résultats présentés dans cette partie ont été générés par BitSimulator, dans lequel DEDeN a été implémenté. Il est disponible dans les versions mises en ligne du simulateur.

Pour rappel, DEDeN est un algorithme distribué qui fournit, pour chaque nœud dans le réseau, une estimation de son nombre de voisins en injectant des paquets dans le réseau. On s'intéresse donc principalement à deux paramètres qui définissent l'efficacité de l'algorithme : l'erreur de l'estimation (la différence entre l'estimation et le nombre réel de voisins) et le nombre de paquets envoyés tout au long de l'exécution de l'algorithme. Ces deux métriques mesurent respectivement la qualité et le coût de l'estimation. On notera qu'on ne s'intéresse pas au coût de la phase d'initialisation étant donné qu'elle ne fait pas réellement partie de l'algorithme et qu'il est possible de l'éviter sans altérer les résultats (en commençant à une date fixée à l'avance ou via un équipement macroscopique envoyant un signal de départ perçu par tous les nœuds du réseau).

Le temps requis pour l'exécution de l'algorithme est particulièrement court. Cela est dû aux échelles de temps minuscules impliquées dans les communications dans les nanoréseaux. Si ces communications ne sont pas ralenties (par un manque d'énergie par exemple), l'exécution de l'algorithme peut se terminer dans un temps de l'ordre de la μs.

Pour valider les résultats théoriques donnés dans la section 4.4, nous avons utilisé BitSimulator ainsi que plusieurs scénarios. La topologie du réseau est simple : une surface carrée ayant pour côté 2.5 mm. Les communications sont omnidirectionnelles avec un rayon de 0.5 mm. Les probes utilisés contiennent 10 bits. Entre 100 et 40 000 nœuds sont placés de façon aléatoire dans cette surface en utilisant une distribution uniforme. On notera que malgré la disposition uniforme des nœuds sur la surface, la densité (nombre de voisins) n'est pas la même partout dans le réseau : d'une part à cause du caractère aléatoire du placement, mais surtout, car le nombre de voisins au niveau des bords du réseau est plus faible.

4.5.1/ QUALITÉ ET QUALIFICATION DE L'ERREUR

DEDeN est conçu pour fournir une estimation dont la confiance et la marge d'erreur sont configurables. Dans un premier temps, nous comparons les distributions d'erreurs théoriques et celles obtenues dans les simulations. Le scénario a 40 000 nœuds. DEDeN est configuré de manière à ce que l'estimation et le nombre réel de voisins ne diffère pas de plus de 10% avec une probabilité d'au moins 95%. L'erreur d'estimation de l'algorithme est définie comme étant $e = (n^{estimated} - n^{real})/n^{real}$.

Sur la Figure 4.8 on observe que les erreurs dans les simulations se comportent comme les erreurs théoriques. Dans ce scénario, les résultats des simulations sont même meilleurs que la confiance initialement requise. Cela est dû au fait que la courbe des résultats théoriques est tracée à partir du seuil de confiance exact, c'est-à-dire pour que les nœuds estiment leurs voisinages avec une erreur d'au plus 10% pour une confiance d'au moins 95%. Dans la simulation, le nombre de paquets reçus dépasse ce seuil, on obtient donc une meilleure qualité d'estimation.

Pour aller plus loin, nous avons simulé plus de 200 scénarios avec le même environnement, mais avec un nombre de voisins variant entre 10 et 4200, $growthRate = 1.6$, $e^{max} = 10\%$, et une confiance de 95%. La Figure 4.9 met en évidence l'erreur d'estimation pour tous les nœuds dans tous les scénarios. La forme globale est la même que sur la Figure 4.8 et prouve la capacité de DEDeN à produire une estimation au moins aussi bonne que celle requise quelle que soit la densité réelle du réseau.

4.5.2/ LE COÛT EN TERMES DE PAQUETS ENVOYÉS

On définit le coût de DEDeN comme étant le nombre de paquets envoyé pour obtenir une estimation avec la précision souhaitée.

On peut distinguer deux cas différents. Dans le premier cas, DEDeN a besoin de plusieurs rounds pour produire une estimation, alors que dans l'autre, il se termine en un round unique. Nous donnerons d'abord plus de détails sur le premier (et principal) cas, puis nous discuterons du second. Pour étudier ce coût, nous utilisons un ensemble de 200 scénarios avec un nombre de nœuds compris entre 100 and 40 000. Il est important de rappeler que la densité dans les scénarios simulés n'est pas homogène. Les nœuds proches des bords de la surface simulée ont moins de voisins que ceux se trouvant au centre. Cela affecte les résultats et justifie de légères différences avec les calculs théoriques dans lesquels on considère un environnement infini et donc homogène.

Comme mentionné précédemment, pour que DEDeN fonctionne correctement dans des environnements non-homogènes, les nœuds participent à un round supplémentaire après avoir obtenu une estimation dont la confiance est suffisante. Cela augmente le coût de l'exécution, mais ce coût additionnel n'est significatif que lorsque le nombre de voisins est relativement petit. Sur la Figure 4.10 est représenté le nombre total de paquets envoyés pour atteindre l'estimation avec $growthRate = 1.6$ et pour différentes valeurs de e^{max}. La courbe présente un aspect similaire à la courbe théorique de la Figure 4.7, mais lissée par la non-homogénéité de la topologie simulée. La Figure 4.10, qui représente le nombre de paquets nécessaires pour atteindre la confiance requise, permet de voir que le nombre de paquets n'augmente plus quand le nombre de voisins devient très grand.

Enfin, nous avons comparé les résultats de DEDeN avec ceux du protocole DIP pour

différentes valeurs de growthRate (Figure 4.11). Pour cette comparaison, $e^{max} = 10\%$, qui est (comme montré dans la Figure 4.10) une confiance importante et coûteuse. Sur ce graphique, on remarque que pour DEDeN le nombre de paquets envoyés par nœud diminue en fonction de la densité. Dans les protocoles DIP et HELLO, nous considèrerons le cas optimal en termes de coût, c'est-à-dire celui dans lequel chaque nœud envoie un paquet lors d'une itération unique. Le nombre de paquets envoyés par nœud est donc de 1. En fonction de la valeur de growthRate, DEDeN est meilleur que DIP si la densité est supérieure de 1800 ($growthRate = 2$) à 4000 voisins ($growthRate = 1.2$). En conclusion, pour les réseaux très denses, notre algorithme nécessite bien moins de messages grâce à son efficacité croissante avec la densité.

Dans de rares cas, la densité du réseau est proche de la densité maximale D_{max} que DEDeN est supposé gérer au regard de ses paramètres (notamment le paramètre growthRate et de la probabilité de départ p_1). Dans ce cas, l'estimation peut être faite en un seul round. DEDeN est alors toujours capable de fournir une estimation correcte étant donné que le nombre de probes reçus dépasse le seuil de confiance en un seul round. Cependant, si la densité du réseau est bien plus grande que D_{max}, DEDeN ne fournit plus un résultat fiable, notamment à cause des collisions sur le canal, mais aussi à cause d'un éventuel manque de ressources pour traiter tous les probes reçus. Si chaque nœud reçoit un trop grand nombre de paquets en simultané, il peut ne pas être capable de tous les considérer. Ce phénomène est expliqué à la section 6.3. Si les nœuds ne sont plus capables de recevoir la totalité des paquets envoyés, l'estimation ne peut pas fournir un résultat de bonne qualité. Dans ces cas, le nombre de paquets envoyés est proportionnel au nombre de nœuds dans le réseau et à la probabilité initiale p_1. Pour pallier ce problème là, il faudrait soit augmenter la durée de chaque round soit commencé l'algorithme avec une probabilité de départ p_1 plus faible.

4.6/ APPLICATIONS

DEDeN estime le nombre de voisins de chaque nœud. Cette information est particulièrement utile pour optimiser des protocoles de routage ou d'autres applications présentes sur les réseaux. Dans le cadre de ma thèse, DEDeN a principalement été utilisé pour paramétrer un schéma de forward (le backoff flooding) et un mécanisme d'endormissement des nœuds. Ces deux applications seront explicitées dans les sections dédiées à ces deux contributions. Dans cette section, je vous présente une application de DEDeN à un algorithme de routage simple : le flooding probabiliste.

4.6.1/ UN SIMPLE FLOODING PROBABILISTE

La diffusion de données multi-sauts est un service d'une extrême importance dans les réseaux ad hoc sans fil. Elle est nécessaire dans de nombreuses applications et protocoles de haut niveau ; elle peut par exemple être utilisée pour répandre une information urgente dans des réseaux devant gérer des désastres. La méthode la plus simple pour implémenter cette diffusion est le flooding : chaque nœud forwarde le paquet qu'il reçoit. Mais le flooding a deux inconvénients. Premièrement, il génère un nombre considérable de paquets, un paquet par nœud ; deuxièmement, pour éviter les retransmissions multiples, chaque nœud doit mémoriser un identifiant de paquet (supposé unique).

Les optimisations traditionnelles impliquent une sélection de sous-ensembles de nœuds forwardeurs. Ce type de méthode est inutilisable dans les nanoréseaux ; notamment à cause de l'impossibilité de construire une liste complète de voisins en raison d'un réseau très dense et d'une mémoire très limitée sur chaque nœud.

Une des solutions classiques à ce problème est de limiter le nombre de forwardeurs de façon probabiliste, en attribuant aux nœuds une probabilité avec laquelle ils forwardent chaque nouveau paquet (les paquets déjà rencontrés ne sont pas retransmis). Selon un "survey" sur la diffusion probabiliste est présenté dans l'article [42], ces schémas de diffusion sont divisés en deux types : ceux à probabilité fixe et ceux à probabilité adaptative. Dans les méthodes à probabilité fixe, une même probabilité de transmission préalablement définie est utilisée par tous les nœuds. La diffusion peut être optimisée en prenant en compte certaines caractéristiques du réseau. Dans les schémas non basés sur un compteur, la probabilité dépend de plusieurs facteurs tels que la densité, la distance, la vitesse de transmission etc. Dans les schémas basés sur un compteur, "le nombre de voisins est la métrique de densité la plus utilisée" (traduit de [42]). Habituellement, la probabilité de transmission est inversement proportionnelle au nombre de voisins d'un nœud :

$$p = k/n_b \tag{4.6}$$

où k est le facteur de propagation et, dans le contexte de l'article [42], n_b (le nombre de voisins) peut être facilement obtenu via des paquets HELLO, ce qui n'est pas vrai dans le cas de réseaux très denses. Les méthodes basées sur des paquets HELLO entraînent un grand nombre d'échanges de paquets et nécessitent bien plus de mémoire pour compter correctement le nombre de voisins. C'est là que DEDeN présente un intérêt majeur : il fournit une bonne estimation du nombre de voisins dans des réseaux avec une densité très élevée.

Nous avons utilisé BitSimulator pour implémenter un flooding à probabilité adaptative qui calcule la probabilité de chaque nœud en fonction de l'estimation fournie par DEDeN. Le facteur k est utilisé conjointement à l'estimation de densité pour contrôler le nombre moyen de retransmissions dans chaque voisinage. Dans cette étude, nous présenterons principalement les limitations et risques de cette méthode simple, puis nous proposerons des méthodes plus adaptées (cf backoff flooding dans le chapitre 5 et la méthode d'endormissement proposée dans le chapitre 7).

La Figure 4.12 est issue de la simulation d'un scénario comprenant 10 000 nœuds dans lequel la diffusion des paquets nécessite plusieurs sauts (multi-hop-broadcasted). Elle montre la distribution du nombre de copies reçues par chaque nœud. Dans le cas présenté ici, le nombre moyen de copies voulues est $k = 5$ (on peut aussi parler de redondance voulue). Mais on peut observer qu'en réalité un grand nombre de nœuds n'ont pas reçu le paquet. Étant donné que chaque nœud prend la décision de forwarder ou non le paquet indépendamment les uns des autres et avec une certaine probabilité, il est possible que tous les nœuds d'un même voisinage décident de ne pas forwarder le paquet. Ici, la propagation s'est arrêtée prématurément, donc certaines parties du réseau n'ont pas reçu le message. On en déduit alors que cette valeur de k est trop petite. Augmenter k augmente également le nombre de copies du paquet envoyés (et donc reçus) à travers le réseau. Il serait particulièrement coûteux d'utiliser une valeur de k qui réduise les chances d'interruptions de propagation à une probabilité suffisamment faible.

Pour conclure, dans les réseaux très denses, spécialement si les ressources sont (très) limitées, les flooding à probabilité adaptatives sont plus efficaces que ceux utilisant une

probabilité fixe. La probabilité utilisée par chaque nœuds est déduite de la densité locale fournie par DEDeN et peut être adaptée à des environnements plus ou moins “hostiles” (au prix d’un plus grand nombre de transmissions). Ce simple flooding probabiliste ne nécessite que très peu de mémoire pour les identifiants de paquets, lui donnant un avantage sérieux face à d’autres floodings adaptatifs plus complexes. Cependant, nous avons également vu que cette méthode simpliste peut mener à des interruptions de diffusion en cas de paramétrage non adapté.

4.6.2/ AUTRES APPLICATIONS POSSIBLES

Connaître la densité locale d’un réseau peut s’avérer très utile dans certains cas. Je fais ici une liste (naturellement non exhaustive) de quelques idées qui exploitent cette information.

DEDeN a été initialement conçu pour servir au backoff flooding. Le backoff flooding (présenté dans le chapitre 5) utilise l’estimation fournie par DEDeN afin de se paramétrer de la manière la plus appropriée possible.

Dans cette thèse, nous proposons aussi, dans le chapitre 7 une méthode d’endormissement des nœuds afin d’économiser des ressources et de faire une meilleure exploitation des capacités du canal. Intuitivement, la durée que peuvent passer les nœuds à dormir dépend directement du nombre de voisins de chacun des nœuds. L’utilisation de DEDeN avec notre méthode d’endormissement n’a pas été exploitée, mais c’est une piste que nous souhaitons explorer sous peu.

L’information de nombre de voisins de chaque nœud peut également être utile pour des protocoles des couches supérieures. Par exemple, dans le cas de la matière programmable, la connaissance de la densité locale des nœuds peut être nécessaire pour garantir la solidité d’une structure. A contrario, une zone déjà suffisamment dense peut “envoyer” des nœuds vers une zone moins solide de la structure.

Les nanoréseaux peuvent aussi trouver leur place en tant que réseaux de capteurs. Ils peuvent être utiles pour faire de la détection d’attaque chimique en étant capable de repérer des molécules très tôt grâce à leur très petite taille. Il est aussi souvent envisagé d’utiliser les nanoréseaux au sein du corps humain. Il est possible d’utiliser la connaissance de la densité locale au sein d’un réseau dédié à la surveillance ou à la détection. Connaître le nombre de nœuds dans une zone donnée apporte une information quant à la qualité de la surveillance. De plus, il devient alors possible de connaître la proportion de nœuds ayant détecté un événement en plus de simplement en connaître le nombre. Cela peut éviter les faux positifs dans certains cas.

4.7/ CONCLUSION

DEDeN est un algorithme distribué d’estimation de la densité de nœuds dans les réseaux très denses, et en particulier dans les nanoréseaux électromagnétiques très denses. L’algorithme est exécuté par chaque nœud et est basé sur des rounds consécutifs. À chaque round, jusqu’à ce qu’un critère d’arrêt soit rempli, la probabilité d’envoyer des paquets augmente et la qualité de l’estimation augmente également. Cette qualité est entièrement adaptable aux exigences des protocoles de la couche supérieure et le critère d’arrêt

dépend (entre autre) de la qualité initialement souhaitée. Le coût en termes de messages échangés est prévisible. Les besoins en mémoire sont minimes : seul un compteur est nécessaire.

L'utilité de l'algorithme a été démontrée sur un algorithme probabiliste d'inondation. L'algorithme a faible coût, en termes de paquet envoyés, par rapport aux inondations classiques et diminue fortement avec la densité. DEDeN aura aussi un rôle prépondérant dans le chapitre suivant : le backoff flooding. Son champ d'application est vaste et de nombreux autres algorithmes de routage, protocoles et applications peuvent en bénéficier. Il s'agit, à notre connaissance, de la première méthode d'estimation proposée adaptée aux très denses et en particulier aux nanoréseaux.

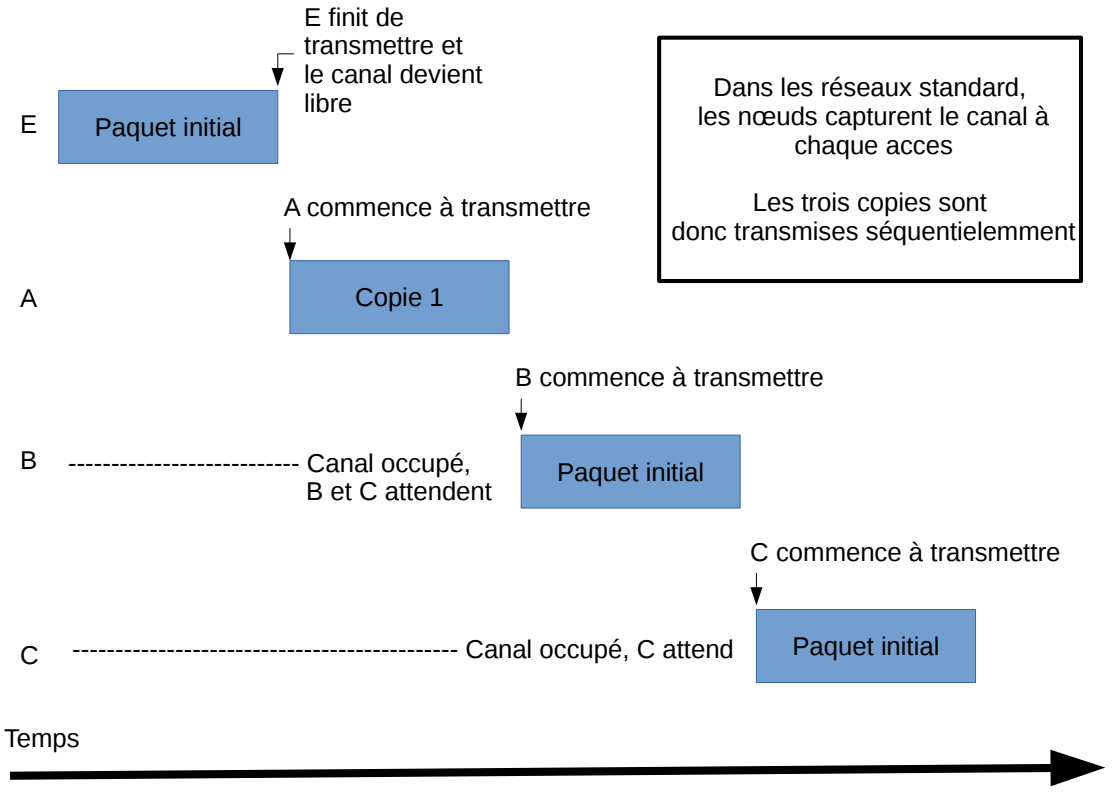


FIGURE 4.1 – Les nœuds accèdent au canal séquentiellement.

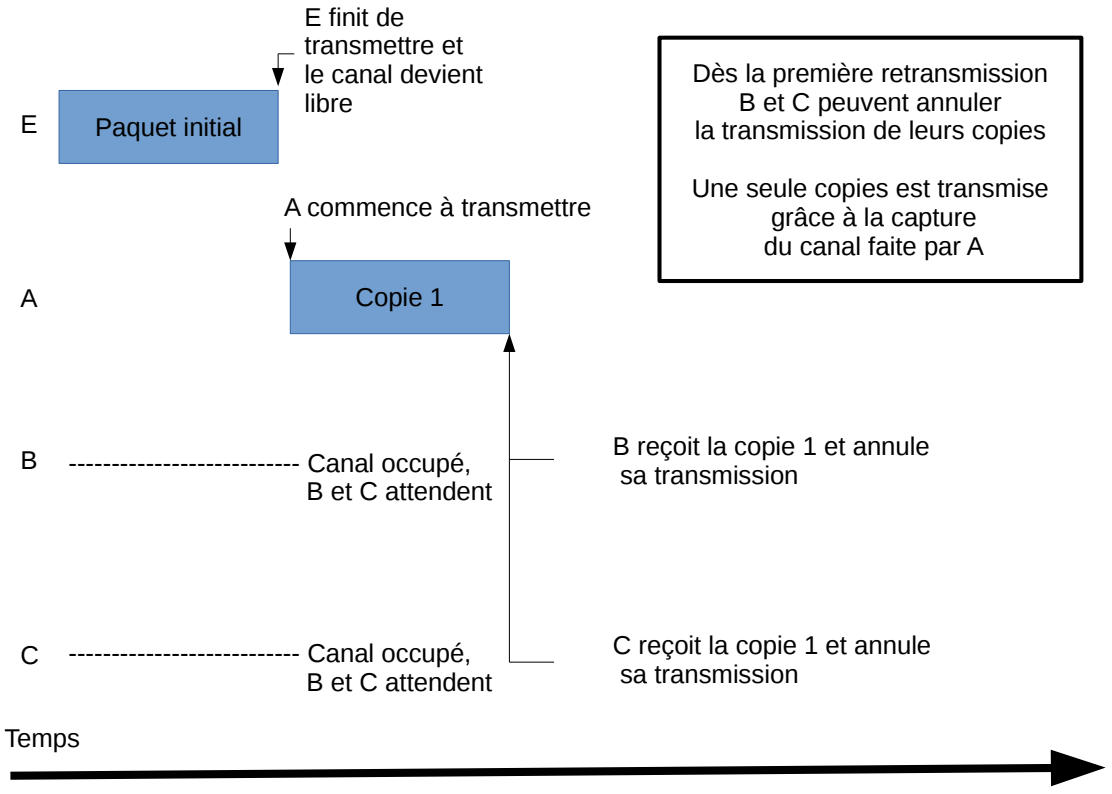


FIGURE 4.2 – Les nœuds annulent leurs transmissions.

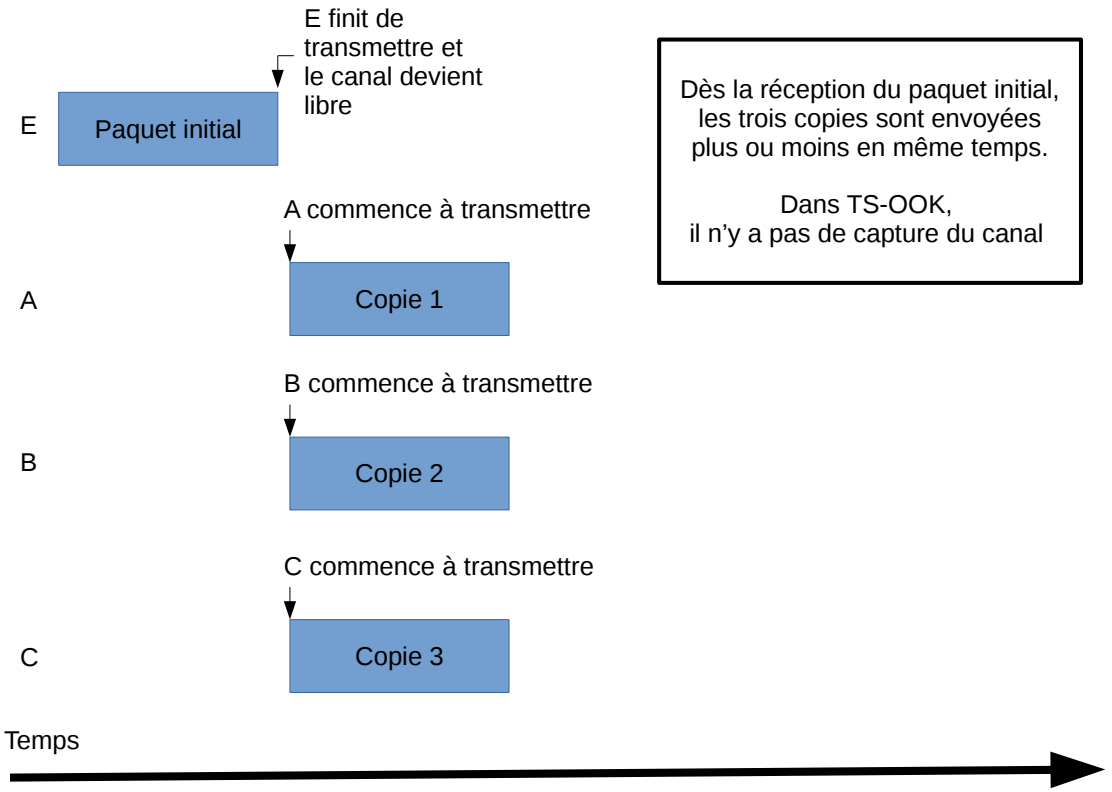


FIGURE 4.3 – Avec TS-OOK les nœuds transmettent en parallèle. Ce qui empêche les nœuds de compter correctement les copies d'un paquets

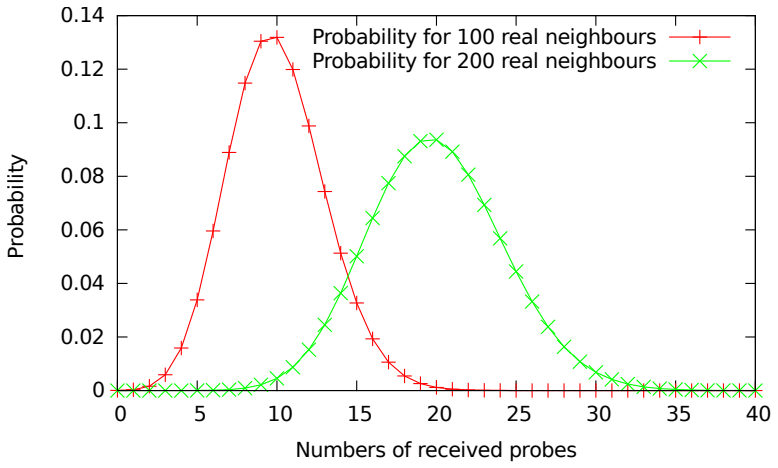


FIGURE 4.4 – Probabilité de recevoir x probes pour un nombre réel de voisins de 100 et de 200.

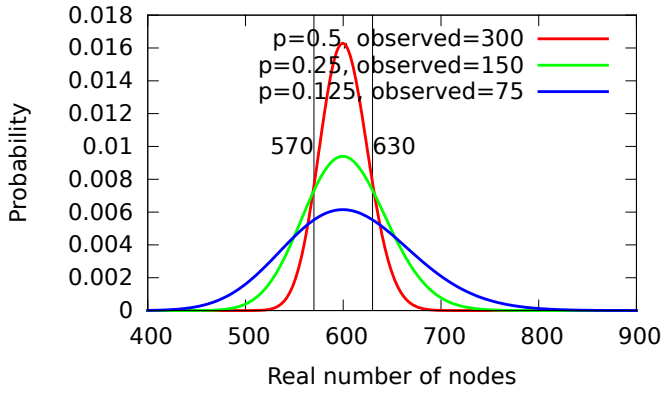


FIGURE 4.5 – Distribution de probablitité de n^{real} pour différentes valeurs de k et de p.

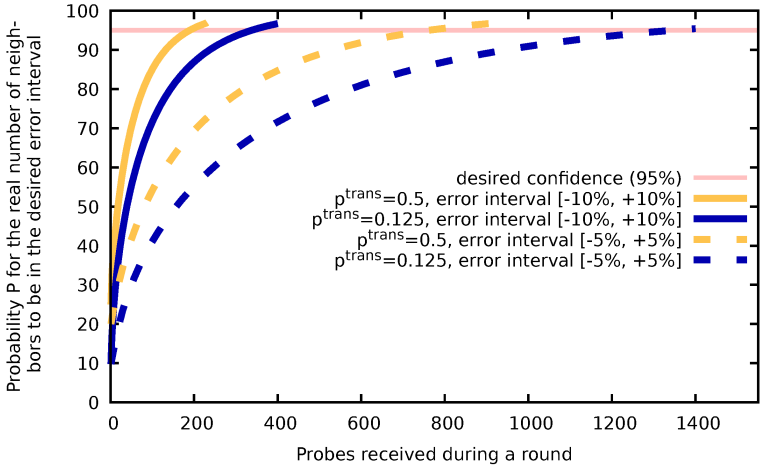


FIGURE 4.6 – Confiance en fonction de la probabilité p et du nombre de probes observés k.

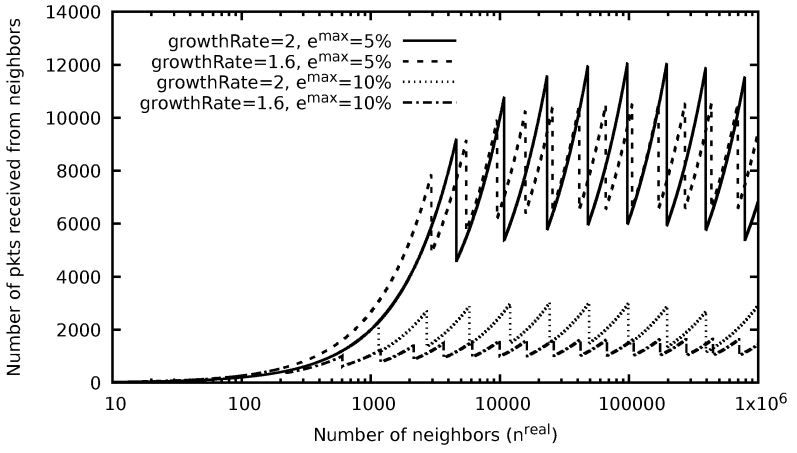


FIGURE 4.7 – Surcout théorique (nombre de paquets reçu par nœud) pour différents parametres.

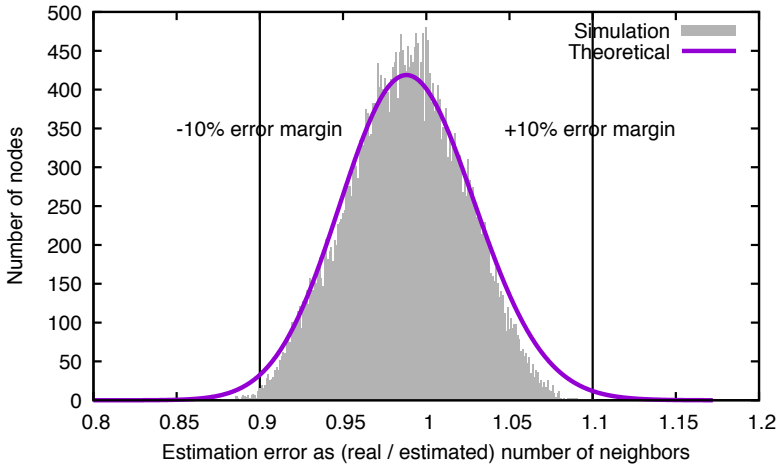


FIGURE 4.8 – Erreur d'estimation dans un scénario de 40 000 nœuds.

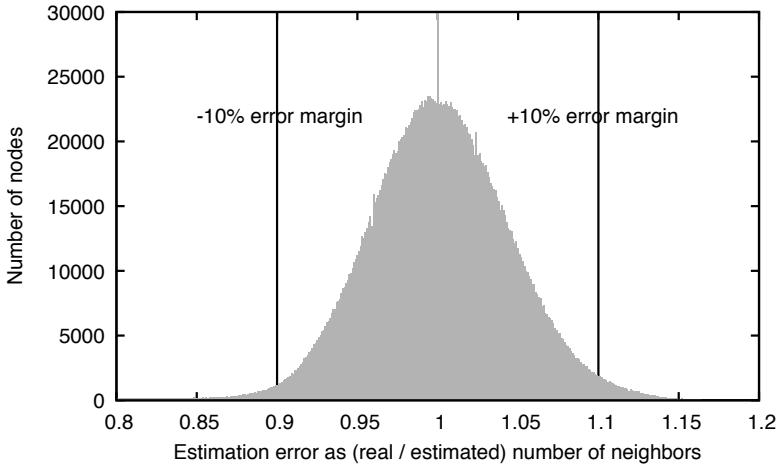


FIGURE 4.9 – Distribution de l'erreur d'estimation moyenne pour tous les nœuds dans 202 scénarios avec des densités différentes.

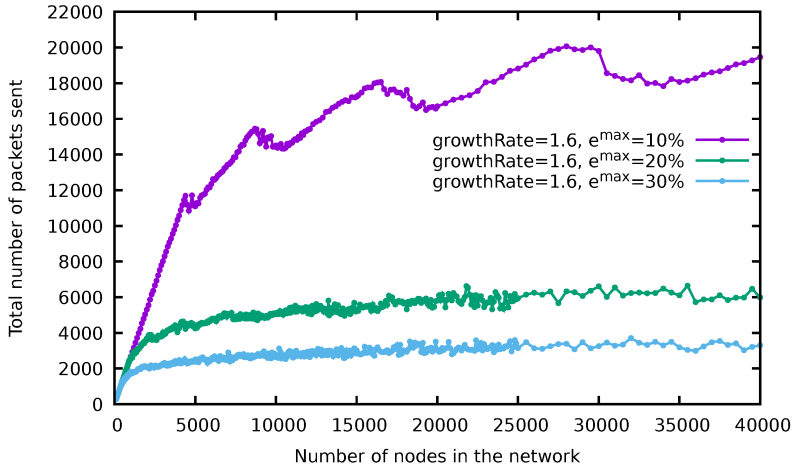


FIGURE 4.10 – Nombre de paquets envoyé en fonction de la densité du réseau et de la marge d'erreur désirée.

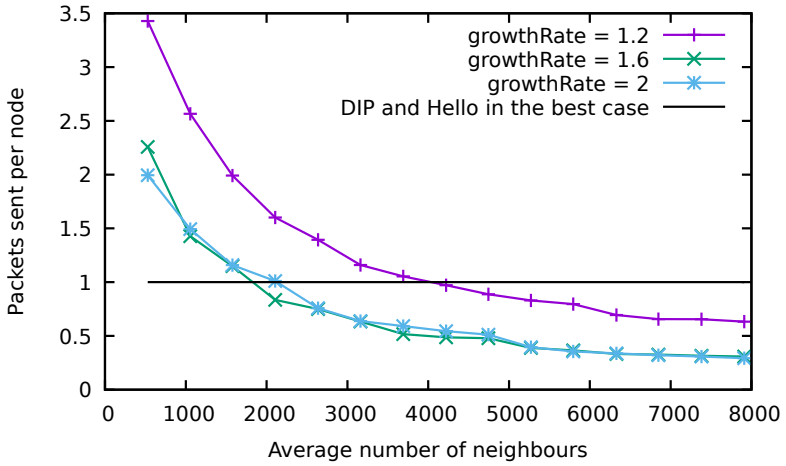


FIGURE 4.11 – Ratio $\frac{\text{paquets envoyes}}{\text{nombre de nœuds}}$.

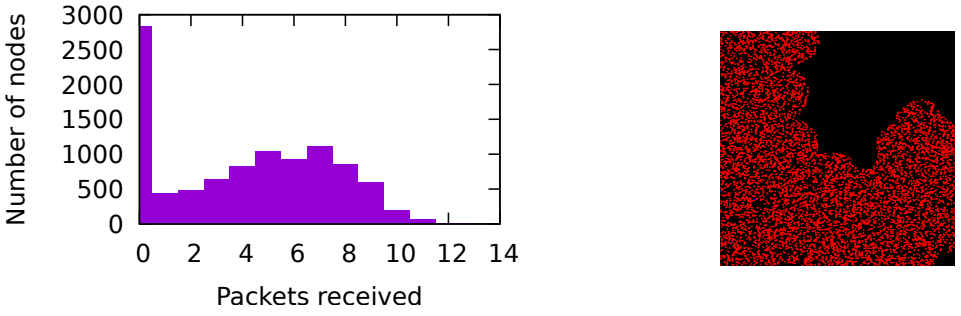


FIGURE 4.12 – Flooding probabiliste simple : distribution du nombre de copies reçues (à gauche), zone de propagation incomplète, en noir (à droite).

5

BACKOFF FLOODING

Lors du flooding dans les nanoréseaux denses, si tous les voisins réémettent tout de suite cela provoque deux problèmes. Premièrement, de nombreuse collisions peuvent apparaître et peuvent dégrader la propagation d'un message. Deuxièmement, étant donné que tous les voisins répète le message, le réseaux devient très redondant. Un grand nombre de copies sont envoyées inutilement et on observe donc une consommation de ressources importante.

Les contraintes des nanoréseaux empêchent souvent l'accès à des solutions optimales en terme de diffusion. Il n'est, par exemple, pas possible de connaitre la positions exacte des voisins d'un nœud. Le backoff flooding est une forme de flooding qui réduit drastiquement le nombre de répétitions de paquets tout en assurant une couverture maximale du réseau. Il empêche notamment les tempêtes de diffusion et réduit la consommation de ressources, principalement le nombre de paquets émis. Cet algorithme contrôle la diffusion en limitant le nombre de forwardeurs. Le backoff flooding est un algorithme de diffusion basé sur un compteur (counter-based).

Dans ce chapitre, je présente le backoff flooding et je l'analyserai selon plusieurs métriques : couverture, coût en nombre de paquets et delais.

Les simulations utilisant BitSimulator qui sont présentées dans ce chapitre ont été réalisé avec la version 0.9.2 du simulateur. Les contributions présentées de se chapitre reposent principalement sur l'article [7].

5.1/ TRAVAUX CONNEXES

La plupart des travaux connexes se basent sur un accès séquentiel au canal, de type 802.11. Dans ce cas, le canal peut être utilisé par un seul nœud à la fois sans provoquer d'erreur. Ainsi lorsque un nœud est en train de transmettre, les autres nœuds doivent attendre. La transmission d'un paquet est une "opération atomique" [25]. Les paquets sont envoyés de façon séquentielle, un à la fois. Cette contrainte, presque systématiquement rencontrée dans les réseaux macroscopiques, est en réalité très forte. Or, dans les nanoréseaux, les paquets envoyés par différents nœuds peuvent être entrelacés, comme illustré dans la section 4.1. C'est la raison principale pour laquelle la plupart des méthodes présentées dans la littérature ne peuvent pas être appliquées dans notre contexte.

5.1.1/ PURE FLOODING

La façon la plus simple de diffuser de l'information à tous les nœuds d'un réseau est l'inondation. Le paquet initial est envoyé par le nœud source, puis tous les nœuds le recevant transmettent une copie de ce paquet (on les appelle forwardeurs). Cette méthode implique un immense coût en termes de nombre de paquets échangés, défaut bien connu souvent appelé "broadcast storm problem" [51] (tempête de diffusion). Ce phénomène explique pourquoi le pure flooding n'est que rarement utilisé tel quel.

5.1.2/ FLOODING PROBABILISTE ADAPTATIF

Comme expliqué dans la chapitre 4, le flooding probabiliste est une méthode simple pour réduire le nombre de paquets transmis lors de la diffusion d'un message à travers le réseau. Cet algorithme calcule une probabilité pour chaque nœud, cette probabilité conditionne directement la transmission ou non d'un nouveau paquet arrivant sur le nœud. On peut définir la redondance du flooding probabiliste comme étant le nombre moyen de forwarders à chaque saut : $r = n \times p$, où r exprime la redondance, n le nombre de voisins participant à la diffusion et p la probabilité de transmission. Cette méthode a l'avantage d'être extrêmement simple et n'implique aucun surcoût. Cependant, du fait de sa nature probabiliste, elle présente le risque que l'information ne soit pas diffusée à tout le réseau. Pour de plus amples explications, se référer à la section 4.6.1.

5.1.3/ PROTOCOLES DE TYPE GEOCASTING

Les protocoles de type geocasting sont basés sur des informations géographiques, comme par exemple la position de nœuds, afin de prendre la décision de forwarder [1]. Le geoforwarding est souvent utilisé dans les WSN (Wireless Sensor Network) pour sélectionner le forwardeur le plus éloigné possible de la source et donc optimiser la couverture de chaque transmission, réduisant ainsi le nombre de forwardeurs nécessaires.

La position des nœuds peut être obtenue de deux façons :
- directe, via un GPS embarqué sur chacun des nœuds ;
- indirecte, via des techniques de triangulation, des positions relatives en utilisant des ancres ou encore des techniques de beaconing se basant sur la puissance du signal reçu [16].

Les méthodes de positionnement des nœuds ne correspondent pas aux contraintes imposées par les nanoréseaux. Nous avons donc souhaité développer un schéma de diffusion qui n'en dépende pas. De par leur taille minuscule et la très faible quantité d'énergie disponible, les nanonœuds ne peuvent pas embarquer de GPS ni calculer leurs positions relatives sans infrastructure.

5.1.4/ OLSR

Les protocoles se basant sur une parfaite connaissance des voisins ou sur des tables de routage complexes, tel que OLSR [19] et ses variantes, construisent de lourdes tables de routage sur chaque noeud. Ces tables contiennent les informations pour joindre chacun des autre nœuds du réseau en sélectionnant à chaque saut le forwardeur idéal (selon

une certaine métrique). Elles sont construites de manière proactive via l'échange d'information concernant les voisins directs et parfois même les voisins à deux sauts. Comme mentionné auparavant, les nanoréseaux peuvent être extrêmement denses. Ce grand nombre de voisins et le peu de mémoire disponible sur chaque nœud rendent impossible l'utilisation de ce type de protocole de routage

5.1.5/ LES RÉSEAUX HIÉRARCHIQUES

Les protocoles de routage hiérarchiques (ou clusterisés) sont fréquemment utilisés dans les WSN [8]. Le réseau est partagé en plusieurs régions (clusters), dans lesquelles un nœud prend le rôle de leader. Ce nœud est communément appelé cluster head. Ces clusters heads sont similaires à des routers dans les réseaux IP. Les informations générées par les nœuds sont routées par les cluster heads, en général vers des gateways (des passages vers d'autres réseaux, internet par exemple) ou des collecteurs de données (sink).

Le découpage en clusters n'est pas une solution adaptée aux nanoréseaux pour plusieurs raisons. Premièrement, ce type de protocole nécessite d'autres protocoles pour fonctionner, pour l'élection de leader par exemple. De plus les cluster heads représentent des fragilités (point of failure). Ce type de système implique donc l'utilisation de techniques de tolérance aux pannes. Enfin, étant donné que les cluster heads traitent toutes les données issues de leurs régions en plus des données issues d'autres clusters heads, ils consomment bien plus d'énergie et nécessitent bien plus de ressources : une plus grande capacité de calcul ou une mémoire plus importante que les autres nœuds (pour les files d'attente de paquets par exemple). Notamment, les ressources (et capacités) de réception de ces cluster head (cf section 6.3) vont être limitantes.

L'application des technique concernant les réseaux hiérarchiques dans les nanoréseaux sont une direction de travail de trois collègues de notre équipe. Ces travaux sont menés par Hakim Mabed, maitre de conférences et Julien Bourgeois professeur et par Lina Aliouat, leur doctorante.

5.1.6/ SCHÉMA ADAPTATIF COUNTER-BASED

Dans les schémas de transmission adaptatifs basés sur un compteur (counter-based scheme), les nœuds comptent le nombre de copies d'un paquet qu'ils reçoivent pour prendre la décision de le transmettre à leur tour ou non. Le comptage des paquets assure que le message est diffusé à tout le réseau. De nombreuses méthodes reposant sur ce principe ont déjà été proposées et ont fait leurs preuves dans les réseaux macro [42].

Par exemple, GOSSIP3 [27] utilise une probabilité de forwarder fixe. Si la décision est prise de ne pas forwarder, le nœud attend un certain temps, et s'il n'a pas vu un nombre minimal donné de copies (nombre calculé à partir de la probabilité de forwarder et du nombre de voisins), il enverra sa propre copie du paquet. Dans AGAR [46], le mécanisme de forward est découpé en deux phases. Dans la première, les nœuds décident de forwarder ou non le paquet avec une probabilité fixe. Les nœuds qui ont décidé de ne pas fowarder le paquet attendent. Dans la seconde phase, si le nombre de copies reçues est au dessus d'un seuil donné, les nœuds vont forwarder le paquet avec une probabilité qui dépend cette fois de la probabilité initiale et du nombre de voisins.

Le schéma adaptatif proposé dans [52] utilise le nombre de voisins pour calculer le nombre de forwardeurs. Dans les réseaux très peu denses, plusieurs forwaders sont nécessaires car chaque nœud doit couvrir une large zone. Au contraire, dans le contexte des nanoréseaux denses, moins de forwards sont nécessaires, et un seul par voisinage est en général suffisant comme nous le verrons plus tard.

De plus, si on considère un accès au canal de type 802.11, comme dans [52] par exemple, la fenêtre de backoff utilisée pour la diffusion de paquets a une taille fixe et est relativement petite (si on considère que les collisions et autres pertes ne peuvent pas être détectées de par le manque d'acquittement en mode diffusion). L'utilisation d'une même fenêtre de backoff sur tous les nœuds du réseau n'est pas appropriée aux nanoréseaux dans lesquels le nombre de voisins varie énormément. C'est pour cela que nous dimensionnons notre fenêtre de backoff en prenant en compte le nombre de voisins de chaque nœud (estimé grâce à DEDeN).

5.2/ BACKOFF FLOODING

Les méthodes de diffusions classiques ne sont pas applicables dans le cadre des nanoréseaux. De plus, les nœuds n'ayant que très peu de ressources (mémoire et capacité de calcul notamment), ils ne peuvent pas construire une connaissance complète de leurs voisinages, ce qui pourrait être très utile pour choisir le forwardeur optimal. Maintenir une simple liste des voisins directs peut devenir excessivement coûteux en mémoire quand le nombre de voisins est de l'ordre du millier ou du million. C'est pourquoi nous proposons un nouvel algorithme de diffusion counter-based demandant peu de ressources et capable de s'adapter aux densités très élevées qu'on peut trouver dans les nanoréseaux.

Le backoff flooding fonctionne comme suit. Quand un nœud reçoit un paquet pour la première fois, il enregistre le paquet et démarre un timer. Le timer est choisi aléatoirement dans une fenêtre d'attente donnée. Si ce nœud reçoit un certain nombre minimal fixé de copies de ce même paquet avant la fin de son timer, il abandonne l'idée de transmettre et efface le paquet en attente. Dans le cas contraire, il transmet sa propre copie du paquet à la fin de son timer. Dans les deux cas, il mémorise l'ID du paquet pour ne pas traiter plusieurs fois le même paquet (dans le cas où une copie arriverait après le processus, à cause des nombreux chemins disponibles dans les nanoréseaux par exemple). Chaque nœud ne traite donc jamais deux fois le même paquet et n'envoie donc jamais deux fois le même paquet.

Pour fonctionner, cette méthode combine un compteur de paquets, une fenêtre d'attente, que l'on appelle "backoff window", et un estimateur de densité utile pour adapter au mieux le backoff window à chaque nœud du réseau.

Le compteur de paquets a une importance capitale. La diffusion d'un paquet à l'ensemble du réseau peut mener à plusieurs problèmes dépendants de la densité et des communications induites par une densité relativement élevée. Par exemple, des paquets peuvent être perdus à cause de collisions. Aussi, les méthodes de diffusion multi-sauts tendent (en général) à choisir un forwardeur de manière aléatoire ; des forwardeurs non-optimaux peuvent donc être choisis et dans le pire des cas le forward peut ne diffuser l'information à aucun nouveau nœud, ce qui pourrait avorter la diffusion du message. Pour éviter cela, le backoff flooding utilise un paramètre de redondance (compteur de paquet), dont le rôle est de garantir que le paquet est transmit au moins un certain nombre de fois

dans chaque voisinage. Ce paramètre, ainsi que l'influence de taille de la fenêtre, seront étudiés dans les sous-sections suivantes.

5.2.1/ TAILLE DE LA FENÊTRE D'ATTENTE

Les méthodes de diffusion multi-sauts counter-based ont un coût modéré. Elles ne nécessitent que peu de calculs et ont une empreinte mémoire limitée et directement liée au nombre de flux en parallèle. Mais les méthodes counter-based classiques ne fonctionnent correctement que si les nœuds accèdent au canal de manière séquentielle et s'ils sont capables de retirer des paquets de leurs files d'envoi (après avoir vu suffisamment de copies, comme présenté dans la Figure 4.2 en page 34). Dans les réseaux macro, du fait de la segmentation en couches protocolaires indépendantes, le protocole de diffusion (souvent implémenté dans la couche du routage ou couche réseau) peut avoir des difficulté à effacer un paquet se trouvant en attente d'envoi dans un buffer de la couche MAC. Ces hypothèses sont en général valides dans les réseaux macro, mais les nanoréseaux n'ont quant-à eux pas un accès séquentiel au canal.

Une partie essentielle de notre méthode est l'utilisation d'une fenêtre de backoff dimensionnée avec précaution. Pour rappel : le multiplexage temporel peut être important dans un réseau utilisant la modulation TS-OOK. De nombreuses copies d'un paquet peuvent être transmises avant qu'un nœud ne les décode (comme vu en Section 4.1). Ce problème peut être résolu en ajoutant une période d'attente avant la transmission d'un paquet, appelé en général backoff, ou encore RAD (Random assessment delay) dans [42], ce qui permet de réduire considérablement le nombre de collisions. Plus important encore, cela évite que trop de transmissions aient lieu en parallèle car elles peuvent empêcher le nœud de compter correctement le nombre de copies reçues, comme montré dans la Figure 5.1. En effet, un nœud doit attendre d'avoir entièrement reçu un paquet pour le décoder et s'assurer qu'il ne contient pas d'erreur. Cependant un backoff trop long ajoute un délai supplémentaire inutilement grand, comme montré dans la Figure 5.2.

Il a été énoncé dans [42] que "il n'y a pas d'évaluation claire du rôle joué par le RAD. Il n'y a aucune évaluation de la valeur optimale du RAD dans les schémas de diffusion counter-based". Cela motive notre étude sur les divers paramètres qui régissent cette durée. Ici, on considère un RAD choisi uniformément entre 0 et t. Intuitivement, le choix de la borne maximale t impose deux contraintes :

- Une petite valeur pour t quand la densité locale est élevée augmente la probabilité d'envoi de paquets en parallèle, et la probabilité de collisions.
- Une grande valeur pour t augmente inutilement le délai de transmission de bout en bout.

C'est pour cela que nous avons choisi de calculer t en fonction de la densité locale, fournie par un estimateur (DEDeN). Le paquet est transmis après le délai le plus court choisi parmi les nœuds. Le délai effectivement induit par l'algorithme dépend donc du nombre de nœuds participant à chaque forward et de la durée maximale t. Cette notion sera étudiée dans une section dédiée 5.2.3.

Pour calculer t, considérons :

- le temps de voyage d'un bit jusqu'à la limite de la portée de communication $t_{dmax} = c/cr$, où c est la vitesse de la lumière et cr la portée de communication.
- le temps de transmission d'un paquet $t_{pkt} = T_s \times s$, où T_s et le temps entre deux bits consécutifs (cf. Section 2.1) et s le nombre de bits du paquet.

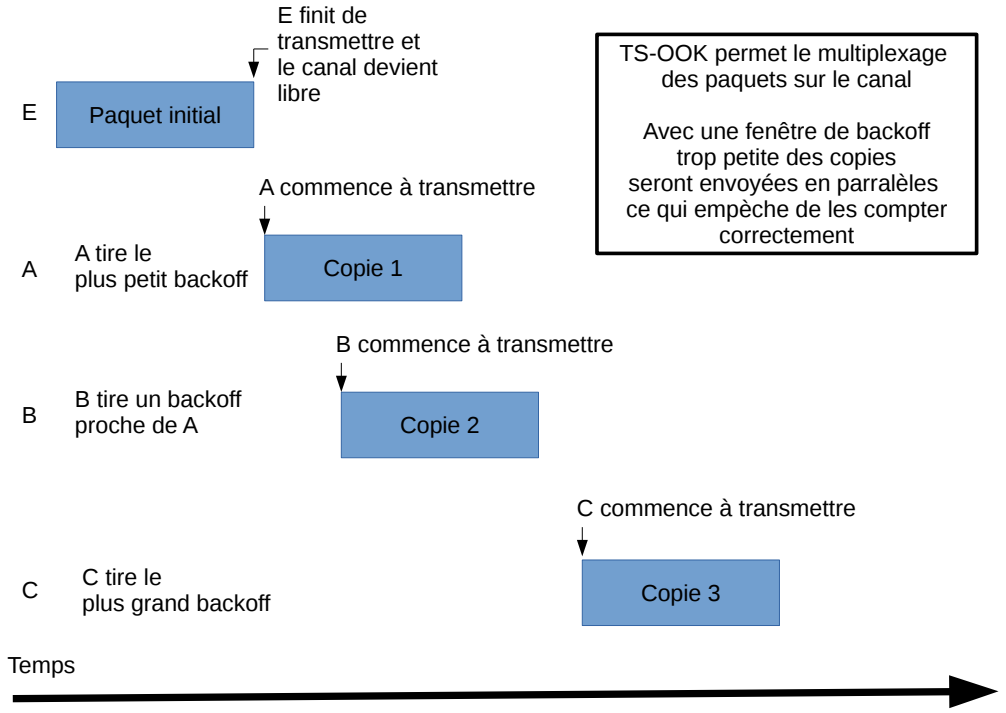


FIGURE 5.1 – Un backoff trop court ne permet pas de compter les paquets correctement.

On note t_{proc} le temps de calcul nécessaire au décodage du paquet et à la prise de décision de le forwarder ou pas. Ainsi, sans aucun backoff, le temps t_{wait} après lequel un nœud peut être sûr qu'un voisin a forwardé le paquet est donc :

$$t_{wait} = 2(t_{dmax} + t_{pkt} + t_{proc}) \tag{5.1}$$

Cette équation prend en compte le temps t_{proc} d'envoi du paquet et la transmission de sa copie aller et retour vers le nœud le plus éloigné possible (à la bordure de la portée de communication) en considérant une taille maximale pour un paquet. Ce temps peut être traité comme une constante et n'influe pas sur le comportement du protocole. Ainsi, la valeur idéale de la taille de la fenêtre de backoff est donc de :

$$t = k * n * t_{wait} \tag{5.2}$$

où n représente le nombre estimé de voisins (donnée par l'estimateur) et k un facteur multiplicateur qui affecte la distribution du nombre de forwardeurs (étudié dans la section suivante), le paramètre n sert à prendre en compte la densité locale du réseau. Ainsi, la taille de la fenêtre s'adapte automatiquement à la densité fournie par l'estimateur et ce, même dans des réseaux à densité hétérogène et dans lesquels la fenêtre idéale n'est pas la même dans tout le réseau.

5.2.2/ EFFET DU MULTIPLICATEUR k SUR LE NOMBRE DE FORWARDEURS

Comme nous avons vu dans la section précédente, la taille de la fenêtre de backoff est fonction du nombre de voisins afin d'éviter les transmissions en parallèle et les collisions

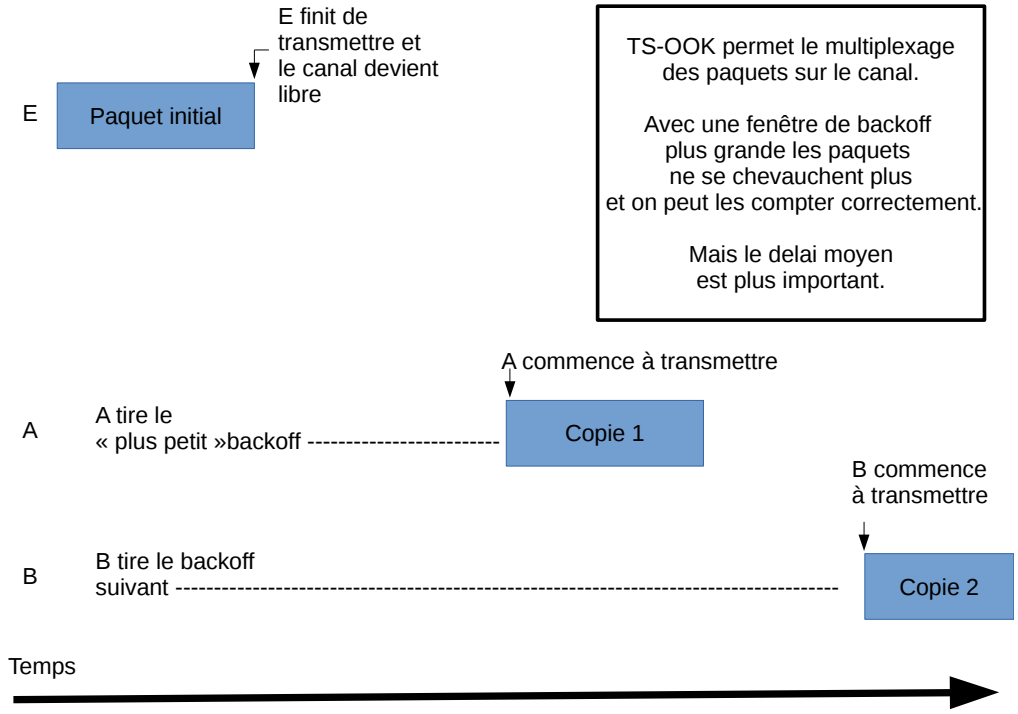


FIGURE 5.2 – Un backoff trop long ajoute un délai inutilement long

autant que possible. Il est important de noter que fondamentalement, le backoff flooding ne peut pas garantir un nombre exact de forwardeurs, et ce, pour deux raisons :

— L'entrelacement des paquets que l'on essaie de réduire en utilisant une taille de fenêtre qui prend en compte ce problème.
— La géométrie inhérente à la position des nœuds et à leurs portées de communication.

Pour appréhender l'influence de l'entrelacement des paquets, il est important de se rappeler que le backoff flooding se base sur des durées d'attente aléatoires. Il est donc possible que des nœuds choisissent des valeurs de backoff très proches dans un même voisinage (surtout si la densité est élevée). À cause des spécificités des nanocommunications au niveau des collisions et du multiplexage des paquets, de multiples copies d'un même paquet peuvent être en effet envoyées quasi-simultanément. Plus k élevé, plus les chances que plusieurs copies soient en cours de transmission au même moment sont faibles, car un même nombre de tirages sera fait dans une fenêtre de temps plus grande. Pour valider cette hypothèse et en mesurer l'ampleur, nous avons développer un programme effectuant les calculs théoriques concernant ce phénomène. Ceci est illustré dans la Figure 5.3 pour différentes valeurs de k avec une redondance voulue de 5, 1150 voisins et $t_{wait} = 8$ns. La Figure 5.3 a été produite par un programme développé par nos soins également. Ce programme ne prend en compte que les paramètres énoncé précédemment. Il ne prend pas en compte le positionnement des nœuds. On peut considérer qu'il s'agit ici d'une analyse théorique simplifiée. Dans ce scénario, les nœuds devraient idéalement recevoir exactement 5 copies d'un paquet diffusé. Mais pour des petites valeurs de k, il arrive fréquemment que la réception d'une copie (la 5ème par exemple) n'ait pas fini d'être reçue au moment où le nœud décide d'envoyer sa propre copie (elle ne peut donc pas être prise en compte). Il en résulte donc que les nœuds

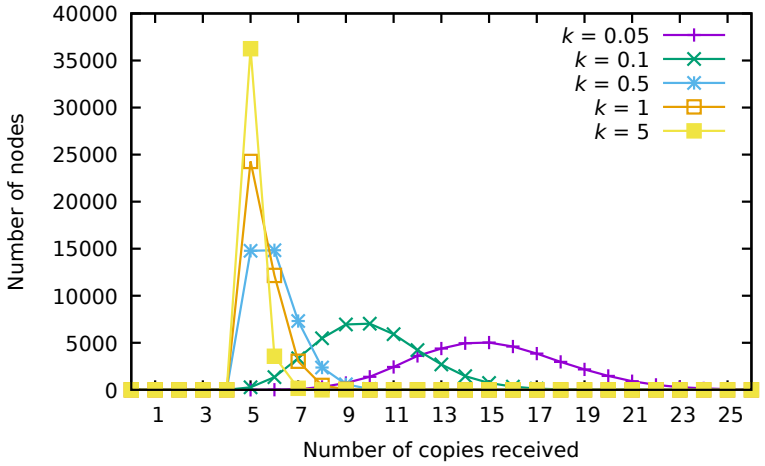


FIGURE 5.3 – Distribution du nombre de copies reçues à cause du multiplexage des paquets pour une redondandce souhaitée de 5 et pour différentes valeurs de k, les paquets sont souvent reçus 6 fois voir plus.

reçoivent bien plus que les 5 copies requises.

Le nombre de copies reçues augmente également selon la géométrie des zones couvertes par la transmission de chaque copie. Ce phénomène est illustré en deux dimensions dans la Figure 5.4 pour une redondance de 1. On considère un rayon de communication omnidirectionnel pour un transmetteur initial T (Figure 5.4a). Parmi tous les nœuds dans la zone de réception, celui qui a tiré la plus petite valeur de backoff ($R1$) forwarde le paquet (Figure 5.4b). Les nœuds dans la zone plus sombre ayant reçu une copie, ils annulent la transmission de leurs copies. En échange, nœuds touchés par la transmission initiale, un croissant n'a pas encore reçu de copie. Un nœud $R2$ dans ce croissant va alors atteindre la fin de sa période d'attente et envoyer une nouvelle copie (Figure. 5.4c). Ce faisant, certains nœuds vont recevoir une deuxième copie pendant que la zone n'ayant pas encore vu de copie se réduit. Un troisième transmetteur $R3$ envoie une copie, assurant enfin que tous les nœuds dans la zone de transmission initiale ont vu au moins une copie (Figure 5.4d). Il est facile de se rendre compte que la plupart des nœuds vont recevoir plus de copies que nécessaire durant la propagation du paquet (Figure 5.4e).

L'influence des deux raisons mentionnées plus haut peut être observée en simulation avec BitSimulator où il est facile de compter le nombre de copies effectivement reçues par chaque nœud pour différentes valeurs de k. Comme précédemment, le nombre moyen de voisins est de 1150, la redondance souhaitée et de 5, et $t_{wait} = 8$ns. DEDeN fournit une estimation avec une confiance de 95% d'avoir au plus 30% d'erreur. La Figure 5.5 présente les résultats. En comparaison avec la Figure 5.3, les courbes sont plus "plates" (le maximum se situe aux alentours de 9000 contre environ 36 000). Cela s'explique par la prise en compte de l'effet géométrique expliqué plus haut. De même, on peut observer que le nombre de copies reçues est toujours au minimum de 5. D'un côté, l'utilisation de valeurs de k plus petites que 0.5 implique une augmentation du nombre de copies transmises. De l'autre, les grandes valeurs de k augmentent la taille de la fenêtre de backoff (voir Section 5.2.1) et donc augmente le délai. Par conséquent, nous recommandons d'utiliser $k = 0.5$ car il représente un bon compromis entre le nombre de copies envoyées et le délai d'attente induit par le protocole.

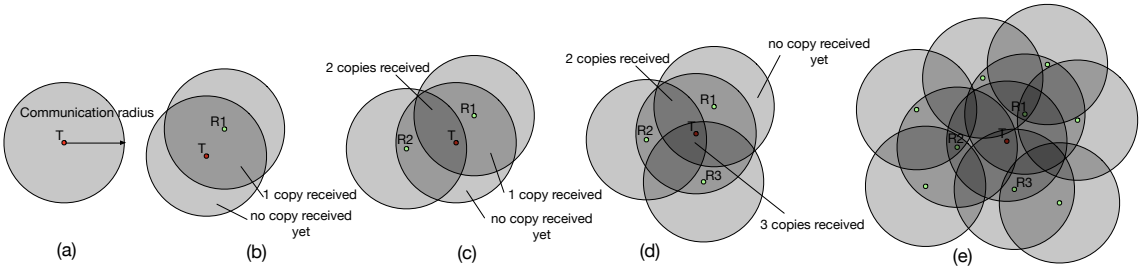


FIGURE 5.4 – Le nombre de copies reçues avec le backoff flooding dépend de la position des nœuds.

5.2.3/ DÉLAI INDUIT

Dans le backoff flooding, le backoff est avant tout utilisé pour choisir le forwardeur d'un paquet, il y a donc un comportement assez différent des systèmes standards. Dans notre protocole, à la première réception d'un paquet, le nœud choisit une valeur dans la fenêtre de backoff t. Le paquet est retardé jusqu'à la fin du backoff (quand le paquet est forwardé), ou jusqu'à ce que le nœud reçoive redundancy copies du paquet (et le paquet est rejeté), en fonction de l'événement qui survient en premier. En attendant, le paquet doit être stocké en mémoire sur le nœud.

Dans la suite, nous étudions de manière analytique le délai ajouté par le backoff flooding pour une redondance de 1. On modélise ce phénomène comme étant la plus petite valeur parmi n (n étant le nombre de voisins) tirées au hasard dans une fenêtre de t fs.

La probabilité qu'un nœud tire un backoff b supérieur à une valeur arbitraire $v \in [0, t[$ (b et v des nombres entiers) est :

$$P(b > v) = \frac{t - v}{t} \tag{5.3}$$

Cependant, l'événement décrit par "le plus petit des backoffs tirés est supérieur à v" est équivalent à l'événement "tous les backoff tirés par les n nœuds sont supérieur à v". Autrement dit ; si tous les backoffs tirés sont supérieurs à v alors le plus petit des backoffs est lui aussi supérieur à v. La probabilité de cet événement est donnée par :

$$P(b_{\min} > v) = (P(b > v))^n = \left(\frac{t - v}{t}\right)^n \tag{5.4}$$

où $b_{\min}$ est le plus petit backoff tiré. La probabilité inverse,que $b_{\min}$ soit inférieur à v, est donc :

$$P(b_{\min} \leq v) = 1 - P(b_{\min} > v) \tag{5.5}$$

La probabilité que $b_{\min}$ soit compris dans l'intervalle $]v_1, v_2]$ peut être calculée en utilisant les formules (5.4) et (5.5) :

$$P(v_1 < b_{\min} \leq v_2) = P(b_{\min} \leq v_2) - P(b_{\min} \leq v_1) \tag{5.6}$$

$$= \left(1 - \frac{v_1}{t}\right)^n - \left(1 - \frac{v_2}{t}\right)^n \tag{5.7}$$

La Figure 5.6 montre la probabilité pour que le plus petit des backoff soit dans un certain intervalle de 1% de la fenêtre de backoff, c'est-à-dire entre $xt/100$ et $(x + 1)t/100$. Cette probabilité est montrée pour plusieurs densités différentes (10, 100 et 1000 voisins), et

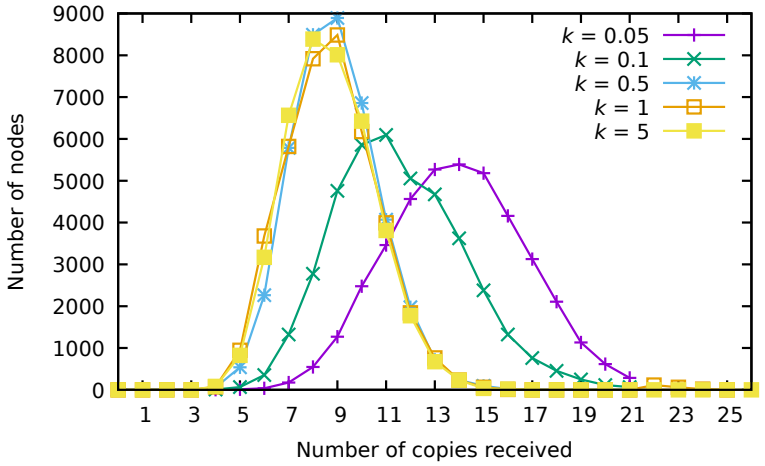


FIGURE 5.5 – Distribution du nombre de paquets reçus à cause du multiplexage des pauets ET de la géométrie pour une redondance souhaité de 5 et pour différentes valeurs de k ; les paquets sont souvent reçus 9 fois voire plus.

donc pour différentes tailles de fenêtre de backoff étant donné que la taille de la fenêtre est liée à la densité (voir l'équation 5.2). La Figure 5.6 montre, par exemple, que pour $n = 100$ la probabilité que $b_{\min}$ soit inférieur à 3% de t est supérieure à 0.95. Au regard de l'équation 5.2, 3% de t est une petite valeur, du même ordre de grandeur que le temps d'aller-retour t_{wait}. De plus, plus la densité est élevée, plus la proportion de la fenêtre de backoff utilisée par $b_{\min}$ est petite.

Le backoff flooding induit donc un certain délai, mais ce délai est négligeable et prédictible.

5.2.4/ COÛT EN MÉMOIRE ET TEMPS DE STOCKAGE DES PAQUETS

Dans le protocole backoff flooding, un nœud qui reçoit un paquet le stocke jusqu'à ce qu'il reçoive un certain nombre de copies du paquet, mais sans dépasser la durée du backoff. On s'intéresse au temps durant lequel les paquets sont stockés au niveau des nœuds, car à cause de leur taille minuscule les nœuds ne disposent que de très peu de mémoire. Les nœuds doivent également mémoriser (durant un certain temps) les ID des paquets afin de ne pas traiter un même paquet plusieurs fois. Cependant, on considérera cette grandeur négligeable, car l'ID d'un paquet est bien plus petit que le paquet lui-même. Si besoin, il est possible d'appliquer des techniques de compression afin de réduire le coût en stockage de ces identifiants. Une idée par exemple serait de ne pas stocker chaque identifiant reçu avec une liste, mais plutôt de ne stocker qu'un descriptif du type : [flux 1 ; paquet reçu 1-5] pour indiquer que pour le flux 1 les paquets de 1 à 5 ont été reçus.

On formalise la quantité de temps comme suit. Nous calculons, pour n'importe quel nombre de nœuds participants n, n'importe quelle redondance requise r et n'importe quelle valeur $v \in [0, t]$, la probabilité P de l'événement "r valeurs sont comprises dans l'intervalle $[0, v[$ et les $n - r$ autres valeurs sont dans l'intervalle $[v, t]$".

Chacune des n valeurs peut être soit strictement inférieure, soit supérieure à v. Cela signifie que l'on peut envisager 2^n cas différents. Parmi eux, $\binom{n}{r}$ cas ont r valeurs strictement inférieures à v. Cependant, ces cas ne sont pas équiprobables : la probabilité de chaque

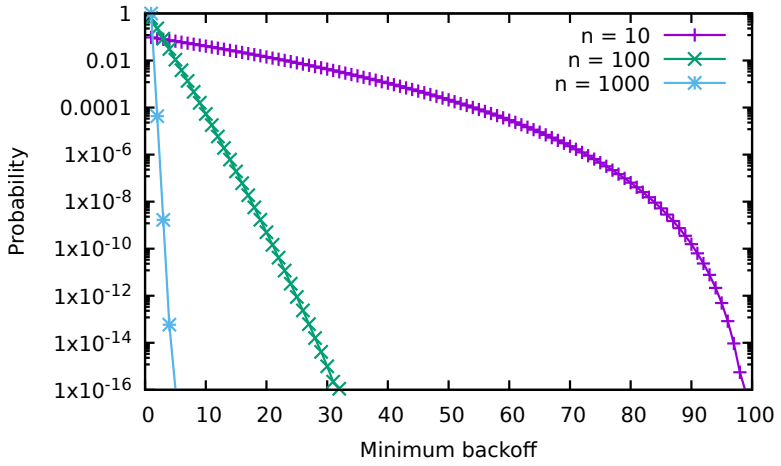


FIGURE 5.6 – Probabilité que $b_{\min}$ soit dans un intervalle de 1% de la fenêtre de backoff t.

cas est le produit des probabilités qu'a chaque valeur d'être inférieure ou supérieure à v. La probabilité p pour une valeur d'être strictement inférieure à v est de v/t, donc la probabilité d'être supérieure à v est de $1 - v/t$. C'est pourquoi la probabilité pour chacun des $\binom{n}{r}$ cas est de $p^r \times (1-p)^{n-r}$. De là, on peut enfin calculer P comme suit :

$$P = \binom{n}{r} \times (p^r \times (1-p)^{n-r}) \tag{5.8}$$

$$= \frac{n!}{r!(n-r)!} \times \left(\frac{v}{t}\right)^r \times \left(1 - \frac{v}{t}\right)^{n-r} \tag{5.9}$$

La Figure 5.7 montre l'équation 5.9 pour $t = 199\,804$ ns, $n = 1000$, des redondances r allant de 2 à 7 en fonction de v. Pour ces paramètres, il est très rare que les nœuds stockent des paquets pour plus de 800 ns. Après cette valeur, la probabilité diminue brusquement : par exemple, la probabilité pour que quatre valeurs ($r = 4$) soit plus petites que 800 ns (qui est approximativement 0.4% de t) est de 0.19, alors que la probabilité que ces valeurs soient plus petites que 1600 ns est de 0.05.

Cela signifie que pour une redondance de 4 et une fenêtre de backoff de $199\,804$ ns, dans 95% des cas le paquet est stocké en mémoire pour au plus 1600 ns, c'est-à-dire 0.8% de la taille de la fenêtre. On considère donc que le temps de stockage est négligeable.

La Figure 5.7 montre aussi que les courbes ne sont pas monotones. Pour expliquer cela, prenons un exemple avec deux valeurs ($r = 2$). Pour des petites valeurs de v (proches de 0), la probabilité d'avoir deux valeurs plus petites que v augmente en même temps que v. De même pour des grandes valeurs de v (proches de t), la probabilité d'avoir $n-2$ valeurs plus grandes que v (c'est-à-dire 2 valeurs plus petites que v) augmente avec la diminution de v. Les augmentations à partir de 0 et de t expliquent le caractère non monotone des courbes de probabilité.

5.3/ SIMULATION

Afin de tester et d'évaluer le protocole que nous proposons nous avons effectué des simulations en utilisant BitSimulator, le simulateur développé dans le cadre de ces travaux

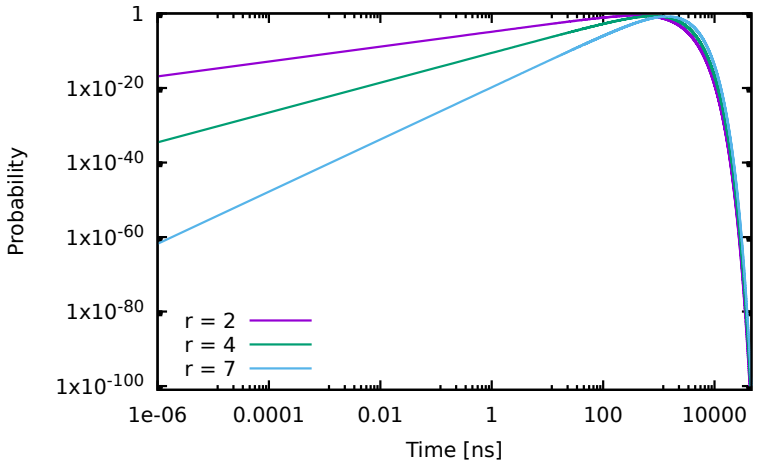


FIGURE 5.7 – Probabilité pour que r valeurs soient plus petites que la durée v dans une fenêtre de 199 804 ns.

de thèse.

5.3.1/ SCÉNARIO ET PARAMÈTRES

Dans toutes les simulations de cette étude, nous utilisons le scénario suivant. Nous simulons un environnement 2D un carré de $s = 6$ mm de coté. Les nœuds ont une portée de communication de $cr = 0.5$ mm. Les paquets contiennent 1000 bits, $\beta = 1000$, et $T_p = 100$ fs [34]. Les nœuds sont toujours placés aléatoirement et de manière uniforme sur toute la surface. Un nœud placé au centre du réseau initie la diffusion d'un paquet à tous les nœuds dans le réseau.

Étant donné la taille de l'environnement simulé et la portée de communication, le nœud le plus éloigné (de l'émetteur initial) dans le réseau se trouve dans un coin, à $\sqrt{2} \times s/2 = 4.2$ mm, c'est-à-dire $4.2/cr = 8.4$ fois la portée de communication. Il faut donc un peu plus de 8 sauts (si les forwardeurs sont placés de manière optimale) pour que le message atteigne le nœud le plus éloigné. La densité (nombre de voisins) moyenne peut être simplement calculée avec

$$density = \frac{\pi cr^2}{s^2} \times n_b \tag{5.10}$$

où n_b est le nombre total de nœuds présents dans le réseau. Ce qui, appliqué à notre scénario, donne : $density = 0.022 \times n_b$.

Dans la suite, nous comparons trois méthodes, le backoff flooding (notre méthode) en utilisant DEDeN, le flooding probabiliste en utilisant aussi DEDeN, et le flooding simple. À la place d'utiliser DEDeN, il est possible d'utiliser n'importe quel autre estimateur tant qu'il fonctionne suffisamment bien dans les nanoréseaux denses.

Comme précédemment, on définit le flooding probabiliste comme présenté dans [17], c'est-à-dire un flooding avec une probabilité de forwarder de :

$$p = \frac{r}{n_e} \tag{5.11}$$

où $r \leq n_e$ est le nombre moyen de forwardeurs souhaité (la redondance), et n_e le nombre estimé de voisins pour chacun des nœuds tel que fourni par DEDeN.

Il est important de noter que le simple flooding et le flooding probabiliste ne fonctionnent pas très bien dans les nanoréseaux pour les raisons suivantes. Quand un nœud envoie un paquet, à cause de la vitesse de la lumière c ; tous ses voisins reçoivent le paquet plus ou moins en même temps, plus précisément dans un intervalle de kT_p, avec k petit (et qui dépend de la portée de communication) et $T_p = 100\,\text{fs}$ (la durée d'un pulse, et donc d'un bit). Plus spécifiquement, $k = cr/(cT_p)$, ce qui donne dans notre scénario $k = 0.5\times10^{-3}/(3\times10^{8}\times100\times10^{-15}) = 16.5$. Que ce soit dans le flooding simple ou le probabiliste dans des réseaux denses, tous les voisins forwardent le paquet dès qu'ils le reçoivent, c'est-à-dire dans une toute petite fenêtre de $k\times T_p$, ce qui engendre un très grand nombre de collisions au niveaux des récepteurs. En effet, dans les nanoréseaux (utilisant TS-OOK) aucun mécanisme empêche les nœuds d'accéder au canal simultanément ; il n'y a pas de capture du canal. Pour éviter cet effet désastreux et être capable d'utiliser le canal avec plus d'efficacité, nous avons ajouté au flooding simple et au flooding probabiliste, un backoff tiré dans une fenêtre de de 10 000 fs.

Les métriques utilisées dans cette comparaison sont le coût et la couverture. La couverture est définie (pour cette étude) comme étant le nombre de nœuds ayant correctement reçu au moins une copie du paquet divisé par le nombre total de nœuds dans le réseau. Une couverture de 1 signifie donc que tous les nœuds ont correctement reçu le paquet. On notera qu'un réseau disjoint ne peut pas atteindre une couverture de 1. On définit le coût comme étant le nombre total de copies de paquets envoyés dans le réseau divisé par le nombre total de nœuds dans le réseau. Le but est donc d'obtenir la plus haute couverture avec un moindre coût.

Dans la suite, une redondance de r signifie que le nœud ne va forwarder un paquet que si et seulement s'il a reçu au plus r copies. Autrement dit, quand un nœud reçoit $r + 1$ copies durant le backoff, il rejette le paquet.

Dans les résultats, chaque point représente la moyenne de 15 simulations qui ne diffèrent que par le choix des graines pour la génération des nombres aléatoires. Ces dernières influent le nombre aléatoire tiré décidant du forward ou non dans le flooding probabiliste, le backoff pour le flooding probabiliste et le flooding simple, mais également pour la valeur du backoff flooding, ce qui influence grandement la sélection des différents forwardeurs. Nous considérons que ces graines génères des scénario suffisamment différents, étant donné le grand nombre de nœuds et la taille du réseau au regard de la portée de communication.

5.3.2/ RÉSULTATS

Nous simulons 10 000 nœuds, ce qui au regard de l'équation 5.10 donne une densité moyenne de 218. On peut donc supposer sans trop de risque que tous les nœuds sont atteignables par l'émetteur initial (au travers de plusieurs sauts). Les résultats sont montrés dans la Figure 5.8.

Étant donné que le backoff flooding assure que au moins un certain nombre de paquets sont envoyés dans chaque voisinage, une redondance de 1 devrait être suffisante pour atteindre une couverture de 1 ; les simulations confirment cette hypothèse : avec une redondance variant entre 1 et 20, la couverture est toujours de 1, c'est-à-dire que tous

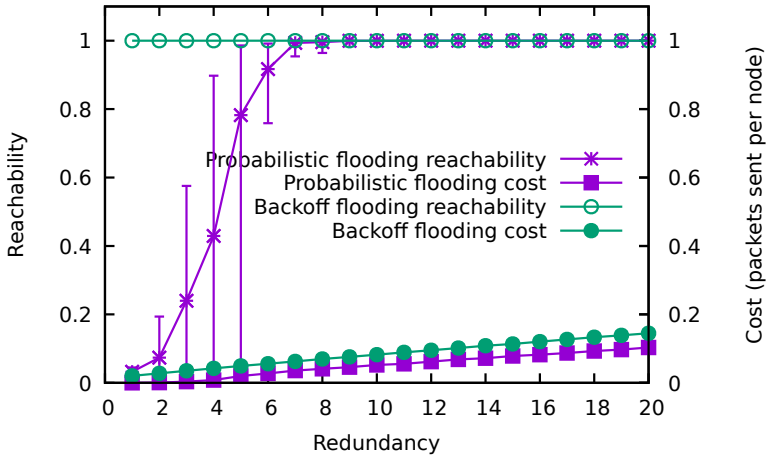


FIGURE 5.8 – Couverture et coût du flooding probabiliste et du backoff flooding dans des réseaux denses. Le flooding simple, non montré dans la figure, à une couverte et un coût de 1.

les nœuds reçoivent le paquet correctement.

Pour le flooding probabiliste, la couverture dépend grandement de la redondance, pour les redondances faibles la couverture est assez basse. Aussi, pour les redondances faibles, une grande variabilité affecte les résultats : la couverture varie et peut atteindre des valeurs proches de 0 et des valeurs proche des 1 (pour la redondance de 5 dans la Figure 5.8 par exemple). Une couverture proche de 0 dans une simulation correspond (en général) au fait que le paquet initial, pour des raisons de probabilité, n'a été forwardé par aucun nœud, et donc seuls les voisins directs du transmetteur initial l'ont reçu (mais ne l'ont pas forwardé). En général, le die out problem (une couverture inférieure à 1) survient quand, pour des raisons liées au caractère aléatoire de l'algorithme, aucun nœud d'une certaine partie du réseau ne reçoit le paquet, ce qui implique que les régions plus éloignées ne le recevront pas non plus. Ce phénomène ne survient pas avec le backoff flooding, car dans ce protocole les probabilités sont utilisées pour sélectionner un forwardeur, et non pour savoir si le paquet est forwardé ou pas.

Un autre avantage en faveur du backoff flooding est que, étant donné qu'une redondance de 1 est suffisante, on peut dire qu'il est "automatique", c'est-à-dire qu'il n'y a pas de paramètre à régler, là où dans le flooding probabiliste il faut chercher la redondance à appliquer pour obtenir une couverture de 1 tout en économisant un maximum de ressources.

En ce qui concerne le coût du flooding probabiliste, le nombre de paquets envoyés est proportionnel à la probabilité de transmission, ce qui, d'après la formule 5.11, est proportionel à la redondance. Pour le backoff flooding également, le redondance donne le nombre de paquets dans un voisinage, le coût est donc aussi proportionnel à la redondance. Les simulations confirment ces résultats : la Figure 5.8 montre que le nombre de paquets augmente linéairement avec la redondance pour les deux méthodes. Cependant, pour une couverture de 1, le coût du backoff flooding (obtenu avec une redondance de 1) est plus faible que le flooding probabiliste (qui nécessite une redondance de 9).

Quant au flooding simple, il obtient toujours le même résultat : une couverture de 1 avec un coût de 1. Le backoff (de 10 000 fs) permet de réduire le nombre de collisions et donc

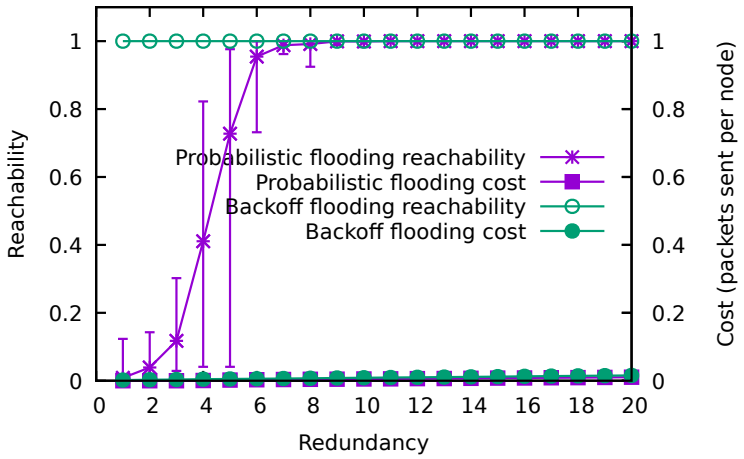


FIGURE 5.9 – Couverture et coût du flooding probabiliste et du backoff flooding dans des résaux très denses. Le flooding simple, non montré dans la figure, à une couverte et un cout de 1.

tous les nœuds reçoivent le paquet au moins une fois, on obtient donc une couverture de 1. La raison pour laquelle le coût est systématiquement de 1 réside dans la nature même du flooding simple : chaque nœud recevant le paquet pour la première fois le forwarde à son tour. En comparaison, le backoff flooding fournit également une couverture de 1, mais provoque l'envoi de bien moins de paquets.

Nous avons également simulé un réseau contenant 100 000 nœuds, soit une densité 10 fois plus grande, c'est-à-dire environ 2180 voisins. Les conclusions concernant la couverture et le coût sont similaires, comme le montre la Figure 5.9. De plus, on remarque que le nombre de paquets envoyés par nœud est similaire au scénario avec 10 000 nœuds, même si la densité du réseau diffère d'un ordre de grandeur.

Le flooding probabiliste et le backoff flooding réduisent tous les deux énormément le nombre de copies générées en comparaison au flooding simple. Cependant, le backoff flooding présente deux avantages par rapport au flooding probabiliste, mais aussi deux inconvénients. Premièrement, il est bien moins susceptible de ne pas couvrir entièrement le réseau, car l'envoi ou non d'un paquet n'est pas probabiliste, et si une redondance est requise le backoff flooding assure cette redondance. Deuxièmement, en utilisant le paramètre k, la variation dans la redondance effectivement perçue par les nœuds peut être réduite. De l'autre coté, il implique un délai qui augmente avec la réduction du nombre de forwardeurs et leur répartition. Étant donné que la décision de transmettre ou de supprimer un paquet n'est pas prise dès la réception de celui-ci, le backoff flooding requiert de garder une copie du paquet en mémoire durant ce temps. Ce délai est cependant assez court, voir la section 5.2.3.

Même si notre méthode n'est pas conçue pour les réseaux peu denses (où le flooding simple peut avoir des résultats satisfaisants), il est intéressant d'observer la couverture que fournit le backoff flooding dans ce cas-là.

On simule un nombre total de nœuds allant de 150 à 500, ce qui, au regard de l'équation 5.10, représente une densité moyenne (nombre de voisins) entre 3 et 11.

On peut observer dans la Figure 5.10 que la couverture est encore de 1 ou proche de 1 pour les densités supérieures à 9.6 avec une redondance de 1. Pour les densités allant

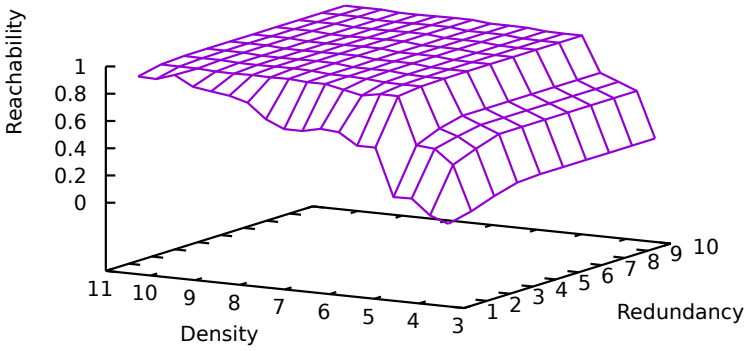


FIGURE 5.10 – Couverture dans des résaux à faible densité.

de 9.2 à 5.1, une redondance de 2 est nécessaire pour obtenir une couverture proche de 1. Pour des densités encore plus basses, inférieures à 5.1 une redondance de 1 donne de mauvais résultats et une densité de 10 donne également des résultats inférieurs à 1, pour une densité de 3.76 par exemple, les couvertures sont de 0.15 et de 0.6, respectivement.

La raison derrière cette couverture incomplète s'explique par une probable disjonction du réseau. Pour une densité de 3.7, avec une redondance de 10, ce qui est plus grand que le nombre moyen de voisins, la couverture est encore de seulement 0.6, ce qui correspond au nombre maximum de nœuds qui peuvent être joints par le transmetteur initial, sachant que le réseau est très éparse.

Pour conclure, dans les réseaux avec une densité basse, la couverture maximale ne peut être atteinte qu'en augmentant la redondance, et pour les densités très faibles tous les nœuds ne peuvent pas recevoir le paquet, car les chances que le réseau soit disjoint sont très élevées.

5.4/ CONCLUSION

Les méthode de diffusion actuelle ne fonctionnent pas dans les nanoréseaux. Elles demandent trop de ressources (mémoire, capacité de calcul, etc.) ou ne peuvent pas s'adapter aux spécificités des nanoréseaux.

Le backoff flooding combine une fenêtre de backoff, un compteur de paquets et nécessite un estimateur du nombre de voisins. Nous avons expliqué en quoi cette méthode est efficace dans des réseaux denses, et nous avons conduit des simulations pour confirmer ces hypothèses. Aussi, en comparaison avec le flooding probabiliste, la méthode proposée est automatique, c'est-à-dire qu'il n'y a besoin d'aucun paramétrage à priori, et notre méthode nécessite moins de paquets échangés sur le réseau. Par exemple, dans un réseau avec une densité de 218 voisins, le backoff flooding envoie 71% de paquets en moins. Il évite le problème de die out et le broadcast storm tout en restant efficace en termes de couverture et de délai. On peut cependant noter que en théorie le backoff flooding peut également souffrir du problème du die out. Cela peut se produire si tous les réémetteurs sont très très proches de l'émetteur initial. Dans ce cas, les nouvelles

réémissions peut ne toucher aucun nœud n'ayant pas déjà reçu le paquet initial. Cette situation aurait pour effet d'effacer toutes les copies stockées dans tous les nœuds en attente sans que le message ne progresse. Cette situation est extrêmement rare, et plus la redondance est élevée moins cette situation est probable. Elle n'a jamais été observée dans nos simulations, c'est pourquoi cette notion n'a pas été développé dans ce chapitre

Le survey [57] présente différents protocoles de routage spécialement conçus pour les nanoréseaux électromagnétiques. Dans ce survey il est question de "serious redundancy and conflicts" quand on parle de solution basée sur du flooding. Ces problèmes de redondance que l'on trouve dans les floodings naïfs n'apparaissent pas avec le backoff flooding puisqu'il limite drastiquement le nombre de paquets envoyés lors d'un flooding. De plus le backoff flooding est également compatible avec d'autres protocoles (de routage par exemple) pour les optimiser. Un exemple d'utilisation du backoff flooding couplé à un autre protocole est montré en Section 6.5.

6

ROUTAGE CAPABLE DE DÉVIATION

La congestion est un problème bien connue dans le monde des réseaux. Les nanoréseaux ne font pas exception et peuvent aussi souffrir de congestion. Cependant cette congestion apparaît pour des raison différentes de celles rencontrées dans les réseaux habituels et peut être adressée par des solutions novatrice. Ce chapitre présente deux contributions distinctes : une méthode spécifique pour détecter la congestion dans les nanoréseaux THz, et une méthode pour dévier la route SLR si cette dernière rencontre de la congestion.

Notre algorithme de déviation s'inspire de la déviation routière. L'idée est de profiter de la topologie des nanoréseaux qui offre de nombreuses routes alternatives pour rallier deux points du réseaux. En effet, les nanoréseaux pouvant être très denses, il existe très souvent de nombreuses routes pour atteindre un point donnée dans le réseau.

Les simulations utilisant BitSimulator qui sont présentées dans ce chapitre ont été réalisées avec la dernière version au moment de la rédaction de ce manuscrit ; c'est-à-dire la version 0.9.4. Les contributions présentées de se chapitre reposent principalement sur l'article [5].

6.1/ INTRODUCTION

La congestion est un état du réseau dans lequel le nombre de paquets est suffisamment élevé pour créer un, voire plusieurs blocages, empêchant ainsi tout ou une partie du système de remplir ses objectifs. On peut également définir la congestion comme étant l'état d'un système dans lequel le débit d'entrée est supérieur au débit de sortie pendant un temps suffisamment grand pour saturer les différents tampons de ce système (si de tels tampons existent) ; on appelle en général ces tampons des buffers. Deux exemples bien connus de systèmes pouvant être sujets à de la congestion sont le trafic routier (les bouchons) et le trafic aérien (aéroports surpeuplés). Dans ces cas là, les solutions habituellement envisagées sont le détournement de trafic (ou déviation) ou la mise en attente (jusqu'à ce que des ressources, terminaux d'atterrissage par exemple, soient de nouveau disponibles).

De même, la congestion dans les réseaux décrit un état où trop de paquets sont échangés, provoquant ainsi la perte de certains d'entre eux [35], voire tous dans les pires cas. On dit qu'un lien (entre deux équipements) subit de la congestion quand une quantité de trafic supérieure à la capacité du canal est envoyé dessus. Dans ce cas, l'équipement réseau situé au début de ce lien absorbe la différence de charge grâce à

des buffers mémoire.

Par exemple, A envoie des données vers B sur le lien AB. Le lien AB à une capacité de 1 paquet par unité de temps, et la taille du buffer sur A est de 2 paquets. Si A transmet des données avec un débit de 1 paquet par unité de temps (ou moins) tout se passe bien, et cette situation peut durer indéfiniment. En revanche, si A transmet à un débit de 2 paquets par unité de temps alors, à chaque unité de temps, le buffer de A se rempli de 1 paquet. Cette situation ne peut durer que 2 unités de temps sans qu'il n'y ait de paquet perdu.

Sur un lien partagé par plus de 2 équipements, les pertes de paquets peuvent être amplifiées par les collisions. Ce cas se présente, par exemple, en Wi-Fi (802.11) lorsque beaucoup d'émetteurs souhaitent transmettre beaucoup de données. Idéalement, les paquets de plusieurs flux concurrents sont envoyés les uns après les autres sur le canal avec peu de temps d'attente "entre chaque" paquet. C'est le mécanisme de backoff qui est en charge de cette répartition, cependant des paquets peuvent tout de même être perdus à cause de collisions (en particulier quand le nombre d'accès concurrent au canal est élevé). Sur un canal à contention comme le mode DCF de 802.11, en cas de congestion, le nombre de collisions augmente. Le mécanisme de détection des paquets endommagés détecte cette situation et pallie le problème en procédant à des retransmissions. Mais ce faisant il provoque une utilisation inefficace du canal qui conduit à un engorgement aggravé des buffers sur les liens.

Les paquets sont donc en général perdus une fois que les buffers sont remplis. Il existe cependant différentes stratégies de choix de paquets à effacer. Dans les protocoles les plus simples, ce sont les nouveaux paquets arrivant sur un équipement dont les buffers sont déjà pleins qui sont effacés ; on dit en général que le paquet est droppé (jeté). Mais d'autres stratégies sont envisageables. Par exemple, il est possible de choisir aléatoirement des paquets déjà en attente avant que la congestion ne survienne ; c'est ce que fait la politique RED (random early detection) [15] et ses variantes.

Les mécanismes pour éviter et réparer ce type de perte sont regroupés sous le terme contrôle de congestion.

6.2/ CONGESTION ET TRAVAUX CONNEXES

Dans le réseau macro internet, le contrôle de congestion se fait de bout-en-bout, c'est-à-dire seulement sur les terminaux finaux. Quand un paquet est perdu par le très largement utilisé protocole TCP [43], l'émetteur considère qu'un lien sur le chemin subit de la congestion et ralentit son trafic en conséquence, afin de participer à la désaturation du réseau et avoir ainsi moins de chances de perdre les paquets suivants.

La littérature concernant le contrôle de congestion est particulièrement abondante, et de nouvelles variantes de TCP sont régulièrement proposées. Habituellement, elles prennent en compte les pertes et le délai (RTT, et le "rythme" des paquets ; c'est-à-dire le moment où les acquittements arrivent à l'émetteur), et changent les paramètres utilisés pour les augmentations et les diminutions de débit. D'autres types de mécanismes, comme celui basé sur des équations [23], sont aussi des mécanismes de bout-en-bout.

TCP multi-path [24] est une version de TCP rétro compatible avec les autres versions. Ce nouvel algorithme propose de tirer partie de la nature multi-chemins de l'internet, mais

aussi de la pluralité des accès au réseau sur un même terminal (données téléphoniques mobiles et Wi-Fi sur un téléphone portable par exemple) pour faire une meilleure exploitation des ressources disponibles dans le réseau. Ce protocole sépare un flux de données en deux sous-flux qu'il envoie par deux routes différentes. Le contrôle de flux et le contrôle de congestion se font ensuite normalement sur chacun des sous-flux de manière indépendante. La déviation que nous proposons ne remplit pas le même rôle. Nous cherchons à éviter que les paquets ne soient perdus à cause de la congestion, alors que les contrôle de congestion de TCP réagit aux pertes de paquets. De plus notre déviation se fait par les routeurs eux même alors que TCP n'opère que de bout en bout. Notre algorithme peux donc être plus réactif.

Une approche un peu différente a été proposée via l'extension ECN du protocole TCP/IP [41], dans lequel les routeurs sont également impliqués dans la gestion de la congestion. Cela dit, leur implication reste toutefois assez limitée. Les routeurs marquent les paquets lorsqu'un danger de congestion potentielle est détecté. La décision quant à la politique à adopter est ensuite prise de bout-en-bout ; ce qui finira probablement par réduire le débit de l'émetteur.

De plus, dans les réseaux sans fils, le taux d'erreur sur le canal peut être élevé (en comparaison aux réseaux filaires) et les pertes de paquets sont interprétées (par TCP par exemple) comme étant de la congestion alors que ce n'est pas le cas.

Il est utile de noter que l'évitement et la "réparation" de la congestion augmentent le temps total de transmission. Cela est dû au fait que le seul levier sur lequel peut agir l'émetteur est son débit de transmission qu'il peut choisir de réduire (ou d'en modifier le profil). Et même pire, les effets du contrôle de congestion n'apparaissent que tardivement, après 1 RTT (round trip time) : du routeur qui détecte la congestion, jusqu'à la destination puis de retour à la source qui peut réduire son débit d'émission et enfin dégorger le routeur qui a détecté de la congestion.

Récemment, avec l'engouement croissant pour divers types de réseaux, comme les réseaux mobiles ad hoc (MANET) et les réseaux de capteurs sans fils, de nouvelles techniques de contrôle de congestion ont été proposées.

Les paramètres utilisés pour détecter et éviter la congestion dans les MANETs sont spécifiques à la couche liaison et ne sont pas applicables aux nanoréseaux. Par exemple, un mécanisme de partage de la charge basé sur le routage est proposé dans l'article [3]. La congestion est détectée en utilisant trois paramètres : la bande passante disponible (basée sur la proportion du temps où le canal n'est pas utilisé), une estimation de la charge (basée sur la fenêtre de contention du Wi-Fi (802.11)), et l'énergie résiduelle des nœuds. En plus de la détection de la congestion, la déviation des paquets se fait avec la découverte de routes. Même si tout comme notre méthode, c'est le nœud qui détecte la congestion (c'est-à-dire le routeur lui-même) et qui agit en conséquence, aucune des méthodes utilisées dans les MANETs ne s'applique aux nanoréseaux. Par exemple :

- La bande passante disponible n'est généralement pas un problème dans les nanoréseaux, alors que les buffers mémoire le sont.
- Il n'y a pas de retransmission façon Wi-Fi.
- L'énergie consommée pour les transmissions est décrite par des formules différentes, envoyer un bit 0 ne consomme pas (ou preque pas) d'énergie, autrement dit l'énergie consommée pour l'envoi d'un paquet est très liée au codage utilisé.
- Il n'y a pas de découverte de route, la route doit s'adapter dynamiquement en

fonction de la situation.
L'équilibrage de charge dans LARA [45] utilise le degré de contention au niveau MAC. Les nœuds doivent (entre autres) garder en mémoire les dernières estimations des buffers mémoire de chacun de leurs voisins. Les nanonœuds ne possèdent pas assez de mémoire pour stoker autant d'information, en particulier dans des scénarios dans un voisinage très dense.

Le routage multi-chemins dans les WSN [4] n'est pas non plus applicable dans le cadre des nanoréseaux. Encore une fois, soit les nœuds ont besoin d'informations précises à propos de leurs voisins, soit les techniques de détection de congestion ne sont pas adaptées aux nanoréseaux

dépendance :

DEDeN, Backoff flooding, SLR

6.3/ LA CONGESTION DANS LES NANORÉSEAUX

Les propriétés électromagnétiques des nanoréseaux sont différentes des réseaux sans fils macro et affectent le fonctionnement des protocoles. L'utilisation de fréquences terahertz permet des bandes passantes et des débits plus élevés, allant jusqu'à plusieurs Tb/s [38]. Les transmissions basées sur des pulses, comme c'est le cas dans TS-OOK, permettent l'entrelacement de bits de plusieurs paquets, et donc dissipent la nature séquentielle de l'accès au canal et de la réception de paquets. Enfin, les ressources limitées des nanoréseaux, en particulier la mémoire et la puissance de calcul, diminuent la capacité des nœuds à traiter les paquets entrants. Comme expliquée plus bas, la congestion dans les nanoréseaux ne survient pas à cause d'un canal saturé, mais plutôt à cause des faibles capacités de chacun des nœuds. Cette hypothèse a été confirmée par Tucker Wilson, étudiant de l'Uniervisté de Virginie aux États-Unis, lors de son stage dans notre équipe de recherche.

La transmission par pulses permet à plusieurs paquets d'être entrelacés, et les nœuds peuvent donc les recevoir en parallèle. Le nombre de transmissions en parallèle qu'il est possible d'avoir sans qu'il n'y ait de collisions dépend uniquement de l'intervalle de temps entre deux bits consécutifs d'un paquet. Le canal lui-même ne pose pas de limite. Ce type d'entrelacement de paquets est très différent des schémas TDMA standards, car dans ces derniers ce sont uniquement les flux qui sont entrelacés. En TMDA, à chaque instant, un seul paquet à la fois est en cours de transmission. De plus, un nœud voulant transmettre une grande quantité de données pourrait théoriquement utiliser tous les slots et donc utiliser toute la bande passante du canal, ce qui n'est pas possible dans les nanoréseaux. Dans un réseau standard, un nœud peut également choisir de recevoir toutes les transmissions provenant de ses voisins directs.

Les nœuds dans les réseaux utilisant des communications par pulses peuvent recevoir des paquets entrelacés, mais il y a forcément une limite au nombre de paquets qu'ils peuvent stocker et traiter en parallèle. À la différence de la plupart des réseaux classiques, la capacité du canal est plus élevée que la capacité de transmission d'un nœud seul. Il y a une forte asymétrie concernant la capacité des nœuds à transmettre et celle à recevoir de l'information. Un nœud n'est peu-être pas capable de transmettre rapidement : même s'il est actif tout le temps, il ne pourrait pas utiliser plus d'une fraction de

la capacité du canal. Cependant, il peut avoir besoin de ”recevoir rapidement”, si une grande quantité de données issues de transmission simultanée est reçue.

Néanmoins, les nanonœuds ne peuvent sûrement pas recevoir, stocker et traiter des Tb/s de données :

— Les limitations matérielles peuvent ne pas permettre de recevoir un grand nombre de paquets en parallèle.
— Les limitations mémoires ou la faible puissance de calcul empêchent les nœuds de stocker ou de traiter les paquets aussi vite qu'ils arrivent. Un nœud doit (malheureusement) traiter des paquets qui ne sont pas pertinents pour lui : il faut bien décoder (au moins en partie) le paquet avant de pouvoir prendre une décision de routage ou d'acceptation.
— Le peu d'énergie disponible ne permet pas d'opération prolongée, en particulier si l'on tient compte de la taille minuscule des batteries et de la vitesse de recharge potentiellement lente (et dépendante de l'environnement).

Ces limites sont connues par les nœuds. Surveiller à quel point un nœud est proche de ses limites se traduit directement par une estimation de la congestion de ce même nœud. Les conditions subies par un nœud, en particulier le nombre de paquets en train d'être reçus, sont susceptibles d'être sensiblement les mêmes sur les nœuds voisins. Ceci est inhérent à la nature sans fil des communications dans les nanoréseaux. Cette information est utile pour détecter une potentielle congestion sur les nœuds voisins.

6.4/ DÉTECTION DE LA CONGESTION

La congestion dans les nanoréseaux survient en majeure partie, comme vue dans la section 6.3, à cause de la saturation des buffers de réceptions. On considère que chacun des buffers de réceptions ne peux accueillir qu'un seul paquet à la fois. Le nombre de buffers de réceptions détermine donc le nombre de paquets que peut recevoir simultanément un nœud. On suppose que chaque nanonœud connaît le nombre de ses de buffers effectivement utilisés à un instant donné. Par exemple, un nœud peut ajouter un flag d'un seul bit à chacun de ses buffers de réception ; ce flag est levé dans le nœud commence à recevoir un paquet et est baissé à la fin de la réception. On peut noter que, contrairement aux méthodes de détection de congestion opérant de bout en bout, cette méthode n'interprète pas les pertes dues aux conditions de canal comme de la congestion.

Un nœud n peut estimer son niveau de congestion au temps t comme suit :

$$c_n(t) = \frac{r_n(t)}{r_{\max}}$$

Ici, $r_n(t)$ est le nombre de buffers de réception du nœud n utilisés au temps t (le nombre de paquets étant en train d'être reçus au temps t), et $r_{\max}$ le nombre total de buffers de réception au niveau d'un nœud.

Le quotient c_n qui en résulte est interprété comme un niveau de congestion : le nœud peut détecter une congestion partielle avant qu'une congestion complète ne survienne et ne bloque le trafic. Le quotient de congestion peut également être interprété comme une propriété spatiale : les nœuds proches d'une zone très active du réseau vont subir un haut niveau de congestion. Ce dernier décline lorsqu'on s'éloigne de cette zone très active.

On peut classifier l'état de congestion autour d'un nœud en utilisant plusieurs quotients seuils. Si le niveau de congestion dépasse le seuil c_U (U pour upper), le réseau peut à tout moment être complètement congestionné, et des mesures préventives devraient être prises. Une congestion de niveau c_L (L pour lower) ou moins indique au contraire que la zone ne présente que très peu de risque de congestion. Dans ce cas-là, un nœud peut forwarder le message normalement, voire même passer sur un schéma de forward optimisé.

À noter qu'un nombre très faible de buffers de réception $r_{\max}$ peut empêcher la détection de la congestion, si c_U est supérieur à $(r_{\max} - 1)/r_{\max}$. Dans ce cas, un nœud ne sera capable de détecter la congestion qu'une fois que le réseau autour de lui est déjà en congestion. Cependant, même s'il envoie un message, les autres nœuds sont très peu susceptibles de recevoir des messages supplémentaires.

Un $r_{\max}$ plus élevé permet une évaluation de la congestion via un gradient et non plus par un simple état binaire (oui/non). Les algorithmes qui en découlent pourraient ensuite utiliser ce gradient pour adapter leur comportement, en modifiant des délais de transmission par exemple, ou encore en déviant plus ou moins les paquets de leurs chemins initiaux.

6.5/ FUSION DE SLR ET DE BACKOFF FLOODING

Dans la version originale de SLR, tous les nœuds des zones sur les chemins forwardent le paquet. Dans les zones très denses, cela peut créer de la congestion, même pour la transmission d'un seul paquet initial. Pour éviter cette situation, nous combinons SLR avec le backoff flooding. Ainsi, un nombre réduit de nœuds forwardent le paquet dans chaque zone. La section 6.3 insiste sur le fait que les faibles capacités des nœuds sont, en majeure partie, la cause de l'apparition de congestion : surveiller le degré de saturation des nœuds est une des clefs de voûte de notre solution. Pour ce faire nous proposons d'améliorer le protocole SLR.

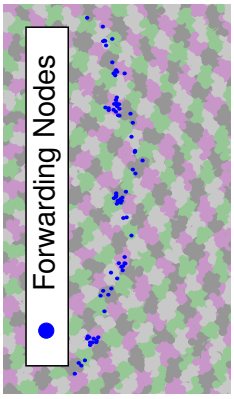
Il est possible de réduire le nombre de forwards en appliquant le schéma de diffusion backoff flooding (cf. Chapitre 5) au protocole SLR. Nous utilisons donc une version combinée des deux protocoles, où SLR calcule la route d'un paquet et le backoff flooding réduit le nombre de forwards nécessaires pour que le paquet arrive à destination.

Pour rappel, SLR (cf. Section 2.2) est un protocole d'adressage et de routage conçu pour les nanoréseaux. Il est capable de créer un système de coordonnées en n'utilisant que 2 ancres (pour un réseau en 2D) qui adressent des zones du réseau, puis de router des messages quelque soit la zone source et la zone destination.

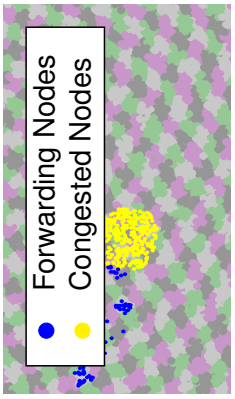
SLR peut être considéré comme une forme de flooding limité géographiquement : seuls les nœuds des zones se trouvant sur le chemin entre la source et la destination transmettent le message. Le backoff flooding, en tant que méthode qui améliore le flooding, est donc compatible avec SLR.

Cela aide déjà à réduire la congestion, étant donné que les ressources du réseau sont mieux utilisées (de manière moins agressive). De plus, il a un impact positif sur la consommation d'énergie. Enfin, il permet aux nœuds de ne pas saturer leur capacité de réception avec un seul message.

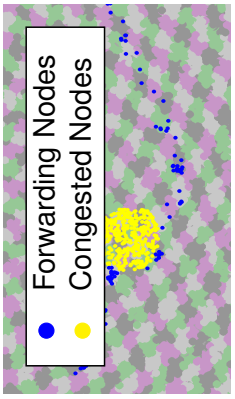
Cependant, l'inclusion du backoff dans SLR génère un nouveau problème qui concerne la taille des zones. Il est possible que les nœuds ne soient pas capables de compter



(a) Pas de congestion : Les messages sont forwardés sur le chemin SLR.



(b) En cas de congestion avec SLR : Le message ne peux pas traverser la congestion.



(c) Congestion avec la déviation : Le routage est dévié quand il rencontre la congestion.

FIGURE 6.1 – Comparaison des schémas de routage en présence de congestion. Les messages se progagent de bas en haut.

tous les paquets envoyés par une zone voisine, à cause de la façon dont les zones sont construites. Ce phénomène mène à des transmissions non nécessaires. C'est pourquoi nous avons modifié la phase d'adressage de SLR. Les nœuds n'utilisent pas une partie de leur puissance d'émission sur la phase d'adressage, ce qui génère de plus petites zones et facilite donc le comptage des paquets pour le backoff flooding.

On appelle ce nouveau protocole SLR modifié.

Une fonctionnalité intéressante qui n'a pas encore été présentée est la capacité de SLR à créer des chemins de différentes largeurs. Le protocole spécifie un paramètre pathWidth qui configure la largeur du chemin SLR en nombre de zones. Par défaut, le paramètre pathWidth est égal à 1 ; les nœuds ne forwardent le paquet que s'ils se trouvent dans une zone se trouvant sur le chemin le plus direct. Augmenter le paramètre pathWidth élargit le chemin, et étale les forwardeurs sur plus de zones. Ce paramètre est appelé m dans l'article original [55], et nous ferons de même dans ce chapitre.

6.6/ LA DÉVIATION DE PAQUETS

Lorsqu'un nœud détecte une congestion trop élevée, il réagit et tente de dévier le paquet qu'il reçoit. Il utilise toujours le chemin donné par SLR comme base, mais modifie la largeur du chemin : si la largeur du chemin augmente, le message se répand pour occuper de zones supplémentaires.

La Figure 6.1 permet de visualiser plusieurs cas. Chaque sous-figure montre en arrière-plan les zones SLR qui se sont répandues par vague. Ces zones sont le résultat de la phase de beaconning de SLR, où chaque nœud détermine sa distance en nombre de sauts par rapport à deux ancres. Par-dessus les zones, les nœuds ayant forwardé le message sont dessinés en bleu foncé, et les nœuds ayant atteint leur r_{max} sont en jaune clair.

La Figure 6.1a donne un exemple de routage sans congestion dans le réseau. Le proto-

cole SLR [55] route les paquets de bas en haut en suivant un chemin “linéaire” qui part des coordonnées source vers la zone aux coordonnées destination. $pathWidth$ est de 1, donc le chemin est large d'une seule zone.

Les nœuds décident de forwarder un paquet (ou non) en utilisant la formule `SLRPath()`. À chaque réception correcte d'un paquet, le nœud vérifie s'il se trouve sur le chemin et donc s'il doit forwarder le paquet :

```
SLRPath(nœud n, adresse src, adresse dst, int m):
- SI n est sur le chemin SLR de largeur m entre src et dst:
  - forwarder le paquet
- SINON:
  - ne rien faire
```

On peut voir que dans chaque zone, seulement quelques nœuds forwardent le paquet. Cela est le résultat du backoff flooding tel que décrit dans la section 5.2. Après que les nœuds aient reçu un certain nombre de copies, la redondance requise est atteinte et les nœuds ne forwardent pas leur copie du message.

Dans la Figure 6.1, la zone en jaune subit de la congestion, et les nœuds dans cette zone ne peuvent plus recevoir de paquets supplémentaires. La Figure 6.1b, montre les nœuds utilisant la version modifiée de SLR (avec le backoff flooding) mais sans tenir compte de la congestion. Par conséquent, le paquet ne peut pas traverser la zone subissant de la congestion, et le paquet est perdu (il n'arrive jamais à destination).

Comme montré dans le section 6.4, nous proposons d'utiliser le niveau de congestion donné par c_n comme indicateur de congestion. Dès que $c_n(t) > c_U$ les nœuds commencent à dévier les paquets en les écartant de la route initialement calculée par SLR, tout en continuant de les rapprocher de la zone destination. Pour ce faire, l'en-tête du paquet stocke une valeur supplémentaire, spécifique à la déviation. Initialement, cette valeur est mise à 1 et reste ainsi tant que les nœuds ne détectent aucune congestion. En cas de congestion, avant de forwarder le paquet, le nœud incrémente cette valeur. Les nœuds suivants utilisent cette valeur de déviation pour remplacer la valeur de m, ce qui modifie et “étale” la route du paquet (le chemin devient plus large).

Cependant, on ne souhaite pas que l'algorithme de déviation occupe plus de zones, car cela aurait pour conséquence d'aggraver la congestion en gênant d'éventuels paquets aux alentours. Ce qui est souhaitable, c'est l'utilisation de zones différentes pour faire progresser le paquet. On modifie alors la fonction de décision en la remplaçant par la fonction :

```
SLRPathDeviation(n, src, dst, m) =
SLRPath(n,src,dst,m) ET (NON SLRPath(n,src,dst,m-1))
```

La fonction `SLRPathDeviation()` “déplace” le paquet tout au bord du chemin de largeur m. Tant que les nœuds détectent de la congestion, ils augmentent la valeur de déviation du paquet, ce qui a pour effet de “pousser” le paquet de plus en plus loin de la zone de congestion. Notre protocole deviation SLR comprend à la fois la détection de congestion et la déviation du chemin SLR. Dans la Figure 6.1c, deviation SLR détecte correctement la congestion et commence à dévier le paquet.

Au bout d'un moment, le paquet va (probablement) atteindre des nœuds où $c_n(t) < c_L$, c'est-à-dire où la congestion est détectée comme n'étant plus dangereuse. Les nœuds

inscrivent alors leurs propres coordonnées SLR dans le paquet comme étant une nouvelle source intermédiaire `src'`, qui se comporte tout comme `src` pour la suite du routage, et remettent la valeur de déviation à 0. Cela provoque un nouveau routage en ligne droite depuis cette source intermédiaire jusqu'à la destination `dst`. L'effet de cette politique peut être observé dans la Figure 6.1c, où une fois la congestion dépassée, la déviation cesse et le message continue son chemin en ligne "droite" jusqu'à sa destination.

6.7/ ÉVALUATION

Le comportement de l'algorithme dépend de l'interaction entre un grand nombre de nœuds, comme c'est typiquement le cas dans les nanoréseaux : Les interactions dans les paquets entrelacés et les limites du nombre de paquets pouvant être reçus simultanément peuvent conduire à des comportements émergeants qui ne sont pas faciles à prévoir à partir de l'algorithme lui-même. De plus, nous avons besoin d'un grand nombre de nœuds dans le réseau afin de construire un réseau où au moins une route alternative existe. Toutes ses raisons justifient l'usage d'un simulateur plutôt que l'usage d'outils analytiques qui pourraient ne pas capturer certains effets émergeants que nous ne maîtrisons pas forcément. Les indications techniques et les informations liées à reproductibilité de ces résultats sont présentées sur la page web du simulateur [1].

Nous avons évalué notre protocole en utilisant notre simulateur, BitSimulator. Nous avons implémenté dans celui-ci l'algorithme de détection de congestion et l'algorithme d'évitement. Il est également facile de modifier le nombre de buffers de réceptions des nœuds.

Nous avons évalué la détection de congestion et l'algorithme de déviation en simulant un nanoréseau dense en 2D. Les réseaux en 2D ne permettent que 2 directions dans lesquelles dévier, ce qui ne permet pas des scénarios très complexes. La plupart des nanoréseaux seront probablement en 3D où la déviation bénéficiera de bien plus de possibilités. (Ce que nous présentons ici est un ensemble de travaux préliminaires, une preuve de concept. Mais des travaux plus approfondis sont encore à mener)

La Table 6.1 donne la liste des paramètres de nos simulations. $r_{\max}$ est de 5 pour tous les nœuds, ce qui signifie que chaque nœud ne peut pas recevoir plus de 5 paquets en parallèle. c_L et c_U sont tous les deux configurés à 0.5. Les nœuds augmentent donc la valeur de déviation au moment où ils reçoivent un troisième paquet ($5/2 = 2.5$) en parallèle.

Le scénario évalué envoie un paquet unique à travers le réseau, en utilisant soit le SLR modifié (avec le backoff flooding) ou avec le SLR modifié amélioré avec la détection de congestion et la déviation, la solution proposée qu'on appelle deviating SLR. Dans le premier cas, le réseau ne présente aucun autre trafic, c'est le cas témoin. Dans le second cas, une zone se trouvant sur le chemin SLR du premier paquet subit de la congestion, c'est le cas problématique. Chacun de ces cas a été simulé 10 fois, et les résultats ont été moyennés en utilisant toutes les simulations. Chacune de ces simulations utilise des graines différentes pour la génération aléatoire des périodes d'attente du backoff flodding et pour l'estimation de la densité, tous les autres paramètres sont identiques, le placement aléatoire des nœuds par exemple.

1. BitSimulator Web page : http://eugen.dedu.free.fr/bitsimulator

TABLE 6.1 – Paramètres de simulation

Taille de la zone simulée	6 mm $*$ 6 mm
Nombre de nœuds	20 000
Portée de communication	350 µm
Nombre moyen de voisins	203
β (ratio d'étalement de TS-OOK)	1000
Taille des paquet	100 bit
Redondance du backoff flooding	1
$r_{\max}$	5
c_L/ c_U	0.5 / 0.5

TABLE 6.2 – Évaluation des résultats des deux algorithmes de routage.

	Nombre moyen de paquets envoyés	Temps d'exécution moyen
Sans congestion :		
SLR modifié	96.4	2.72 µs
Deviating SLR	87.3	2.63 µs
Avec congestion :		
SLR modifié	–	–
Deviating SLR	135.0	4.10 µs

La Figure 6.1 montre les scénarios simulés. Le paquet se propage vers le haut, de la zone initiale située en bas de l'image, vers la zone destination située en haut.

On voit d'abord que l'algorithme de déviation fonctionne comme prévu dans un réseau qui présente de la congestion : il détecte correctement la congestion, dévie le paquet et contourne la zone encombrée, et route le paquet avec succès vers la zone destination, et ce dans chacune des simulations, de la même façon que montrée dans la Figure 6.1.

Nous comparons ensuite la version modifiée de SLR et deviating SLR en termes de nombre de paquets envoyés et de temps écoulé entre l'envoi initial et la réception du paquet dans la zone destination. La Table 6.2 montre les résultats de cette évaluation. Les deux protocoles se comportent de manière très similaire dans un réseau ne présentant pas de congestion (les deux premières lignes de la Table 6.2), les seules différences sont dues à l'aléatoire induit par les temps d'attente du backoff flooding.

Dans un réseau présentant de la congestion, notre solution requiert $135/87.3 = 55\%$ de message et $4.10/2.63 = 56\%$ de temps en plus comparé au cas sans congestion. La version modifiée de SLR (sans la déviation) n'est pas capable d'acheminser correctement le paquet dans un réseau présentant de la congestion, aucun résultat n'est donc disponible dans ce cas-là.

La Figure 6.1 montre aussi que la déviation fait partie du processus de routage : ni l'émetteur initial ni le destinataire ne peuvent détecter que le paquet a été dévié durant sa transmission ; deviating SLR ne réduit pas le débit du trafic comme pourrait faire TCP par exemple. Comme expliqué dans la Section 6.6, cette approche consomme l'énergie (et les ressources) d'une partie non utilisée du réseau pour ne pas trop dégrader le délai de bout en bout, même dans les cas où le chemin devrait croiser une zone de congestion. De plus, il induit une forme de partage de charge dans la mesure où le protocole emploie des zones du réseau non sollicitées pour remplir ses objectifs.

6.8/ COLLABORATION

Ces travaux sur la déviation de route ont été menés en étroite collaboration avec Florian Büther, doctorant à Institute of Telematics de l'Université de Lübeck en Allemagne. Cette collaboration s'est faite après une première rencontre lors de la conférence NanoCom 2018, puis lors d'un mini workshop organisé entre notre groupe de recherche et le leur. Le fruit de cet échange fut notamment l'ensemble des recherches et la rédaction de l'article traitant de la déviation. L'ensemble de ces travaux a été mené main dans la main via de nombreux échanges de mails et des visioconférences régulières et via un depôt gitLab.

6.9/ CONCLUSION

Nous avons présenté un algorithme de détection de congestion pour les nanoréseaux et de déviation de route pour éviter les zones de congestion. La détection et la déviation sont faites par les routeur eux même. La détection prend en compte la nature des communications utilisant TS-OOK (l'entrelacement des paquets) en particulier le ratio entre le nombre de paquets étant en train d'être reçus et le nombre maximal de paquets pouvant être reçus en parallèle sur chaque nœud. Si de la congestion est détectée, il utilise la largeur m du chemin SLR pour dévier la route et exploiter des parties moins sollicitées du réseau.

Notre évaluation montre que cette approche parvient à délivrer le paquet avec succès, là où il aurait autrement été perdu à cause de la congestion. La déviation fait partie intégrante du routage, les paquets ne sont déviés que si cela est nécessaire et ce mécanisme ne requiert aucun message de contrôle supplémentaire.

Nous avons validé cette approche en utilisant des simulations sur des scénarios en 2D où un paquet est dévié pour contourner une zone de congestion. Cette déviation augmente le nombre de paquets envoyés de 55 % et prend 56 % plus de temps pour délivrer le paquet. Si aucune congestion n'est présente dans le réseau, notre algorithme ne modifie pas le chemin et le processus de transmission est aussi efficace que la version modifiée de SLR avec seulement le backoff flooding. Il est probable que deviating SLR soit encore plus efficace dans des réseaux 3D, où les possibilités de chemins ne se coupant pas sont plus nombreuses.

La détection de la congestion et le mécanisme de déviation peuvent être séparés en deux algorithmes très distincts : Il est donc possible de changer une seule des deux parties de l'algorithme. De plus, les deux techniques peuvent être utilisées séparément dans le cadre d'autres applications, par exemple, pour dévier une route autour d'un trou dans le réseau.

On suppose que les résultats donnés dans l'étude de ce protocole varient en fonction de la topologie du réseau, de la "forme" de la zone subissant de la congestion, du moment et de l'endroit où la congestion survient (proche de la source ou de la destination) etc. Cette étude représente avant tout une preuve de concept et un premier jet pour un algorithme plus complexe pouvant, par exemple, gérer des scénarios en 3D. Ce protocole a été développé en partenariat avec Florian Büther (doctorant allemand), et la collaboration est encore en cours aujourd'hui. Une suite pour ces travaux est donc fortement probable.

7

ENDORMISSEMENT DES NŒUDS

L'idée de cette contribution est de profiter des spécificités des nanoéquipements, de TS-OOK et de l'entrelacement de paquets sur le canal pour diminuer la densité perçue par les nœuds et optimiser l'utilisation du canal. En effet, en faisant dormir une partie des nœuds dans chaque zone de réseau à chaque instant, tous les nœuds ne participent pas à toutes les communications. Le réseau se comporte alors un peu comme si le nombre de voisins réels était plus faible. Les contributions présentées dans ce chapitre ont été menées en collaboration avec Benchaïb Yacine ATER dans l'équipe OMNI durant ma troisième année de thèse (2018/2019).

Les simulations utilisant BitSimulator qui sont présentées dans ce chapitre ont été réalisées avec la dernière version au moment de la rédaction de ce manuscrit ; c'est-à-dire la version 0.9.4. Les contributions présentées de se chapitre reposent principalement sur l'article [11].

7.1/ INTRODUCTION

Le processus de diffusion peut consommer beaucoup de ressources. Nous avons déjà proposé une méthode, le backoff flooding au Chapitre 5, qui réduit drastiquement le nombre de forwardeurs et par conséquent le nombre de paquets envoyés lors d'une diffusion. Cependant, le nombre de réceptions lui n'est pas réduit, c'est-à-dire tous les nœuds voisins d'un transmetteur reçoivent le message et ce même si seulement quelques-uns ont besoin de le forwarder. Recevoir un paquet consomme de la puissance de calcul et de la mémoire (ainsi que des ressources physiques, les buffers de réception etc.) Ce chapitre se penche sur ce problème. La méthode proposée endort et réveille les nœuds de sorte que les nœuds endormis ne soient pas capables de recevoir des paquets. Seuls les nœuds éveillés reçoivent et traitent les paquets qui arrivent.

L'endormissent des nœuds d'un réseau est une solution bien connue. Habituellement, les nœuds restent éveillés pour une durée suffisamment longue pour envoyer et/ou recevoir plusieurs paquets, puis s'endorment à nouveau pour une durée prédéfinie. La particularité de notre solution, est que nous utilisons des cycles de sommeil/éveil très courts, particulièrement appropriés aux communications dans les nanoréseaux et à la modulation TS-OOK. En effet, dans notre solution, les nœuds ont des cycles sommeil/éveil d'un T_s, qui est le temps qui sépare deux bits consécutifs d'un même paquet. L'idée est de faire rester éveillé chaque nœud durant une certaine proportion de chaque T_s, et donc de dormir le reste du temps. Étant donné que T_s est une durée courte, on parle d'endor-

missement à grain fin ou de nanosleeping. Les nœuds s'éveillent et s'endorment donc plusieurs fois durant la réception d'un même paquet.

On se questionne donc quant à l'effet d'un tel type d'endormissement sur les connexions dans le réseau. Trois paramètres nous intéressent tout particulièrement : la couverture, la connectivité, et le nombre de paquets reçus. La couverture, comme montré plus loin, est définie comme étant l'ensemble des points de l'espace dans le réseau qui sont "visibles" par au moins un nœud, métrique utile dans le cas des réseaux de surveillance par exemple. La connectivité est définie comme le nombre de zones connectées (capable de communiquer) avec d'autres zones, une zone étant simplement une région du réseau ; on dit que deux zones sont connectées au temps t si chacune d'entre elles a au moins un nœud réveillé au temps t capable de communiquer directement (les nœuds sont donc voisins). Enfin, le nombre de paquets reçus décrit le fait que lorsqu'un nœud émet un paquet, d'autres nœuds le reçoivent ; cette métrique est différente de la connectivité étant donné qu'elle tient compte du délai de propagation. Deux nœuds peuvent ne pas être connectés (pas réveillés en même temps) mais peuvent tout de même communiquer si le nœud récepteur se réveille entre le moment où l'émetteur a transmis un bit et le moment où ce bit arrive au niveau du récepteur ; les communication peuvent donc ne pas être symétriques. On rappelle qu'à cette échelle de temps et de taille, le délai de propagation (la vitesse de la lumière) ne peut pas être négligé, même entre deux nœuds voisins, comme mentionné dans la section 2.1.

Dans ce chapitre, nous proposons une méthode d'endormissement appropriée aux nanoréseaux. Pour tester notre algorithme, nous avons utilisé deux simulateurs. Un simulateur spécifique a développé pour tester la connectivité dans un nanoréseau implémentant notre solution, et BitSimulator pour lequel nous avons ajouté un nouveau type de nœud capable d'effectuer des cycles d'endormissement. Même si leurs environnements ne sont pas tout à fait les mêmes, nous pouvons comparer leurs résultats en termes de connectivité et de diffusion de message. Les résultats des deux simulateurs montrent que dans les réseaux denses il est possible de faire dormir les nœuds une grande portion de temps tout en maintenant la connectivité du réseau.

dépendance :

SLR,DEDEN

7.2/ TRAVAUX CONNEXES

Quand les nœuds capteurs sont réveillés et sans configuration particulière, ils écoutent le canal en continu pour la réception de nouveaux paquets. Ces paquets doivent être traités par le nœud, même si le nœud n'est pas destinataire du paquet. Cette situation a un impact significatif sur la consommation d'énergie des nœuds [10]. De plus, dans le cas des nanoréseaux, chaque nœud transmet les messages reçus à tous les nœuds dans son voisinage. Ce type de comportement génère un important trafic sur le réseau, en particulier dans les réseaux très denses et très redondants, ce qui peut provoquer un grand nombre de collisions entre les paquets [51] et, dans notre cas (nanoréseaux reposant sur TS-OOK), une saturation des ressources de réception. Faire dormir les nœuds est une méthode particulièrement adaptée aux situations évoquées. Nous allons passer en revue différentes méthodes d'endormissement proposées dans la littérature.

7.2.1/ LA PORTÉE DE COUVERTURE DANS LES RÉSEAUX DE CAPTEURS

Un des problèmes fondamentaux dans les réseaux de capteurs (qui ont des similarités avec nos nanoréseaux) est la couverture de perception. Cette métrique indique à quel point un réseau de capteurs couvre la zone qu'il doit surveiller. La couverture est définie par la probabilité qu'a un point quelconque dans le réseau d'être situé dans la portée d'au moins un nœud, et ce, à un moment donné. Autrement dit, c'est le pourcentage de surface couverte par au moins un nœud. On parle parfois des régions non couvertes sous le nom : régions vacantes.

Habituellement, les travaux traitant de la couverture dans les réseaux de capteurs se focalisent sur la qualité d'un service de surveillance, quand un réseau est peu dense, c'est-à-dire non-complètement couvert pas les nœuds. Les questions assez naturelles qui se posent alors incluent : Quel est le pourcentage du réseau qui n'est pas couvert ? Quel est le temps moyen de non-couverture d'une zone ? Quelle est la durée maximale de non-couverture d'une zone (de combien de temps dispose un intrus avant d'être perçu) ?

Habituellement dans les travaux que l'on trouve dans la littérature, une k-couverture avec $k = 1$ est considérée, une zone étant couverte quand 1 nœud s'y trouve. Certains travaux généralisent pour $k > 1$, par exemple : $k > 2$ est souhaitable pour des raisons de tolérance aux pannes, et $k = 3$ est utile pour géolocaliser un intrus par triangulation [28]. Dans les travaux menés durant cette thèse, on considère le cas $k = 2$, car nous nous intéressons à un aspect communication.

Plusieurs articles dans la décennie 2000–2010 traitent de la couverture dans les réseaux de capteurs. Cependant, il est important de noter que la couverture est différente de la connectivité, comme décrit dans la phrase suivante : "couverture de surveillance et connectivité du réseau" (traduit de [58]). Par exemple, "un réseau connecté ne garantit pas sa couverture indépendamment des portées. C'est parce que la couverture s'intéresse aux endroits non couverts, alors que la connectivité assure que tous les nœuds éveillés soient connectés. [...] Si une région est couverte, alors les nœuds couvrant cette région sont connectés tant que $R_c \geq 2R_s$" [58], où R_c est la portée de communication, et R_s la portée de perception.

Au lieu de cela, on considère le cas de réseaux denses, où le réseau est pleinement couvert ; non seulement chaque nœud est couvert, mais chaque point de l'espace est couvert par une multitude de nœuds. On fait dormir les nœuds (pour réduire la consommation de ressources du réseau), faisant ainsi en sorte que quelques nœuds seulement soient réveillés en même temps dans chaque voisinage. La question est : Quelle est la proportion de temps que les nœuds peuvent passer à dormir tout en maintenant une certaine connectivité dans le réseau ?

[56] calcule simplement la probabilité d'avoir un nombre donné de nœuds dans une zone (2D) ou dans un espace (3D) donné étant donnée une certaine densité de nœuds et montre que c'est une distribution de Poisson.

Plus spécifiquement, plusieurs articles considèrent le cas où chaque point de la surface est dans la portée de communication de plusieurs capteurs, et s'intéressent au nombre de nœuds qu'il est possible d'éteindre (d'endormir) pour maintenir la connectivité tout en réduisant les ressources consommées. Une partie de ces travaux est présentée dans la suite, et nous pointerons les hypothèses qui ne sont pas pertinentes dans le cas des nanoréseaux.

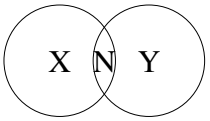


FIGURE 7.1 – Un exemple montrant la différence entre couverture et connectivité, et que simplement vérifier les voisins n'assure par la connectivité : quand le nœud N au milieu se réveille, il voit deux nœuds voisins, et donc il se rendort ; cependant, le réseau n'est pas connecté, étant donnée que les nœuds réveillés X et Y ne peuvent pas communiquer entre eux.

Par exemple, [37] traite des "réseaux à large échelle". Les auteurs définissent trois mesures de la couverture, à savoir la couverture de zone (sans l'endormissement des nœuds), la couverture des nœuds, et la détectabilité. La mesure qui est intéressante pour nous est la proportion de couverture des nœuds, c'est-à-dire la proportion de nœuds qui peuvent être retirés (ou endormis) sans réduire la zone couverture (couverture des nœuds). Bien que la proportion de couverture de nœud est similaire à notre métrique, on note deux différences : nous ne calculons pas le nombre de nœuds à éteindre (endormir) mais la durée de l'endormissement ; nous n'avons pas besoin de maintenir la connectivité, mais de calculer la connectivité en fonction de la durée d'endormissement. Cependant, nous remarquons que cet article ne donne pas d'analyse de la couverture des nœuds, c'est-à-dire le nombre de nœuds à éteindre (endormir) pour maintenir une connectivité donnée. Les auteurs font simplement remarquer que : "Il est difficile de trouver une expression de forme fermée pour la couverture de nœuds dans un réseaux de capteurs 2D", et simulent un nombre relativement petit de nœuds. Pour conclure, cet article ne se préoccupe pas des réseaux denses (où le nombre de nœuds à dormir est important) et ne donne pas d'intuitions (ni de formule) quant au nombre de nœuds pouvant dormir tout en maintenant la connectivité.

[50] propose une méthode pour éteindre et allumer les nœuds pour réduire l'énergie utilisée par ces derniers tout en maintenant la couverture du réseau. Cette méthode se déroule en deux étapes : les nœuds envoient leurs positions à leurs voisins, ensuite la deuxième étape, les nœuds calculent la portée de communication de leurs voisins pour savoir s'ils peuvent s'endormir ou pas. La méthode a été testée sur 100 nœuds, et n'est pas appropriée dans notre cas de réseaux très denses. Le nombre très élevé de voisins causerait la génération de très nombreux paquets, de plus, les nanonœuds n'ont pas (ou très peu) d'informations quant à leurs positions.

[59] Présente une autre méthode très intéressante. L'idée de base est d'éteindre (endormir) les capteurs redondants pour économiser de l'énergie. Un capteur endormi se réveille de temps en temps et envoie un probe à ses voisins. Il se remet pleinement en route si aucun autre capteur à portée n'est réveillé. Il est possible de modifier la redondance des capteurs réveillés en ajustant la portée de probes. Les nœuds n'épuisent pas leur batterie petit à petit, mais certains d'entre eux sont constamment éveillés jusqu'à l'épuisement de leur énergie, puis d'autres prennent le relais. Les nœuds sont donc éveillés en continu et reçoivent tous les paquets, ce qui peut conduire à un débit d'entrée sur chaque nœud relativement élevé si le réseau est dense. Ce débit d'entrée élevé nécessite beaucoup de ressources, notamment de la mémoire pour stocker (même temporairement) les paquets reçus. Enfin, cette méthode n'assure pas la connectivité étant donné qu'un nœud ne peut pas savoir si ses voisins sont connectés ou non, voir Figure 7.1.

[28] présente une méthode intéressante. Étant donné un ensemble de capteurs déployés dans une zone cible, les auteurs veulent déterminer si la zone est suffisamment k couverte, c'est-à-dire que chaque point de la zone cible est couvert par au moins k capteurs, où k est un paramètre donné. Pour calculer la couverture, cet algorithme utilise la position géographique exacte des nœuds voisins. Les fonctions trigonométriques permettent de calculer la partie du périmètre de la portée de communication qui est couverte par un nœud. La couverture du périmètre se traduit ensuite par la couverture des zones et de chaque point de l'espace. Les auteurs fournissent un algorithme pour calculer k (la k couverture) d'une zone donnée. Les auteurs proposent une méthode pour désactiver certains nœuds tout en maintenant la k-couverture. Lorsqu'un nœud épuise sa batterie, un autre nœud prend sa place. La complexité de cet algorithme est $O(nd \log d)$, où d est la densité et n le nombre de nœuds du réseau. L'algorithme peut être exécuté de manière centralisée ou sur chaque nœud. Les auteurs présentent des simulations prenant en compte jusqu'à 1000 nœuds. Au lieu de cela, nous fournissons une formule qui donne la durée de sommeil pour les nœuds, chaque nœud peut l'utiliser, donc la complexité est en $O(1)$. Cela fonctionne particulièrement bien dans les réseaux denses, par exemple 100000 nœuds et 1000 voisins. Dans notre cas, les nœuds ne connaissent pas leur position exacte, mais seulement la zone dans laquelle ils se trouvent, et ne disposent pas de capacités de traitement suffisantes pour faire des calculs comme ceux proposés dans l'article [28].

Également, on considère que tous les nœuds dorment et se réveillent régulièrement, nous nous intéressons au calcul de la période de sommeil, autrement dit, au nombre de nœuds qui dorment à chaque instant, de sorte que la surface soit 2-couverte. Nous pensons qu'il vaut mieux faire en sorte que tous les nœuds épuisent peu à peu leur énergie que de voir certains nœuds s'éveiller jusqu'à ce qu'ils s'épuisent, après quoi d'autres prennent leur place et ainsi de suite. Cette façon de penser est induite par la capacité des nœuds à recharger leur énergie de façon autonome [32] et donc à former un réseau à énergie perpétuelle. Avec une bonne gestion de l'énergie, un nanoréseau pourrait avoir une durée de vie potentiellement infinie (si on ne tient pas compte des pannes).

D'autres articles considèrent des contraintes qui sortent du cadre des travaux menés durant cette thèse. Par exemple, [48] propose une heuristique pour sélectionner des ensembles de nœuds, mutuellement exclusifs (chaque nœud ne peut se trouver que dans un seul ensemble), de sorte que chaque ensemble de nœuds puisse fournir une couverture complète de la zone contrôlée. Cependant, l'algorithme est centralisé, et est en $O(n^2)$ dans le pire des cas (ce qui n'est pas envisageable au vu du nombre de nœuds impliqués dans nos études), et nécessite beaucoup de mémoire dans les réseaux denses ; les nœuds ont besoin de connaître la position de leurs voisins pour calculer la zone couverte par chacun d'eux.

[49] propose SMACS, un protocole distribué qui permet à une collection de nœuds de découvrir leurs voisins et d'établir des horaires d'émission/réception pour communiquer avec eux. On suppose que les nœuds sont capables de régler la fréquence porteuse sur différentes bandes et que le nombre de bandes disponibles est relativement important. Ce type de technique est impossible dans les nanoréseaux utilisant TS-OOK où il n'y a pas de porteuse. De plus, il n'est pas non possible de prévoir un ordonnancement basé sur le temps d'envoi des pulses ; les pulses ne sont pas reçus au même moment par tous les nœuds dans un même voisinage.

En conclusion, la littérature sur la détection de la couverture est abondante, mais nous n'avons trouvé aucun article contenant une solution pouvant s'adapter à nos contraintes et à nos objectifs. En effet, notre problème est nouveau du point de vue de la densité (les nœuds ont des milliers de voisins par exemple) et de la période de sommeil à grain fin (sur une échelle de temps femtométrique)

7.3/ LE NANOSLEEPING

Nous proposons un type d'endormissement novateur conçu spécialement pour les nanoréseaux. L'idée est de profiter des spécificités du modèle de communication TS-OOK pour faire dormir les nœuds entre la réception de deux pulses. Les nœuds s'endorment et se réveillent de manière cyclique afin de pouvoir recevoir correctement un paquet. Chaque nœud détermine le début de sa période d'éveil aléatoirement. C'est la durée de cette période que l'on va chercher à configurer (la durée d'éveil n'est pas aléatoire).

7.3.1/ SCHÉMA

Faire dormir les nœuds est une solution courante pour économiser les ressources (énergie, mémoire...). Un nœud dormant consomme moins d'énergie (la radio est éteinte et effectue moins de traitement de paquets) et moins de mémoire (ne stocke pas tous les paquets entrants). Dans le schéma de sommeil des réseaux classiques, les nœuds restent éveillés pour recevoir (ou envoyer) au moins un paquet complet. Dans les nanoréseaux utilisant des communications par pulses (TS-OOK), les nœuds peuvent effectuer plusieurs cycles sommeil/éveil pendant la réception d'un seul paquet. Dans notre schéma, le cycle de sommeil des nœuds est comparable à la durée pour recevoir un symbole (bit). Par exemple, avec la transmission d'un symbole $T_s = 100\,000$ fs et une durée de sommeil de 1%, un nœud est éveillé pendant 1000 fs et dort ensuite pendant 99 000 fs avant de se réveiller à nouveau. Les nœuds ne reçoivent que les bits qui arrivent pendant leurs périodes d'éveil. Ainsi, lorsqu'un paquet est envoyé, le nombre de nœuds le recevant dépend du moment choisi par l'expéditeur pour transmettre le message et de la durée du réveil. Étant donné les très courts intervalles impliqués, nous appelons cela le nanosleeping.

L'endormissement peut être mis en œuvre de différentes manières, et nous en présentons trois ici :

1. Les nœuds dorment, consomment moins d'énergie
2. Les nœuds dorment, mais se réveillent aussi pendant la courte période d'envoi de bits
3. Les nœuds ne dorment pas, mais ils ne prennent tout simplement pas en compte les bits reçus pendant la période de "sommeil" ; cela réduit le nombre de paquets traités

Dans la première hypothèse, les nœuds ne peuvent pas recevoir de bits lorsqu'ils dorment. Mais ils ne peuvent pas non plus en envoyer. Nous avons plutôt considéré l'hypothèse 2 (ou 3, vu qu'on ne se préoccupe pas de l'implémentation physique) dans laquelle les nœuds, tout comme dans l'hypothèse 1, ne reçoivent pas les bits lorsqu'ils sont endormis. En revanche, ils peuvent transmettre à n'importe quel moment. On considère

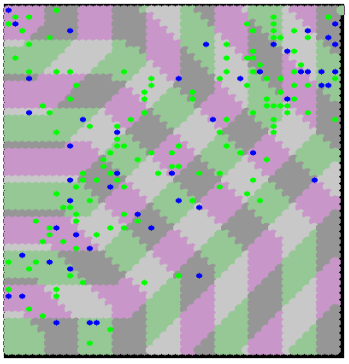


FIGURE 7.2 – Le problème de die out : diffusion incomplète (seulement quelques zones reçoivent le paquet) car la durée d'éveil est trop courte.

que si un nœud a quelque chose à transmettre, que ce soit un paquet initial ou un forward, il se réveille pour émettre.

La durée du réveil (ou le ratio $\frac{eveil}{sommeil}$) doit être choisie avec soin. D'une part, une durée de réveil trop courte peut empêcher la réception du message dans certaines zones. Ceci peut conduire au problème de die out, qui apparaît lorsque le paquet n'atteint pas certaines zones du réseau, comme le montre la Figure 7.2. Cette figure a été tracée avec notre simulateur (BitSimulator) pour le grand scénario avec une durée de réveil de 1500 fs (soit une durée d'éveil de 1,5% du T_s). D'autre part, une durée d'éveil trop longue entraîne un gaspillage des ressources. L'objectif est de trouver une durée où un maximum de ressources sont économisées (les nœuds dorment autant que possible), tandis que le paquet est transmis de manière fiable, c'est-à-dire que le die out n'apparaît pas.

Notre schéma d'éveil est présenté dans l'algorithme 2. Le temps est divisé en cycles de durée T_s (le temps entre l'envoi de deux symboles). Un cycle est divisé en intervalles dont la durée est T_s divisé par le nombre de voisins (que l'on peut obtenir avec DEDeN par exemple). Au lieu du nombre de voisins, on peut utiliser une autre étape de discrétisation. La durée de l'éveil est fixée et est la même pour tous les nœuds. Chaque nœud choisit de se réveiller au début d'un slot choisi au hasard. Une fois réglé, ce temps reste fixe et ne change jamais.

Algorithm 2 Définition du temps d'éveil pour un nœud.

```
nbSlots = number of neighbors of the node
slotLength = T_s / nbSlots
slotNumber = random (nbSlots)
eventDuration = awaken duration
eventBeginning = slotNumber × slotLength
eventEnd = eventBeginning + eventDuration
add the event to the event list
```

7.3.2/ PROPRIÉTÉS

Dans les méthodes classiques d'endormissement, la durée d'éveil est comparable ou plus grande que l'émission ou la réception d'un paquet. La méthode de nanosleeping que nous proposons utilise un cycle de T_s, c'est-à-dire le temps entre deux bits consécutifs.

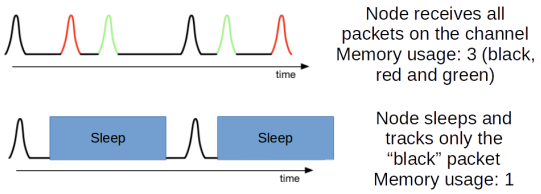


FIGURE 7.3 – Le schéma de d'endormissement proposé réduit le nombre de bits (ou de paquets) reçus : en haut, le nodes est constamment éveillé, en bas, le nœud suit un cycle d'endormissement et d'éveil.

Ce système fonctionne avec des équipements dont les capacités de calcul sont très limitées et il est entièrement distribué et auto-organisé.

Notre solution est entièrement distribuée, les nœuds décident au hasard quand ils commencent à se réveiller, la durée de l'éveil quant à elle est fixée. Une fois qu'un nœud choisit quand il est réveillé, il ne change jamais son cycle. Autrement dit, un nœud reste éveillé pendant un temps donné, puis dort pour compléter un T_s avant de se réveiller à nouveau, etc.

Étant donné que, lorsqu'un nœud dort, il ne reçoit pas de paquets, ce schéma réduit la mémoire utilisée par les nœuds, car les nœuds ne reçoivent (et donc ne traitent) pas tous les paquets qui passent. On peut aussi dire qu'il réduit le débit d'entrée sur les nœuds. Étant donné qu'il réduit le nombre de récepteurs, il réduit aussi naturellement le nombre de forwardeurs : un nœud qui n'a pas reçu un paquet ne peut pas le faire suivre.

Une limitation de ce schéma est qu'il ne fonctionne que lorsque le ratio d'étalement β est le même pour toutes les communications. Ou alors il faut que certains nœuds se synchronisent avec certains β, ce qui peut poser de nouveaux problèmes auxquels nous n'avons pas encore réfléchi.

Il est à noter que, étant donné que la période du cycle éveil/sommeil est de T_s, les bits de n'importe quel paquet sont reçus en même temps modulo T_s, donc un nœud reçoit soit tous les bits d'un paquet, soit aucun d'eux.

Ce schéma de sommeil réduit le débit entrant des nœuds. La partie supérieure de la Figure 7.3 décrit un point de vue de nœud non endormi. Le nœud reçoit 3 paquets (en noir, rouge et vert) et doit les stocker jusqu'à la fin de chaque réception de paquets pour les décoder et donc décider comment traiter le paquet (drop, forward, etc.) Dans ce cas, tous les 3 paquets doivent être stockés au moins pendant la durée de réception. La partie inférieure de la Figure 7.3 décrit le point de vue d'un nœud dormant. Le nœud ne reçoit plus qu'un seul paquet (en noir) puisqu'il dort entre deux bits noirs. Les deux autres paquets doivent être reçus par d'autres nœuds, éveillés à des moments différents. La mémoire utilisée sur chaque nœud est réduite et la charge est mieux répartie entre les nœuds. (Pour des raisons de lisibilité, l'échelle temporelle sur l'axe des abscisses n'est pas respectée).

De plus, ce nanosleeping permet une meilleure utilisation du canal en réduisant la congestion. Comme expliqué dans la section 2.1 les paquets peuvent être entrelacés sur le canal. Il est donc possible que les nœuds reçoivent plusieurs paquets différents "en même temps". Cependant, le nombre de paquets qu'il est possible de recevoir correctement en parallèle est forcément limité par les capacités (physiques ou logicielles) des nœuds.

Pour montrer l'utilité que peut avoir notre méthode d'endormissement sur la congestion, on va s'intéresser à un cas d'exemple similaire à celui utilisé dans le chapitre traitant de la déviation. Dans la Figure 7.4a un cas de référence est montré. Un message est envoyé à travers le réseau (du bas vers le haut). Le message traverse le réseau sans difficulté. Dans cet exemple, nous avons utilisé le backoff flooding couplé à SLR, ce qui réduit le nombre de messages transmis et donc la congestion également.

La Figure 7.4b montre un cas où un message rencontre une zone de congestion et n'arrive pas à destination (la destination se trouve vers le haut du réseau) alors que le backoff flooding couplé à SLR est utilisé. Cependant, on remarque que le nombre de collisions (en rouge) n'est pas très élevé au regard des paquets "ignorés" (en jaune), ce qui signifie que le canal n'est pas l'élément limitant, mais les capacités de réceptions limitées des nœuds. Ces paquets ignorés sont les paquets arrivant sur un nœud, mais ne pouvant pas être pris en charge, car trop de paquets sont déjà en cours de réception. Lorsque que le message arrive dans la zone déjà en congestion, aucun nœud ne peut prendre en charge de nouveaux paquets et le message est alors perdu. Les spécificités de la congestion au sein des nanoréseaux THz sont expliquées plus longuement dans la section 6.3.

Le nanosleeping a deux conséquences, très liées entre elles, qui agissent conjointement sur ce phénomène. Notre solution d'endormissement influe sur les méthodes basées sur le flooding (comme SLR dans l'exemple) en réduisant le nombre de paquets envoyés et le nombre de paquets reçus sur chaque nœud.

- *réceptions :* Lorsqu'un nœud transmet un paquet, tous les nœuds voisins ne reçoivent pas le paquet étant donné qu'une certaine proportion des nœuds dorment au moment de la réception. Inversement, si plusieurs paquets arrivent dans une zone, ils peuvent être pris en charge par des nœuds différents en fonction du moment exact où les différents paquets arrivent dans la zone. Chaque paquet va donc "consommer" les ressources de réception de nœuds différents permettant ainsi une meilleure utilisation du canal étant données que les ressources de réceptions des nœuds sont moins vite saturées.
- *émissions :* Étant donné que tous les nœuds ne reçoivent pas les paquets reçus, tous les nœuds ne participent pas au forward du message. Donc même en utilisant un mécanisme de routage aussi simple que le pure flooding, on réduit le nombre de paquets transmis. Dans cet exemple, on utilise le protocole de routage SLR qui utilise une forme de flooding contrôlé.

La Figure 7.4c montre un cas similaire, mais où le nanosleeping est appliqué à la place du backoff flooding. On peut voir que cette fois-ci, le message réussit à atteindre sa destination située en haut du réseau. De plus, on peut voir tout au long du chemin que tous les nœuds ne reçoivent pas le paquet (nœuds en vert). La problématique concernant notre méthode d'endormissent est donc de trouver la durée d'éveil idéale. Une durée d'éveil trop longue et les ressources ne seront pas économisées au mieux et la congestion peut devenir problématique comme montré en Figure 7.4b, une durée d'éveil trop courte et les nœuds ne vont pas forwarder le message et ne pas atteindre sa destination (ou ses destinations pour le cas d'un multicast ou d'un broadcast comme en Figure 7.2).

Cependant, dans cette étude on ne s'intéresse pas en priorité aux propriétés sur la congestion du nanosleeping, considérations qui seront explorées plus en profondeur dans des études futures. On s'intéresse à la couverture de diffusion, c'est-à-dire la proportion du réseau qui reçoit correctement le message. Habituellement, la diffusion est évaluée en fonction du nombre de récepteurs. Cependant, comme les nœuds dorment pendant la diffusion, le rapport n/N, où n est le nombre de nœuds qui ont reçu le paquet

et N le nombre total de nœuds, n'est pas pertinent. Ce qui est important, c'est que toutes les zones du réseau reçoivent le paquet initial. Une zone est simplement une région du réseau, un carré dans une surface de grille par exemple. Ainsi, dans cette étude, nous considérons la connectivité comme le rapport entre le nombre de zones où au moins un nœud a reçu le message et le nombre total de zones dans le réseau.

On peut noter que parfois dans trois zones consécutives il est possible que les deux zones d'extrémité soient connectées (si la portée de communication est beaucoup plus grande que la taille de la zone), et que la zone centrale ne soit pas connectée. Cela peut se produire lorsqu'un nœud d'une zone d'extrémité envoie un paquet, que le paquet se déplace dans la zone centrale, mais qu'aucun nœud n'est éveillé pendant ce temps, et qu'il arrive finalement dans l'autre zone d'extrémité, où un nœud éveillé le reçoit.

7.4/ SIMULATION ET RÉSULTATS DE CONNECTIVITÉ

L'ensemble de cette section présente des travaux principalement menés par Benchaïb Yacine. Ces travaux sont complémentaires à mes travaux sur l'endormissement et apportent des contributions pertinentes et utiles à mes travaux, c'est la raison pour laquelle je la présente ici.

Un simulateur dans le langage Scala été développé par Yacine Benchaïb. Il place les nœuds dans une grille, les fait dormir de manière cyclique pendant une période donnée et affiche des statistiques sur la connectivité réseau.

7.4.1/ SCENARIOS

Le réseau simulé est rectangulaire, et les nœuds forment une grille 2D, avec s la distance entre 2 nœuds adjacents. La couverture d'un nœud, c'est-à-dire la surface qui reçoit un paquet envoyé par le nœud, est un cercle avec un rayon $cr \gg s$. La densité peut être calculée comme $d = \pi \times cr^2/s^2$.

Notre but est d'évaluer la connectivité dans le réseau, c'est-à-dire le nombre de zones ayant au moins un nœud éveillé et ayant une zone adjacente ayant également au moins un nœud réveillé et au même moment. Les zones sont carrées.

À l'intérieur de chaque T_s, les nœuds sont éveillés pour une durée qui varie entre 0 et T_s, avec T_s de 100 000 fs (temps séparant deux bits à l'intérieur du même paquet).

Nous évaluons deux scénarios. Le large scenario utilise $N = 30 \times 30 = 900$ nœuds, une taille de zone de $5s$, et un rayon de couverture $cr = 5s$ (cinq fois la distance entre 2 nœuds adjacents), ce qui donne un maximum de 78 voisins par nœud (c'est-à-dire $\pi \times cr^2$). Le small scenario utilise $N = 50 \times 50 = 2500$ nœuds, une taille de zone de $10s$, et un rayon de couverture de 10 pour un maximum de 314 voisins (c'est-à-dire $\pi \times cr^2$).

7.4.2/ RÉSULTATS

Nous avons simulé les 2 scénarios. Nous présentons les résultats pour 10 séries, chacune avec une graine différente pour le générateur de nombres aléatoires. Les nombres

aléatoires sont utilisés pour définir le début de la période de sommeil de chaque nœud, soit le moment auquel chaque nœud est réveillé ou endormis.

Les graphiques des la Figures. 7.5 et 7.6 montrent une version modifiée des tracés habituels : la ligne rouge représente la médiane, les limites de la boîte correspondent aux quartiles supérieur et inférieur des données, et les valeurs extrêmes sont placées hors des moustaches pour éviter une vision erronée de la distribution des données. Ces figure montrent que dans le large scenario les nœuds peuvent ne rester éveillé que 5% du temps et une couverture de 100% est presque préservée. De manière générale la plus petite durée d'éveil permettant une couverture de 100% est à privilégier. Aussi, les figures montre, comme attendu, que le pourcentage de zone couvertes augmente avec la durée d'éveil et que la variances diminue également. Au-delà d'une durée d'éveil de 17% pour le small scenario et de 5% pour le large scenario la couverture du réseau dépasse les 99%.

7.5/ RÉSULTATS DE SIMULATION POUR LA DIFFUSION DE PAQUET

Intuitivement, dans les réseaux denses, tous les nœuds n'ont pas besoin de participer au transfert pour couvrir l'ensemble du réseau. Nous validons cette intuition par la simulation de paquet.

Pour notre étude, nous avons implémenté le schéma de nanosleeping dans BitSimulator, et nous présentons les résultats de simulation dans ce qui suit.

7.5.1/ SCÉNARIOS

On essaie d'utiliser exactement les mêmes scénarios que précédemment. Cependant, étant donné les différents environnements de simulation, quelques différences apparaissent, présentées ci-dessous. Les zones ne sont pas carrées, mais sont des minizones SLR, étant donné que nous utilisons SLR qui se charge de l'adressage ; cependant, la surface des minizones est plus ou moins égale à la zone de la section précédente. L'intervalle de temps pour la discrétisation est beaucoup plus petit, égal à l'unité de temps dans le simulateur, c'est-à-dire 1 fs. Le rayon de couverture représente la portée de communication dans le contexte de la diffusion de paquets.

De plus, on souhaite considérer des dimensions réelles, puisque le délai de propagation dépend des distances entre les sources et les destinations. Pour le petit scénario, le monde simulé est un carré de 2 500 000 nm, la distance d entre 2 nœuds adjacents est 49 999 nm, la portée de communication cr est 500 000 nm, les minizones SLR sont 270 000 000 nm, soit au total 100 zones SLR. Pour le grand scénario, le monde simulé est un carré de 3 000 000 000 nm, d est 99 999 nm, cr est 500 000 nm, minizone SLR est 500 000 nm (la zone ne peut être réduite davantage), donnant 45 zones SLR.

Un nœud tiré au hasard lance la diffusion d'un paquet de 1 000 bits. On compare notre solution de nanosleeping avec la méthode classique, où tous les nœuds sont toujours éveillés. En mode sommeil, tous les nœuds recevant le paquet, c'est-à-dire les nœuds éveillés au moment de la réception des bits du paquet, le forwardent. Le routage utilisé est donc du pure flooding. Nous présentons la moyenne du pourcentage de zones atteintes en fonction de la durée du sommeil.

La diffusion utilise pure flooding, qui est la méthode classique et la plus simple pour diffuser des informations à tous les nœuds d'un réseau. Dans cette méthode, le paquet initial est envoyé par le nœud source, et chaque nœud recevant le paquet en transmet une copie (on l'appelle forwardeur). Cette méthode génère une immense surcharge en termes de paquets échangés, un problème connu sous le nom de problème de tempête de diffusion [51]. Au vu de son coût, le pure flooding est rarement utilisé tel quel.

Il est à noter que d'autres méthodes plus efficaces ne sont pas nécessairement compatibles ou utiles dans le cas de l'endormissement. Par exemple, le "backoff flooding" en section 5. Le backoff flooding utilise le fait que dans un réseau sans fil tous les voisins d'un nœud reçoivent les paquets transmis et utilisent cela pour compter le nombre de copies qui ont déjà été envoyées. En dormant, les nœuds ne reçoivent pas tous les messages envoyés par leurs voisins et ne sont donc pas en mesure de compter tous les paquets. Le but du backoff flooding est de faire en sorte que tous les nœuds reçoivent correctement une information donnée (ou un paquet) alors que la technique d'endormissement se contente d'une réception dans chaque zone.

Les flooding probabilistes réduisent également le nombre de forwardeurs. En revanche, le nanosleeping réduit le nombre de récepteurs et, par conséquent, le nombre de forwardeurs. La réduction du nombre de réceptions est une problématique intéressante, notamment dans le cas où plusieurs flux tentent de se diffuser dans le réseau. Si le backoff flooding est employé, chaque nœud va recevoir plusieurs copies d'un même paquet (comme vu à la section 5.2). Si plusieurs flux arrivent au même moment, les nœuds sont susceptibles de voir leurs ressources saturées et donc certains flux pourraient ne pas être diffusés correctement. La solution de nanosleeping permet à plusieurs flux de cohabiter dans le réseau. Dans cette étude, nous ne nous intéressons qu'au cas où un seul flux est diffusé dans le réseau. Les cas présentant plusieurs flux seront étudiés dans des travaux futurs.

Dans le contexte de la diffusion de paquet en mode sommeil, nous avons besoin de coordonnées pour éviter de reculer : sans information de localisation, il est impossible pour un nœud de savoir si le paquet qu'il reçoit provient de la direction du nœud source (le paquet va de l'avant) ou de la direction opposée (le paquet va en arrière). Lorsqu'un nœud achemine un paquet, les coordonnées de la source et de la destination sont écrites dans le paquet. Cela permet au nœud destinataire de savoir s'il doit transmettre le message ou non. Notez que ce problème n'apparaît pas lorsque les nœuds ne dorment pas : tous les voisins reçoivent le paquet et donc (même s'ils ne transfèrent pas le message) sont capables de l'enregistrer, ou son identifiant, permettant aux nœuds de ne pas le transférer deux fois. De plus, comme le paquet vient de la direction de la source au début, le message ne recule jamais.

Nous utilisons le protocole d'adressage et de routage SLR [55], car il est dédié aux nanoréseaux. Les zones correspondent à des zones SLR. On notera que les zones SLR ne sont pas régulières, les formes des zones diffèrent légèrement entre elles. De plus, le SLR d'origine ne convient pas aux nœuds dormants, car lorsqu'un nœud envoie un message, il ne garantit pas qu'au moins un nœud éveillé dans la zone suivante le reçoit. Par conséquent, nous avons utilisé une portée de communication plus petite pendant la phase de balisage. Les minizones ainsi créées garantissent que, quel que soit le nœud qui transmet un paquet, tous les nœuds des zones voisines sont atteints augmentant ainsi les chances qu'une zone voisine ait au moins un nœud réveillé au moment de la réception.

7.5.2/ RÉSULTATS

Dans cette section, on définit la couverture comme le nombre de zones atteintes divisé par le nombre total de zones. Une couverture de 100% signifie que toutes les zones SLR ont reçu le message.

La Figure 7.7 montre la couverture pour les deux scénarios. Intuitivement, lorsque la durée d'éveil augmente, de plus en plus de nœuds sont éveillés en même temps et donc reçoivent le message, cf. Figure 7.8. Chaque nœud qui reçoit le paquet participe au processus de forward. Comme les nœuds sont plus ou moins répartis uniformément entre les zones, l'augmentation du nombre de réceptions augmente les chances que chaque zone soit atteinte par au moins une copie du paquet. Bien sûr, on observe un phénomène de seuil de couverture : lorsque le nombre de réceptions est suffisamment élevé pour aboutir à une couverture de 100%, plus de réceptions ne seraient pas utiles.

La Figure 7.8 compare également la solution de nanoslepping et le pure flooding par rapport au nombre de réceptions. Dans le cas du sommeil, le nombre de réceptions n'est pas linéaire en raison des limites de ressources. En effet, lorsque les nœuds dorment beaucoup, le nombre de paquets envoyés (comme le montre la Figure 7.9) est assez petit et tous peuvent être traités. Ceci explique la première partie de la courbe. Ensuite, les nœuds commencent à atteindre leur limite de buffers de paquets et ne peuvent plus traiter tous les paquets entrants, donc la courbe augmente de moins en moins, pour finalement atteindre un seuil donné par le nombre de réceptions de la méthode du pure flooding. Autrement dit, en cas de pure flooding (ce qui équivaut à dormir avec une durée d'éveil de 100%), chaque nœud reçoit et transmet le message qui sature le réseau. Le mécanisme de sommeil montre que cette saturation est (presque) atteinte avec une durée d'éveil de 45%. C'est pourquoi la simulation n'a pas été conduite au-delà de cette valeur.

Le nombre total d'envois de paquets, indiqué dans la Figure 7.9, a la même tendance que les réceptions. Cela s'explique, car les deux nombres sont étroitement liés : plus il y a de réceptions, plus il y a de forwards, et vice-versa.

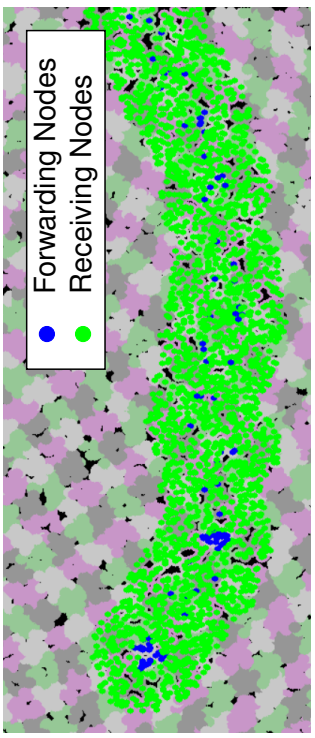
De même les Figures 7.7, 7.8 et 7.9 montrent le même phénomène sur le petit scénario. Cependant, dans ce scénario, pour une même valeur de durée d'éveil, le nombre de réceptions et le nombre de forwards sont plus éloignés du seuil donné par la méthode de pure flooding. Ceci est dû à la différence de densité entre les deux scénarios, le petit scénario a une densité plus faible, donc les ressources du réseau sont moins gaspillées. De plus, la couverture prend plus de temps pour atteindre (et se stabiliser à) 100% pour les mêmes raisons. Comme la densité est plus petite, il y a moins de nœuds éveillés pour une durée de réveil donnée.

Dans les scénarios flooding impliquant un unique flux, les collisions ne sont pas très destructrices. Les collisions ne se produisent pas sur tous les nœuds d'un voisinage donné parce qu'à ces échelles de temps et de taille, la vitesse de propagation du signal ne peut pas être négligée, autrement dit deux voisins ne reçoivent pas le même paquet en même temps et n'ont donc pas les mêmes collisions. Lors de la réception de paquets entrés en collision, il est très probable qu'au moins un de ses voisins ait correctement reçu le même paquet. Ce nœud a donc plusieurs chances de recevoir correctement le paquet : plusieurs chances lorsque que le paquet progresse, car potentiellement plusieurs nœuds dans la direction de la source ont forwardé le paquet, puis de nouveau plusieurs chances car potentiellement plusieurs voisins du nœud (dans la direction opposée) ont forwardé le paquet. De plus, nous sommes intéressés par la couverture en termes de zones, ce

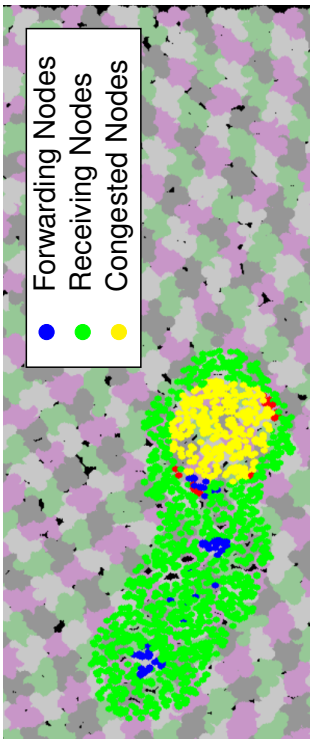
qui signifie qu'il suffit qu'un seul nœud par zone reçoive correctement le paquet. Par conséquent, il n'est pas nécessaire de mener une étude approfondie sur le nombre de collisions.

7.6/ CONCLUSION

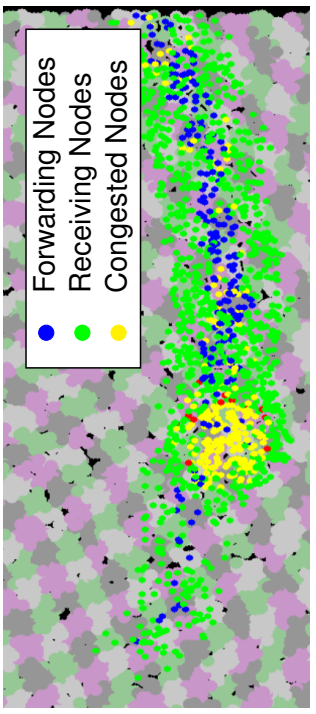
Ce chapitre présente un schéma de nanosleeping, qui fait dormir les nœuds et les réveille régulièrement avec un cycle très court, comparable à la durée du temps entre deux envois consécutifs de bits. Les résultats montrent qu'il est possible de réduire considérablement les données envoyées et transmises par la méthode de diffusion, et ce, de façon distribuée et auto-organisée. L'approche que nous avons proposée permettra à l'utilisateur de choisir une couverture désirée du réseau, et de connaître la durée de réveil correspondante. Les travaux que nous allons mener ensuite visent notamment à mieux comprendre la corrélation entre la connectivité et la réception des messages.



(a) Témoin : message traversant le réseau en utilisant le backoff flooding



(b) Problème : le message n'atteint pas la destination à cause de la congestion



(c) Solution : le message atteint destination grâce au nanosleeping

FIGURE 7.4 – Effet du nanosleeping sur une route SLR rencontrant de la congestion

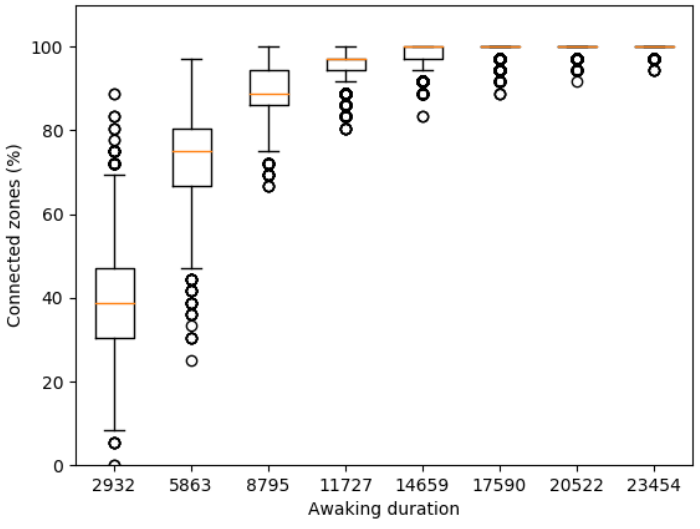


FIGURE 7.5 – Pourcentage de zone couverte pour le grand scenario.

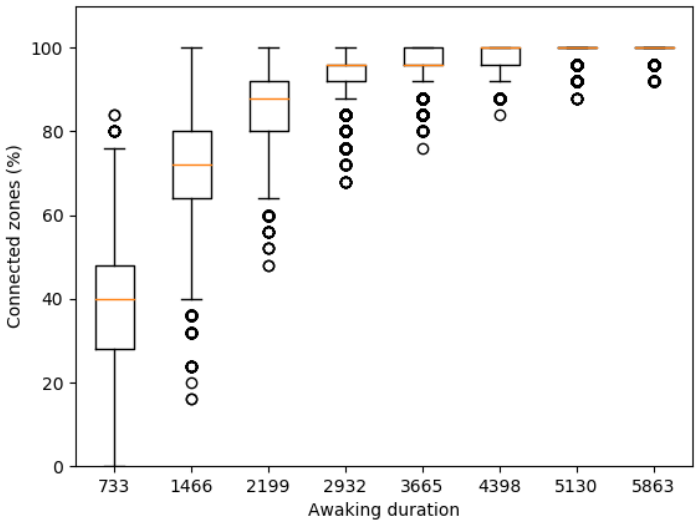


FIGURE 7.6 – Pourcentage de zone couverte pour le grand scenario.

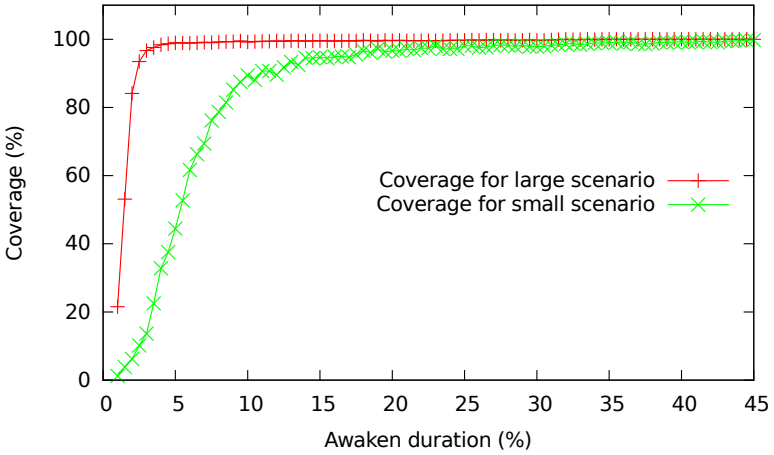


FIGURE 7.7 – Couverture pour différentes durées d'éveil.

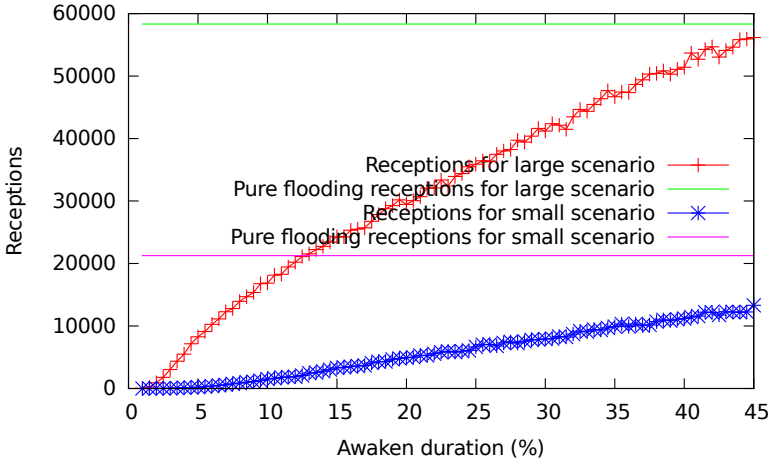


FIGURE 7.8 – Nombre de réceptions pour différentes durée d'éveil.

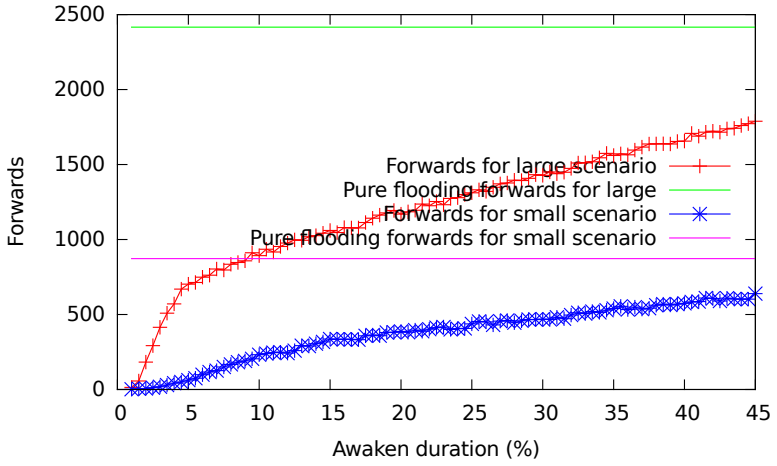


FIGURE 7.9 – Nombre de paquets forwardés pour différentes durée d'éveil.

8

CONCLUSION ET TRAVAUX FUTURS

L'ensemble des travaux menés au cours de cette thèse de doctorat a pour but de créer des outils utiles à l'étude des nanoréseaux et de mené des premières études sur les aspects réseaux de ces derniers. Un outil de visualisation, VisualTracer a également été développé. Cet outil permet une visualisation graphique du déroulement des simulations et offre une aide précieuse dans la compréhension de certains phénomènes (comme les collisions par exemple) et du comportement des différents algorithmes.

D'une part des outils logiciel avec BitSimulator, un simulateur de nanoréseaux présentant des fonctionnalités uniques et qui prend en compte une partie des spécificités de ce type de réseaux. Il est notamment capable de gérer la réception de multiples paquets sur un même nœud et de considérer les collisions au niveau des bits.

De nombreuses améliorations et optimisations sont encore à prévoir. Par exemple, un nouveau modèle de collisions est en train d'être développé. Ce nouveau modèle sera capable de prendre en compte l'accumulation d'énergie par un ensemble de pulses envoyés par des émetteurs lointains qui pris séparément ne pourraient pas être détecté mais qui collectivement peuvent impacter d'autres communications. Une optimisation concernant la mémoire utilisée par le stockage des différentes copies d'un même paquet est également prévue.

Cette thèse traite aussi des outils algorithmiques avec un ensemble de protocoles permettant d'améliorer les communications dans les nanoréseaux notamment en évitant la congestion.

DEDeN est un estimateur de densité spécialement conçu pour fonctionner dans des réseaux très denses (des milliers de voisins par nœuds). Cette estimation est faite avec un faible coût (en nombre de paquets envoyés) si la densité du réseau est suffisamment dense. C'est un algorithme entièrement distribué qui n'a besoin que d'une synchronisation très locale assurée par sa conception. DEDeN permet à tous les nœuds du réseau d'obtenir une estimation du nombre de voisins les entourant et ce même dans un environnement hétérogène. Cette information est très utile pour optimiser d'autre protocoles de communication et de paramétrer au mieux le comportement de chaque nœud. DEDeN est configurable, permettant de choisir une marge d'erreur pour l'estimation et une confiance dans cette marge d'erreur. Ces choix impactent le coût de l'algorithme. Une analyse analytique de la marge d'erreur et de la confiance a été proposée dans ce manuscrit. Bien que l'estimateur obtienne déjà des résultats très satisfaisants, des travaux sur l'estimation de densité sont toujours en cours.

Il est notamment prévu de rendre l'estimateur compatible avec les réseaux dynamiques.

En effet, dans sa version actuelle, DEDeN ne fonctionne que sur des réseaux statiques. En cas de mobilité des nœuds l'estimation doit être entièrement relancée. Une des amélioration possible pour DEDeN est de le rendre capable de fonctionner dans un réseaux où les nœuds vont et viennent.

Le backoff flooding est une optimisation pour les algorithmes se basant sur le principe du flooding. Cette optimisation fonctionne avec des flooding naïfs comme le pure flooding mais aussi avec d'autre types de flooding, comme SLR par exemple. Le backoff flooding est un algorithme qui réduit le coût d'un flooding : moins de paquets sont nécessaires pour que les nœuds reçoivent l'information diffusée. Dans le cas d'une diffusion à tout le réseau, la couverture est assurée, à la différence du flooding probabiliste qui lui peut subir le problème du die out. Cette optimisation se base sur un schéma de forward "counter-based" et implique donc de stoker les paquets pour quelque temps. Cependant, nous avons fait une étude analytique de la durée de ce stockage et avons montré que cette durée est faible.

La déviation est un algorithme inspiré des déviations routières. Cet algorithme repose à la fois sur le routage SLR et sur le backoff flooding. Nous avons implémenté un système de détection de la congestion qui se base sur le nombre de paquets en train d'être reçus sur chaque nœud. Quand ce système détecte de la congestion, la déviation entre en jeu et utilise les spécificités de SLR pour dévier la route SLR et éviter les zones de congestion. L'ensemble de cette contribution a été produite en collaboration avec Florian Büther, un doctorant à l'Institute of Telematics, Université de Lübeck, Allemagne.

Cet algorithme peut déployer son plein potentiel dans les réseaux en 3D, il est donc prévu d'améliorer cet algorithme pour le rendre compatible avec des réseaux en 3D pour exploiter pleinement le caractère multi-chemins des nanoréseaux dense. Ainsi les flux seront capable de s'entremêler sans jamais se couper, à la façon d'un plat de spaghetti, pour utiliser au mieux les ressources disponible dans l'ensemble du réseau.

Nous avons enfin proposé un schéma d'endormissement des nœuds. Cet endormissement est novateur de part les échelles de temps qu'il considère. L'endormissement utilise une échelle de temps très petite, d'une part à cause des durées impliquées dans la transmission des paquets, et d'autre part car le cycle d'endormissement / éveil se fait à l'échelle des bits et non des paquets. Ce schéma d'endormissement permet une meilleure répartition des ressources entre les nœuds d'un même voisinage et donc de profiter au mieux des capacités du canal THz. Une études sur la couverture du réseau a été mené pour étudier l'impact de l'endormissement sur la couverture dans un réseau quadrillé par le protocole SLR.

Cependant il reste compliqué de rejoindre un nœud en particulier. Il est probable que le nœud destinataire dorme lors de la réception du paquet. Un système d'adressage par fonctionnalité capable de gérer l'endormissement est prévu pour de prochains travaux. De plus, un estimateur de densité peut aider à adapter la durée d'endormissement de nœuds. Cela peut permettre, dans les réseaux présentant une densité hétérogène, d'avoir une bonne couverture sans gaspiller de ressources.

Cette thèse à posé des bases pour des études futures concernant les nanoréseaux, tout en posant les premières pierres de mécanismes visant à établir du contrôle de congestion dans ce type de réseaux encore très jeune. Les nanoréseaux restent méconnus et "promettent" des découvertes inattendues et des applications surprenantes dans les année à venir.

BIBLIOGRAPHIE

[1] Nael Abu-Ghazaleh, Kyoung-Don Kang, and Ke Liu. Towards resilient geographic routing in WSNs. In *1st ACM international workshop on Quality of service and security in wireless and mobile networks (Q2SWinet)*, pages 71–78, Montreal, Canada, October 2005. ACM.

[2] Ian F Akyildiz and Josep Miquel Jornet. The internet of nano-things. *IEEE Wireless Communications*, 17(6) :1–6, 2010.

[3] M. Ali, B. G. Stewart, A. Shahrabi, and A. Vallavaraj. Congestion adaptive multipath routing for load balancing in mobile ad hoc networks. In *8th International Conference on Innovations in Information Technology (IIT)*, pages 305–309, Abu Dhabi, United Arab Emirates, March 2012. IEEE.

[4] Aboli Arun Anasane and Rachana Anil Satao. A survey on various multipath routing protocols in wireless sensor networks. In *International Conference on Communication, Computing and Virtualization (ICCCV)*, pages 610–615, Mumbai, India, February 2016. Elsevier.

[5] T Arrabal, F Büther, D Dhoutaut, and E Dedu. Congestion control by deviation routing in electromagnetic nanonetworks. In *6th ACM/IEEE International Conference on Nanoscale Computing and Communication (NanoCom)*, pages 1–6, Dublin, Ireland, September 2019. ACM/IEEE.

[6] Thierry Arrabal, Dominique Dhoutaut, and Eugen Dedu. Efficient density estimation algorithm for ultra dense wireless networks. In *27th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9, Hangzhou, China, July-August 2018. IEEE.

[7] Thierry Arrabal, Dominique Dhoutaut, and Eugen Dedu. Efficient multi-hop broadcasting in dense nanonetworks. In *17th IEEE International Symposium on Network Computing and Applications (NCA)*, pages 385–393, Cambridge, MA, USA, November 2018. IEEE.

[8] M. Aslam, N. Javaid, A. Rahim, U. Nazir, A. Bibi, and Z. A. Khan. Survey of extended leach-based clustering routing protocols for wireless sensor networks. In *14th IEEE International Conference on High Performance Computing and Communication and 9th IEEE International Conference on Embedded Software and Systems*, pages 1–8, Liverpool, UK, June 2012. IEEE.

[9] B. Atakan, O. B. Akan, and S. Balasubramaniam. Body area nanonetworks with molecular communications in nanomedicine. *IEEE Communications Magazine*, 50 :28–34, 2012.

[10] Prithwish Basu and Jason Redi. Effect of overhearing transmissions on energy efficiency in dense sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 196–204. ACM, 2004.

[11] Y Benchaïb, T Arrabal, D Dhoutaut, and E Dedu. Influence of node nanosleeping on nanonetwork connectivity. In *International Conference on Embedded Wireless Systems and Networks (EWSN)*, pages 1–12, Lyon France, February 2019. ACM.

[12] Giuseppe Bianchi and Ilenia Tinnirello. Kalman filter estimation of the number of competing terminals in an IEEE 802.11 network. In *22th IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9, San Francisco, CA, USA, July 2003. IEEE.

[13] Nicolas Boillot, Dominique Dhoutaut, and Julien Bourgeois. Going for large scale with nano-wireless simulations. In *2nd ACM International Conference on Nanoscale Computing and Communication (NanoCom)*, pages 1–2, Boston, MA, USA, September 2015. ACM.

[14] Julien Bourgeois, Benoit Piranda, Andre Naz, Nicolas Boillot, Hakim Mabed, Dominique Dhoutaut, Thadeu Tucci, and Hicham Lakhlef. Programmable matter as a cyber-physical conjugation. In *Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on*, pages 002942–002947. IEEE, October 2016.

[15] Bob Braden, David Clark, Jon Crowcroft, et al. Recommandations on queue management and congestion avoidance in the internet. IETF standard, April 1998. RFC 2309.

[16] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications*, 7 :28–34, October 2000.

[17] Julien Cartigny and David Simplot. Border node retransmission based probabilistic broadcast protocols in ad-hoc networks. *Telecommunication Systems*, 22(1–4) :189–204, January 2003.

[18] Marco Cattani, Macro Zuniga, Andreas Loukas, and Koen Langendoen. Lightweight neighborhood cardinality estimation in dynamic wireless networks. In *13th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–11, Berlin, Germany, April 2014. ACM/IEEE.

[19] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR), October 2003. RFC 3626.

[20] Eugen Dedu, Julien Bourgeois, and Muhammad Agus Zainuddin. A first study on video transmission over a nanowireless network. In *ACM International Conference on Nanoscale Computing and Communication*, 1, pages 1–6, Atlanta, Georgia, USA, May 2014. ACM.

[21] Dominique Dhoutaut, Thierry Arrabal, and Eugen Dedu. BitSimulator, an electromagnetic nanonetworks simulator. In *5th ACM/IEEE International Conference on Nanoscale Computing and Communication (NanoCom)*, pages 1–6, Reykjavik, Iceland, September 2018. ACM/IEEE.

[22] Alotaibi Eiman and Mukherjee Biswanath. A survey on routing algorithms for wireless ad-hoc and mesh networks. *Computer Networks*, 56 :940–965, February 2012.

[23] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer. TCP Friendly Rate Control (TFRC) : Protocol specification, September 2008. RFC 5348.

[24] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP extensions for multipath operation with multiple addresses. IETF standard, January 2013. RFC 6823.

[25] Matthew Gast. *802.11 Wireless Networks : The Definitive Guide*. O'Reilly, April 2002.

[26] Seth Copen Goldstein and Todd C. Mowry. Claytronics : A scalable basis for future robots. In *RoboSphere 2004*, Moffett Field, CA, Nov 2004.

[27] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14 :479–491, June 2006.

[28] Chi-Fu Huang and Yu-Chee Tseng. The coverage problem in a wireless sensor network. *Mobile Networks and Applications*, 10(4) :519–528, August 2005.

[29] D. Johnson, Y. Hu, and D. Maltz. The dynamic source routing protocol (DSR) for mobile ad hoc networks for ipv4, February 2007. RFC 4728.

[30] J Jornet. A joint energy harvesting and consumption model for self-powered nano-devices in nanonetworks. In *IEEE International Conference on Communications (ICC)*, pages 6151–6156, Ottawa, ON, Canada, June 2012. IEEE.

[31] J.M. Jornet and I.F. Akyildiz. Low-weight channel coding for interference mitigation in electromagnetic nanonetworks in the terahertz band. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–6, Kyoto, Japan, 2011. IEE.

[32] Josep Miquel Jornet. A joint energy harvesting and consumption model for self-powered nano-devices in nanonetworks. In *Communications (ICC), 2012 IEEE International Conference on*, pages 6151–6156, Ottawa, ON, Canada, 2012. IEEE.

[33] Josep Miquel Jornet and Ian F Akyildiz. Information capacity of pulse-based wireless nanosensor networks. In *8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 80–88, Salt Lake City, UT, USA, June 2011. IEEE.

[34] Josep Miquel Jornet and Ian F. Akyildiz. Femtosecond-long pulse-based modulation for Terahertz band communication in nanonetworks. *IEEE Transactions on Communications*, 62(5) :1742–1753, May 2014.

[35] James F. Kurose and Keith W. Ross. *Computer Networking : A Top-Down Approach Featuring the Internet*. Pearson Education, Inc., 2003.

[36] Binbin Li, Yuan He, and Wenyuan Liu. Towards constant-time cardinality estimation for large-scale RFID systems. In *44th International Conference on Parallel Processing (ICPP)*, pages 1–10, Beijing, China, September 2015. IEEE.

[37] Benyuan Liu and Don Towsley. A study of the coverage of large-scale sensor networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, pages 475–483, Fort Lauderdale, FL, USA, October 2004. IEEE.

[38] Josep Miquel Jornet Montana. *Fundamentals of Electromagnetic Nanonetworks in the Terahertz Band*. PhD thesis, Georgia Institute of Technology, 12 2013.

[39] Ali N. A, Aleyadeh W, and AbuElkhair M. Internet of nano-things network models and medical applications. In *2016 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 211–215, September 2016.

[40] Giuseppe Piro, Luigi Alfredo Grieco, Gennaro Boggia, and Pietro Camarda. Nano-Sim : simulating electromagnetic-based nanonetworks in the network simulator 3. In *6th International ICST Conference on Simulation Tools and Techniques (SimuTools)*, pages 203–210, Cannes, France, March 2013. ACM.

[41] K. Ramakrishnan, Sally Floyd, and David Black. The addition of explicit congestion notification (ECN) to IP, September 2001. RFC 3168.

[42] Daniel Gutiérrez Reina, Sergio Toral, Princy Johnson, and Federico Barrero. A survey on probabilistic broadcast schemes for wireless ad hoc networks. *Ad Hoc Networks*, 25 :263–292, February 2015.

[43] M. Rey. Transmission control protocol. IETF standard, September 1981. RFC 793.

[44] Niky Riga, Ibrahim Matta, and Azer Bestavros. DIP : Density Inference Protocol for wireless sensor networks and its application to density-unbiased statistics. Technical Report 2004-023, University of Boston, MA, USA, May 2004.

[45] Vikrant Saigal, Ajit K. Nayak, Sateesh K. Pradhan, and R. Mall. Load balanced routing in mobile ad hoc networks. *Computer Communications*, 27(3) :295–305, February 2004.

[46] Zhongmin Shi and Hong Shen. Adaptive gossip-based routing algorithm. In *23rd IEEE International Conference on Computer Communications*, pages 1–4, Phoenix, AZ, USA, April 2004. IEEE.

[47] Prateek K. Singh, Gregory Aizin, Ngwe Thawdar, Michael Medley, and Miquel Jornet. Graphene-based plasmonic phase modulator for graphene-based plasmonic phase modulator for terahertz-band communication. In *Proc. 10th European Conference on Antennas and Propagation (EuCAP)*, pages 1–6, Davos, Switzerland, 2016. IEEE.

[48] Sasa Slijepcevic and Miodrag Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, pages 472–476, Helsinki, Finland, June 2001. IEEE.

[49] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7(5) :16–27, October 2000.

[50] Di Tian and Nicolas D. Georganas. A coverage-preserving node scheduling scheme for large wireless sensor networks. In *1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 32–41, Atlanta, GE, USA, September 2002. ACM.

[51] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless Networks*, 8(2–3) :153–167, March 2002.

[52] Yu-Chee Tseng, Sze-Yao Ni, and En-Yu Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. *IEEE Transactions on Computers*, 52(5) :545–557, May 2003.

[53] A. Tsioliaridou, C. Liaskos, S. Ioannidis, and A. Pitsillides. GPS precision as a function of session duration and reference frame using multi-point software. *GPS Solutions*, 16 :191–196, 2012.

[54] A. Tsioliaridou, C. Liaskos, S. Ioannidis, and A. Pitsillides. Lightweight, self-tuning data dissemination for dense nanonetworks. *Nano Communication Networks (Special Issue on EM Nanonetworks)*, 8 :2–15, 2016.

[55] Ageliki Tsioliaridou, Christos Liaskos, Eugen Dedu, and Sotiris Ioannidis. Packet routing in 3D nanonetworks : A lightweight, linear-path scheme. *Nano Communication Networks*, 12 :63–71, June 2017.

[56] Yun Wang, Xiaodong Wang, Dharma P. Agrawal, and Ali A. Minai. Impact of heterogeneity on coverage and broadcast reachability in wireless sensor networks. In *15th International Conference on Computer Communications and Networks (ICCCN)*, pages 63–67, Arlington, VA, USA, October 2006. IEEE.

[57] Yao Xin-Wei, Wu Ye-Chen-Ge, and Huang Wei. Routing techniques in wireless nanonetworks : A survey. *Nano Communication Networks*, 21(13), September 2019.

[58] Guoliang Xing, Xiaorui Wang, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration for energy conservation in sensor networks. *ACM Transactions on Sensor Networks*, 1(1) :36–72, August 2005.

[59] Fan Ye, Gary Zhong, Jesse Cheng, Songwu Lu, and Lixia Zhang. PEAS : A robust energy conserving protocol for long-lived sensor networks. In *23rd International Conference on Distributed Computing Systems (ICDCS)*, pages 28–37, Providence, RI, USA, May 2003. IEEE.

[60] Hang Yu, Bryan Ng, and Winston K.G. Seah. Pulse arrival scheduling for nanonetworks under limited IoT access bandwidth. In *42nd IEEE Conference on Local Computer Networks (LCN)*, pages 18–26, Singapore, Singapore, October 2017. IEEE.

[61] Hossain Zahed, Xia Qing, and Jornet Josep Miquel. TeraSim : An ns-3 extension to simulate terahertz-band communication networks. *Nano Communication Networks*, 17(9) :36–44, September 2018.

[62] Muhammad Agus Zainuddin, Eugen Dedu, and Julien Bourgeois. Nanonetwork Minimum Energy coding. In *IEEE International Conference on Ubiquitous Intelligence and Computing (UIC)*, 11, pages 96–103, Bali, Indonesia, December 2014. IEEE.

[63] Muhammad Agus Zainuddin, Eugen Dedu, and Julien Bourgeois. Low-weight code comparison for electromagnetic wireless nanocommunication. *IEEE Internet of Things Journal*, 3(1) :38–48, February 2016.

[64] Muhammad Agus Zainuddin, Eugen Dedu, and Julien Bourgeois. SBN : Simple block nanocode for nanocommunications. In *ACM International Conference on Nanoscale Computing and Communication*, 3, pages 1–7, New York City, NY, USA, September 2016. ACM.

[65] Yuanqing Zheng and Mo Li. Towards more efficient cardinality estimation for large-scale RFID systems. *IEEE/ACM Transactions on Networking*, 22 :1886–1896, November 2013.

Document réalisé avec LaTeX et :
le style LaTeX pour Thèse de Doctorat créé par S. Galland — http://www.multiagent.fr/ThesisStyle
la collection de paquets `tex-upmethodology` — http://www.arakhne.org/tex-upmethodology/