Réseaux de capteurs sans fil étendus dédiés aux collectes de données environnementales

Affoua Thérèse Aby

N° d'ordre : D. U : 2671
E D S P I C : 742

UNIVERSITÉ BLAISE PASCAL - CLERMONT II

ÉCOLE DOCTORALE
SCIENCES POUR L'INGÉNIEUR DE CLERMONT-FERRAND

THÈSE

Présentée par

ABY AFFOUA THÉRÈSE

pour obtenir le grade de

DOCTEUR D'UNIVERSITÉ

SPÉCIALITÉ : INFORMATIQUE

Réseaux de capteurs sans fil étendus dédiés aux collectes de données environnementales

Soutenue publiquement le 29 Janvier 2016 devant le jury :

Rapporteurs :	M. Éric RONDEAU	Professeur à l'Université de Lorraine
	M. Bernard TOURANCHEAU	Professeur à l'Université de Grenoble Alpes
Examinateurs :	M. Adrien VAN DEN BOSSCHE	MCF à l'Université de Toulouse 2
	M. Benoit HILT	MCF à l'Université de Haute-Alsace
Directeur :	M. Michel MISSON	Professeur à l'Université d'Auvergne
Encadrant :	M. Alexandre GUITTON	MCF-HDR à l'Université Blaise Pascal

Remerciements

Pour commencer, j'adresse mes remerciements aux personnes qui ont cru en moi et qui m'ont permis d'arriver au bout de cette thèse.

En premier lieu, je remercie mon directeur de thèse M. Michel MISSON pour m'avoir fait confiance dès notre première rencontre où je répétais ma présentation pour postuler à la bourse ministérielle. Vous m'aviez dit une phrase que je n'oublierai jamais : "vous, si vous obtenez la bourse ministérielle pour la thèse, je vous donnerai des cours", vous l'avez dit avec une telle intensité que cela m'a tout de suite mise en confiance. Je ne vous remercierai jamais assez pour votre clairvoyance et votre rigueur scientifique qui ont été d'un apport essentiel dans la réalisation de cette thèse.

Ensuite, tous mes remerciements vont à l'égard de mon encadrant de thèse M. Alexandre GUITTON. Alex, j'ai tant à te dire que, consacrer ces deux pages ne suffiront pas. Le mot merci me parait insuffisant, que Dieu te bénisse pour tout! Sache que c'est un réel délice de travailler avec une personne comme toi. Tu es brillant, rigoureux, compétent dans pas mal de choses et surtout très disponible pour tous. J'étais toujours sure que, ce que j'avais à faire était possible et que je pouvais y arriver, car tu le ferais en un rien de temps si je n'y arrivais pas. Ce qui m'a permis de me surpasser pour être à la hauteur. Merci encore Alex :)

J'exprime également mes remerciements à tous les membres de mon jury.

M. Bernard TOURANCHEAU et M. Éric RONDEAU merci d'avoir accepté d'être les rapporteurs de cette thèse. Le travail de lecture approfondie que vous avez réalisé a apporté une grande amélioration à la qualité de mon manuscrit.

M. Benoit HILT merci d'avoir été examinateur et président de mon jury. Merci

également pour vos remarques et suggestions faites après lecture de mon manuscrit. M. Adrien VAN DEN BOSSCHE, c'est toujours un plaisir d'avoir des discussions de travail avec vous. Merci d'avoir été examinateur de mon travail et d'avoir partagé ce moment avec nous.

J'adresse mes remerciements à tous les membres de l'équipes réseaux et protocoles pour leurs suggestions et contributions. Vous êtes des personnes formidables au milieu desquelles je prend toujours plaisir à travailler. Je remercie particulièrement Marie-Françoise SERVAJEAN, ma Tati:), pour ses qualités humaines exceptionnelles et aussi pour la lecture complète de mon manuscrit. J'adresse ma grande sympathie à mes collègues de bureau, mes amis : notre ambiance de poilu (Xavier) et de poulet piquet (Malick) me donne une raison supplémentaire de venir au bureau tous les matins. Honoré, merci d'avoir immortalisé ma soutenance avec toutes ces photos et vidéos. Malik N'Doye, alias mon Pooooote pour toujours, merci pour ton soutien indéfectible dans la réalisation de mon cocktail ivoirien LOL.

Je remercie aussi les enseignants de R&T à l'IUT pour leur soutien.

Je remercie tous ceux qui de loin ou de près ont apporté leurs pierres à la réalisation de mon édifice : ma thèse.

J'adresse ma profonde gratitude à mon amie, ma sœur Nadine pour le soutien morale qu'elle m'apporte depuis toujours, merci ma Reine et aussi à Philippe pour avoir été toujours là quand j'en ai eu besoin!

Enfin, j'adresse tout mon amour et ma reconnaissance à ma famille.

Maman merci pour tes bénédictions : tu m'as répété tant de fois "que Dieu fasse en sorte que tu obtiennes ce que tu recherches"! C'est l'aboutissement aujourd'hui, merci maman.

Ma grande sœur Jacki, merci pour ton amour, ta confiance et ton soutien qui m'ont porté tout ce temps.

Une pensée pour toi papa qui n'a pas pu voir l'aboutissement de mon travail, je suis convaincue que où que tu sois tu es très fier de ta fille!

Je vais terminer ces remerciements par une dédicace spéciale à la prunelle de mes yeux mon fils Joris Dimitri, pour toutes ces fois où je n'ai pas été là pour prendre soin de toi comme la mère qui t'aime tant que je suis l'aurais voulu. La douleur de ne pas t'avoir à mes côtés a été ma plus grande force. Toute cette thèse je te la dédie mon bb!

À mon fils Joris Dimitri...

Abstract

Wireless sensor networks are used in many environmental monitoring applications (e.g., to monitor forest fires or volcanoes). In such applications, sensor nodes have a limited quantity of energy, but must operate for years without having their batteries changed. The main mechanism used to allow nodes to save energy is to sequence periods of activity and inactivity. However, the design of MAC and routing protocols for applications with low duty-cycle is still a challenge.

The MAC and routing protocols where the nodes synchronize their periods of activity and inactivity generally ensure fairness in energy consumption and allow to predict the network lifetime. However, the implementation of synchronization is complex and difficult to achieve when there are a large number of sensor nodes. Likewise, the synchronous protocols are not suitable for the low duty-cycle, because the synchronization overhead exceeds the communication cost. The MAC and routing protocols that do not need that the nodes are synchronized have the advantage to remove the synchronization complexity, but the protocols of the literature based on these assumptions do not generally guarantee fairness in energy consumption, as some nodes remain active longer than others.

In this thesis, we proposed unsynchronized MAC and routing protocols for wireless sensor networks devoted to environmental monitoring applications. The main specificity of our protocols is that they are adapted to very low duty-cycle (less than 1 % for all nodes).

Our protocols are analyzed and compared to existing protocols by simulation and experimentation on TelosB nodes. Despite this low duty-cycle for all nodes, our protocols are able to achieve good performance, unlike other protocols in the literature, which are not adapted to these extreme conditions.

Keywords: wireless sensor networks, MAC protocols, routing protocols, low duty cycle, energy efficient, environmental monitoring.

Résumé

Les réseaux de capteurs sans fil sont utilisés dans de nombreuses applications de surveillance de l'environnement (par exemple, pour surveiller les volcans ou pour détecter les incendies de forêts). Dans de telles applications, les nœuds capteurs disposent d'une quantité limitée d'énergie, mais doivent fonctionner pendant des années sans avoir leurs batteries changées. La principale méthode utilisée pour permettre aux nœuds d'économiser leur énergie est de séquencer les périodes d'activité et d'inactivité. Cependant, la conception de protocoles MAC et de routage pour les applications avec des taux d'activité faibles est un défi.

Les protocoles MAC et routage dans lesquels les nœuds synchronisent leurs périodes d'activité et d'inactivité assurent en général une équité dans la consommation énergétique et permettent de prédire la durée de vie du réseau. Mais, la mise en œuvre de la synchronisation est complexe et difficile à réaliser lorsque le nombre de nœuds capteurs est important. Aussi, les protocoles basés sur une synchronisation ne sont pas adaptés lorsque le taux d'activité des nœuds est trop faible, car le surcoût de la synchronisation dépasse le coût de communication. Les protocoles MAC et routage qui ne nécessitent pas que les nœuds soient synchronisés ont l'avantage d'éliminer la complexité de la synchronisation, mais les protocoles de la littérature basés sur ces hypothèses ne garantissent généralement pas l'équité dans la consommation d'énergie, car certains nœuds restent plus longtemps en activité que d'autres.

Dans cette thèse, nous avons proposé des protocoles MAC et routage non synchronisés pour les réseaux de capteurs sans fil dédiés à la surveillance environnementale. La spécificité principale de nos protocoles est qu'ils sont adaptés à de faibles taux d'activité (moins de 1% pour tous les nœuds).

Nos protocoles sont analysés et comparés aux protocoles existants par simulation et par expérimentation sur des nœuds TelosB. Malgré un taux d'activité faible pour tous les nœuds, nos protocoles sont capables d'obtenir de bonnes performances, contrairement aux autres protocoles de la littérature, qui ne sont pas adaptés à opérer avec de faibles taux d'activité.

Mots-clés : réseaux de capteurs sans fil, protocoles MAC, protocoles de routage, taux d'activité faible, économie d'énergie, surveillance environnementale.

Table des matières

Table des figures

Chapitre 1

Introduction

La nécessité de préserver l'environnement est devenue un véritable défi sociétal et économique. C'est ce qui explique la quantité importante de travaux visant les applications de surveillance avec des objectifs variés tels que la détection rapide de catastrophes comme les feux de forêt [YWM05], [HB07], les inondations [LKK+08], le suivi de la pollution des rivières [HUM+11], [HC12], ou encore la surveillance de grands édifices comme les ponts [NUF11], ou les bâtiments [MDZ+16], [MZD+16]. C'est aussi le cas pour la compréhension de phénomènes naturels comme les éruptions de volcans [WAJR+05] et les glissements de terrain [Ram10]. La préservation de l'environnement peut aussi se faire par une meilleure gestion de l'irrigation et une optimisation de l'utilisation des ressources (engrais, pesticides, etc.) en agriculture en vue d'augmenter la production agricole [Bag05], [BHB+08].

Cependant, de telles applications sont généralement déployées sur des zones étendues, non accessibles et sans infrastructure préexistante (ou qui nécessiteraient d'énormes coûts de réaménagement). De plus, elles doivent fonctionner pendant des mois (voire des années) sans intervention humaine.

Les réseaux de capteurs sans fil (notés RCSF) sont généralement utilisés pour mettre en œuvre de telles applications.

Réseaux de capteurs sans fil

Un RCSF est composé d'appareils à bas prix, autonomes en énergie, capables de surveiller les conditions physiques ou environnementales (température, humidité, bruit, vibration, pression, mouvement, pollution, etc.), d'effectuer certains calculs, et de collaborer pour transmettre leurs données *via* des liaisons sans fil à un destinataire principal qui se charge de recueillir ces données. Les RCSF comportent une (ou plusieurs) station(s) de base, appelée puits, qui agit comme une passerelle entre

les utilisateurs finaux et le RCSF. Un utilisateur final peut récupérer les données nécessaires à partir des informations recueillies par ce puits, puis les analyser afin de prédire ou anticiper certains phénomènes. La figure 1.1 montre un exemple de réseau de capteurs sans fil.

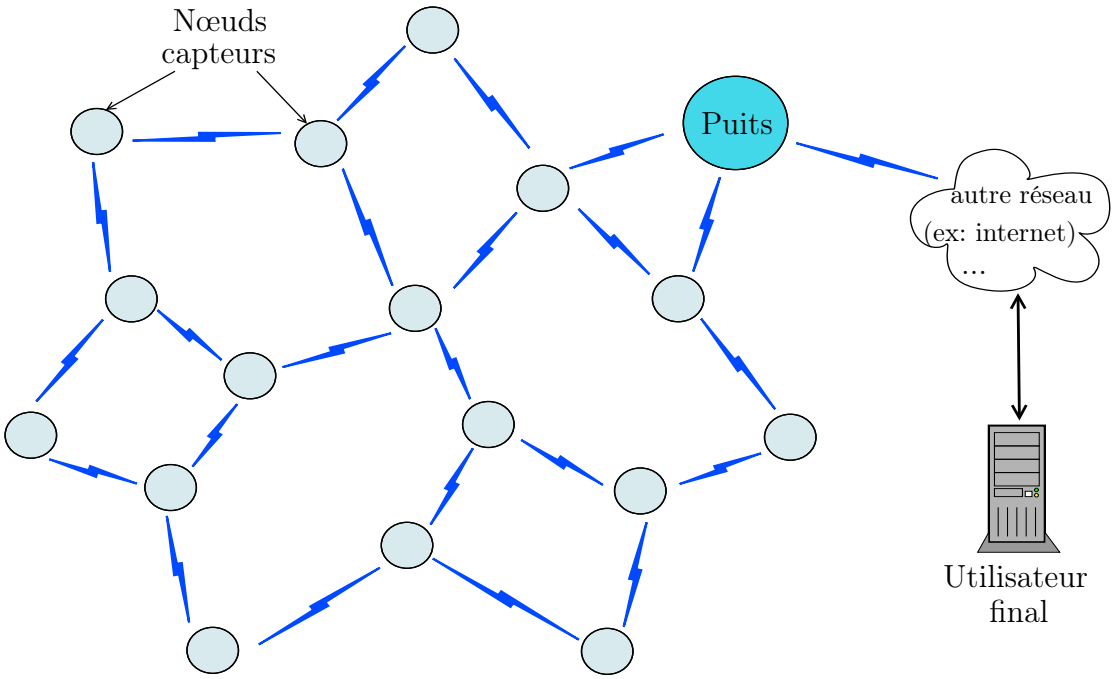


Figure 1.1 – Topologie d'un réseau de capteurs sans fil pour la surveillance environnementale.

Un nœud capteur sans fil est composé de quatre composants de base (voir figure 1.2) à savoir : l'unité d'acquisition, l'unité de traitement, l'unité de communication et l'unité source d'énergie. Un capteur peut également disposer, de certains dispositifs tel qu'une unité pilotant un ou des actionneurs ainsi qu'un système de localisation comme un GPS (pour *Global Positioning System*).

L'unité d'acquisition comprend deux sous-unités : le capteur et le convertisseur analogique-numérique (ADC pour *Analog to Digital Converter*). Le capteur est chargé de mesurer une grandeur physique. Le convertisseur analogique-numérique convertit les signaux analogiques en données numériques et les envoie à l'unité de traitement.

L'unité de traitement comporte l'unité de stockage et l'unité de calcul généralement un micro-contrôleur. L'unité de stockage permet de stocker les données environnementales recueillies. Ces données sont ensuite traitées par le microprocesseur.

L'unité de communication comporte un émetteur-récepteur radio (appelé module

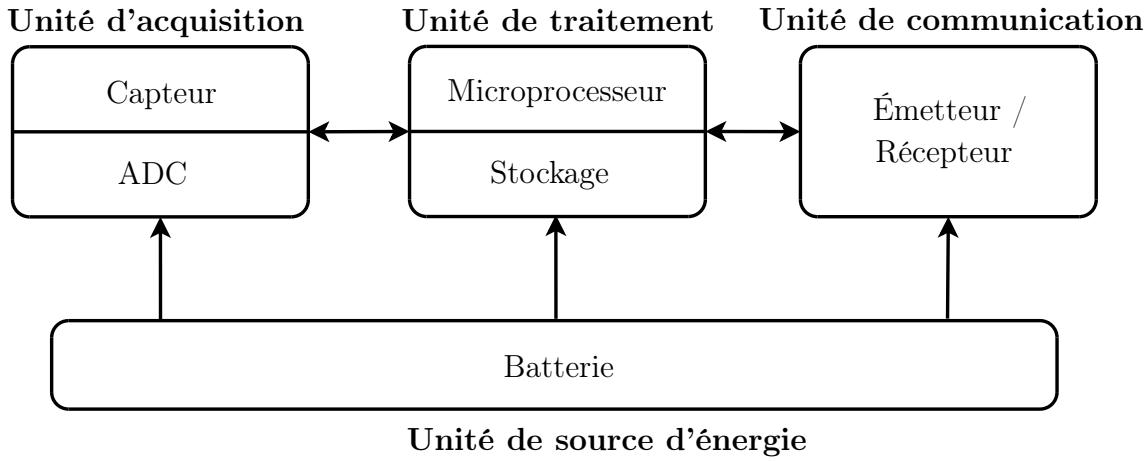


Figure 1.2 – Architecture d'un nœud capteur.

radio) qui se charge de transférer les données par une liaison sans fil *via* d'autres nœuds capteurs, si nécessaire, jusqu'au puits.

L'unité source d'énergie (piles ou une batterie) est souvent la seule source d'énergie qui fournit à toutes les autres unités, l'énergie nécessaire pour effectuer les tâches mentionnées. Notons que, des techniques alternatives d'alimentation (comme solaire, vibration, etc.) sont proposées dans la littérature [SET09]. Ces techniques supposent que les nœuds capteurs sont équipés de dispositifs qui permettront de collecter l'énergie de leur environnement. Toutefois, ces techniques ne garantissent pas une alimentation en continu des nœuds capteurs et restent très dépendantes de nombreux facteurs environnementaux [SET09], mais peuvent être utilisées en complément à toutes les solutions des RCSF qui économisent de l'énergie.

Avantages des RCSF

Les nœuds capteurs ont de nombreux avantages pour la surveillance environnementale : ils sont autonomes en énergie, ont un faible coût (et un faible coût de déploiement), sont faciles à déployer et ont peu d'impact sur l'environnement. Ils peuvent fonctionner aussitôt après déploiement et ne nécessitent aucune intervention humaine. Ce sont tous ces avantages des nœuds capteurs qui font du RCSF, un bon candidat pour réaliser les applications de surveillance environnementale.

Contraintes liées aux RCSF

Un RCSF subit néanmoins des contraintes qui lui sont propres et font de sa mise en œuvre un véritable challenge. Un RCSF doit faire face à des défis techniques venant des faibles ressources intrinsèques des nœuds capteurs. Les nœuds ont une quantité d'énergie limitée, une faible portée de communication, un débit limité, une capacité de mémoire et de stockage faibles et une puissance de calcul réduite. Une fois que les capteurs sont déployés, ils doivent être opérationnels pendant longtemps

sur des sites étendus sans infrastructure fixe, ce qui nécessite une gestion efficace de l'énergie (qui est limitée et souvent irremplaçable), une capacité d'auto-organisation pour faire face à l'évolution des liaisons radio, et une redondance dans la topologie pour compenser la défaillance de certains nœuds.

Problématique et contributions de ce travail

La problématique de ce travail consiste à concevoir des protocoles de communication robustes pour les applications de surveillance environnementale, qui font face à l'évolution des liaisons radio, qui gèrent efficacement l'énergie, qui nécessitent peu de configurations, et qui garantissent de bonnes performances en termes de délai et de taux de livraison des données pour un nombre important de nœuds capteurs.

Il est clair que l'énergie est la plus grande contrainte pour un RCSF, étant donné qu'il est difficile (parfois impossible) de remplacer les batteries (ou les piles) des nœuds capteurs dans les applications de surveillance environnementale.

La durée de vie du réseau dépend donc de l'utilisation de la batterie (ou des piles) des nœuds capteurs. La consommation énergétique d'un nœud capteur se repartit sur les trois unités que sont : l'unité de communication, l'unité de capture et l'unité de traitement (voir figure 1.2). Une étude réalisée par Hill et al. dans [HSW+00a] montre qu'un octet transmis dans un réseau de capteurs sans fil consomme autant d'énergie que l'exécution d'environ 1000 instructions par un microprocesseur. L'unité de communication est souvent plus coûteuse en énergie que l'unité de traitement et de capture des données dans les RCSF (puisque l'unité de capture n'est pas forcement alimentée de façon permanente).

La technique consistant à séquencer les périodes d'activité (pendant lesquelles le module radio est allumé) et de sommeil (pendant lesquelles le module radio est éteint) pour permettre aux nœuds d'économiser leur énergie s'est imposée. Elle est utilisée dans la norme IEEE 802.15.4 [IEE11] et dans la plupart des protocoles de la littérature. La proportion de la période active sur la durée totale du cycle (période active + période de sommeil) représente le taux d'activité. Un taux d'activité de 1 %, avec par exemple un cycle de 10 secondes signifie que les nœuds gardent leur radio éteinte durant 9.9 secondes et l'allument uniquement durant 0.1 s chaque 10 secondes. Dans l'état actif, un nœud est capable d'écouter la radio, de transmettre ou recevoir des données, mais dans l'état de sommeil, le nœud éteint complètement sa radio pour économiser l'énergie et ne peut ni recevoir, ni transmettre de données.

Notons que même si l'énergie consommée par un nœud pendant la période de sommeil est considérée négligeable par rapport à la consommation lorsqu'il est en activité, l'énergie consommée pour activer et désactiver le module radio est quant à elle non négligeable suivant le type de sommeil (de sommeil léger à sommeil profond)

et la fréquence de passage en mode sommeil [JRO10]. Par exemple un composant CC2420 met 2,4 ms à se réveiller lorsqu'il est en sommeil profond et 30 μs en sommeil léger [JRO10], alors que sur les composants Freescale le module radio met 310 μs et le microcontrôleur 235 μs à se réveiller [FVDBV12]. Aussi, la consommation d'énergie des radios (lorsqu'elles sont allumées) peut varier considérablement d'un composant à l'autre même lorsqu'ils sont dans un même état [YSH06]. Par conséquent, le taux d'activité peut être un bon indicateur pour évaluer la consommation énergétique des nœuds indépendamment des plate-formes comme dans les travaux antérieurs [BGAH06], [KHCL07], [SGJ08].

Une étude réalisée dans [LG09] montre par exemple qu'un composant CC2420 (comme un nœud TelosB ou MicaZ) épuisera ses deux piles (piles traditionnelles AA avec une capacité de 2700 mAh par pile) en quatre jours environ si son taux d'activité est 100 %, et peut prolonger sa durée de vie jusqu'à environ un an avec un taux d'activité de 1 %.

Plusieurs protocoles MAC (pour *Medium Access Control*) et routage économes en énergie pour les RCSF proposés dans la littérature sont basés sur le mécanisme à faible taux d'activité. Les protocoles MAC avec un taux d'activité faible sont soit synchrones, soit asynchrones. Les protocoles MAC synchrones génèrent un nombre important de messages de contrôle pour réaliser la synchronisation avant l'échange des données, ce qui limite leur utilité quand la durée de l'activité des nœuds est très courte. Les protocoles MAC asynchrones ne nécessitent pas de synchronisation mais entraînent généralement de grands délais (dus à des périodes d'activité communes peu fréquentes entre les nœuds voisins) et certains nœuds restent actifs beaucoup plus longtemps que d'autres.

Les protocoles de routage pour les RCSF avec un taux d'activité faible doivent faire face à des liens de voisinage qui peuvent être dynamiques et aussi aux contraintes liées au protocole MAC utilisé.

Les contributions principales de ce travail visent le contexte des réseaux de capteurs sans fil dédiés aux applications de surveillance environnementale telles que la surveillance de volcans ou de forêts. Dans ce type d'application le réseau doit être opérationnel quand un événement critique survient. Ce qui fait que pour permettre aux nœuds d'économiser plus d'énergie (afin d'assurer une longue durée de vie au réseau), deux phases de surveillance s'avèrent nécessaires. Une phase d'attente pendant laquelle les nœuds doivent opérer en étant très économes en énergie et une phase de travail intensif pendant laquelle les nœuds doivent être très réactifs. Nos propositions portent sur la sous-couche MAC et la couche réseau. Nous proposons des protocoles MAC non synchronisés pour les réseaux de capteurs sans fil dédiés à

ce type de surveillance environnementale et des protocoles de routage adaptés. La spécificité principale de nos protocoles est que certains sont adaptés à de faibles taux d'activité (moins de 1 % pour tous les nœuds (pour la phase d'attente)) et d'autres sont adaptés à de grands taux d'activité (plus de 1 % pour tous les nœuds (pour la phase de travail intensif)).

Organisation du manuscrit

Ce document est organisé comme suit. Dans le chapitre 2, nous présentons les principaux protocoles MAC et protocoles de routage existants qui fonctionnent avec un faible taux d'activité. Nous montrons leurs difficultés à être appliqués avec des taux d'activité inférieure à 1 %. Le chapitre 3 décrit nos différentes contributions. Nous décrivons d'abord notre proposition de protocoles MAC asynchrones, aveugles, et à rencontres opportunistes, ainsi que les améliorations apportées en vue d'augmenter ses performances. Ensuite, nous décrivons nos propositions de protocoles de routage par gradient et par inondation, basés sur notre protocole MAC. Le chapitre 4 présente la validation par simulation et par expérimentation de nos protocoles. Nous évaluons par simulation nos protocoles MAC et routage par comparaison avec l'existant, puis nous réalisons des tests sur des nœuds TelosB pour prouver que nos protocoles fonctionnent sur de réels nœuds capteurs. Enfin, dans le chapitre 5 nous concluons nos travaux et donnons les différentes orientations offertes pour étendre la portée de cette thèse.

Chapitre 2

État de l'art

Dans ce chapitre, nous décrivons les principaux protocoles pour les couches 2 et 3 du modèle OSI (pour *Open Systems Interconnection*) [Zim80], économes en énergie proposés dans la littérature pour les réseaux de capteurs sans fil. Comme nous l'avons dit précédemment, le composant radio est souvent considéré comme le composant le plus consommateur en terme d'énergie [LG09] par exemple sur les nœuds TelosB ou MicaZ. Ainsi, pour permettre aux nœuds capteurs de conserver leur ressource énergétique (qui est très souvent limitée), la principale technique utilisée dans la littérature est de garder le composant radio éteint le plus souvent possible. Ce mécanisme est réalisé en séquençant les périodes d'activité (pendant lesquelles les nœuds ont leur composant radio allumé) et d'inactivité (pendant lesquelles les nœuds ont leur composant radio éteint pour économiser de l'énergie). Nous décrivons particulièrement les protocoles basés sur ce mécanisme.

2.1 Protocoles MAC pour les réseaux de capteurs sans fil

Les protocoles MAC pour les réseaux de capteurs sans fil dans lesquels les nœuds ont une séquence de période d'activité et d'inactivité sont appelés protocoles à taux d'activité. Le taux d'activité représente le pourcentage du temps passé en période d'activité. On peut classer les protocoles à taux d'activité en deux grandes catégories suivant le séquencement des périodes d'activité et d'inactivité des nœuds. La première catégorie est celle des protocoles MAC synchrones, dans laquelle tous les nœuds ont leur période d'activité simultanément ou planifiée. La deuxième catégorie est celle

des protocoles MAC asynchrones, dans laquelle les nœuds ont des périodes d'activité indépendantes.

2.1.1 Protocoles MAC synchrones

Dans les protocoles MAC synchrones, les nœuds sont synchronisés et partagent un calendrier commun pour leurs périodes d'activité et d'inactivité. Dans cette partie, nous décrivons tout d'abord le principal protocole MAC synchrone de la littérature, qui est décrit dans la norme IEEE 802.15.4 en mode avec suivi de balises, puis d'autres protocoles MAC synchrones.

2.1.1.1 Norme IEEE 802.15.4 en mode avec suivi de balises

Les réseaux sans fil personnels à faible débit ont fait l'objet d'une norme (IEEE 802.15.4 [IEE03]) qui sert de base à de nombreuses solutions. La norme IEEE 802.15.4 [IEE03] spécifie une couche physique (notée PHY) et une sous-couche de contrôle d'accès au médium (notée MAC) pour les réseaux sans fil à faible débit notés LR-WPANs (pour *Low-Rate Wireless Personal Area Networks*). La dernière version de cette norme, présentée dans [IEE11], prend en charge une variété de caractéristiques physiques permettant son adoption dans de nombreux types d'applications. Récemment, certains amendements ont été apportés à la norme dans IEEE 802.15.4e [IEE12] en vue d'améliorer les fonctionnalités de contrôle d'accès au médium pour soutenir les applications industrielles ayant des exigences en qualité de service particulières. Dans la suite de cette partie, nous faisons une description des topologies supportées, d'une couche physique et de la sous-couche MAC de la norme IEEE 802.15.4 [IEE11].

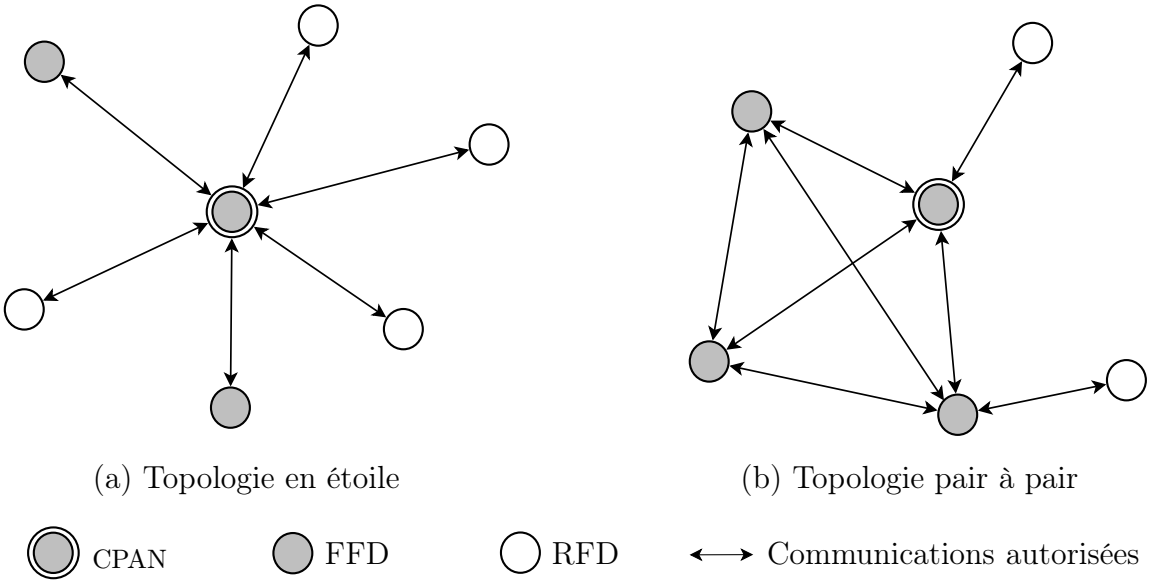
Description des topologies

Un réseau IEEE 802.15.4 est composé de nœuds FFD (pour *Full-Function Devices*) et de nœuds RFD (pour *Reduced-Function Devices*). Un FFD peut communiquer avec des RFD et des FFD, mais un RFD ne peut communiquer qu'à un seul FFD. En général, les FFD ont plus de ressources que les RFD.

Un FFD peut avoir la fonction de coordinateur de PAN (pour *Personal Area Network)*, de coordinateur ou de feuille, tandis qu'un RFD ne peut être qu'une feuille. Le coordinateur de PAN (noté CPAN) est l'initiateur et le gestionnaire principal du PAN. Un coordinateur est un périphérique chargé de gérer un ou plusieurs autres coordinateurs et des feuilles. Les feuilles sont des périphériques d'extrémité du réseau sans fonctionnalités de routage. Une feuille peut être un RFD ou un FFD. Toutes ces entités sont capables de supporter 0 à N nœuds capteurs.

La norme spécifie deux types de topologies : les topologies en étoile (voir fi-

gure 2.1(a)) et les topologies pair à pair (voir figure 2.1(b)). Dans la topologie en étoile, les communications sont effectuées uniquement entre le CPAN et les feuilles. Dans la topologie pair à pair, tout coordinateur peut communiquer avec n'importe quel autre coordinateur du moment où ils sont à portée respective de communication.



(a) Topologie en étoile (b) Topologie pair à pair

Figure 2.1 – Topologies de la norme IEEE 802.15.4.

Couche physique de la norme IEEE 802.15.4

La couche physique assure l'activation et la désactivation du *transceiver* radio, la sélection du canal de communication, le test de l'occupation du canal noté CCA (pour *Clear Channel Assessment*), la détection d'énergie notée ED (pour *Energy Detection*) sur un canal, l'indication de la qualité d'un lien noté LQI (pour *Link Quality Indicator*), ainsi que la transmission et la réception des bits à travers le canal.

Sous-couche MAC de la norme IEEE 802.15.4

La sous-couche MAC de la norme utilise l'accès multiple à détection de la porteuse avec évitement de collision, noté CSMA/CA (pour *Carrier Sense Multiple Access with Collision Avoidance*), comme méthode d'accès au canal. Le protocole MAC de la norme IEEE 802.15.4 prend en charge deux modes de fonctionnement pour l'accès au canal : un mode avec suivi de balises et un mode sans suivi de balises. Dans cette partie, nous détaillons le mode avec suivi de balises dans lequel tous les nœuds ont des séquences d'activités et d'inactivités synchronisées. Le mode sans suivi de balises sera décrit dans la partie 2.1.2.

CSMA/CA en mode avec suivi de balises

Le mode avec suivi de balises du CSMA/CA utilise des trames balises transmises par le CPAN pour délimiter une structure temporelle appelée supertrame. La supertrame commence avec l'envoi ou la réception d'une balise et sa structure est décrite par les valeurs de BO (pour *macBeaconOrder*) et SO (pour *macSuperframeOrder*). BO et SO définissent respectivement l'intervalle entre deux balises, qui représente le cycle noté BI (pour *Beacon Interval*), et la durée de la supertrame, notée SD (pour *Superframe Duration*). SD représente la longueur de la partie active de la supertrame. Le taux d'activité est égal à SD/BI. BI et SD sont définis comme suit :

$$\begin{cases} BI = aBaseSuperframeDuration \times 2^{BO}, \\ SD = aBaseSuperframeDuration \times 2^{SO}, \end{cases}$$

où $aBaseSuperframeDuration = 15.36$ ms et $0 \leq SO \leq BO \leq 14$.

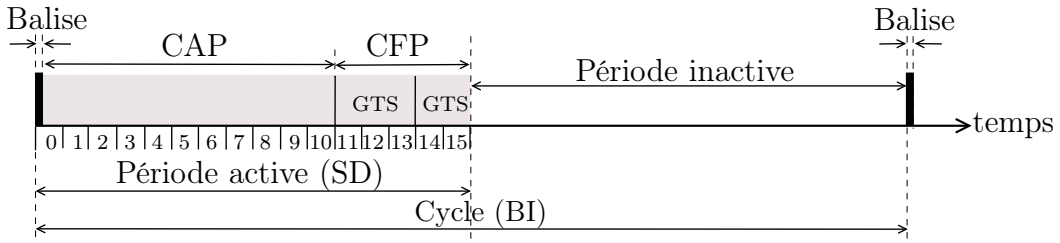


Figure 2.2 – Exemple de structure de supertrame dans la norme IEEE 802.15.4.

La figure 2.2 illustre une structure de supertrame de la norme 802.15.4. Elle peut comporter une période active et une période inactive. Durant la période inactive tous les nœuds (y compris le coordinateur) peuvent être en mode sommeil. La période active de chaque supertrame est divisée en seize intervalles de temps égaux et se compose de trois parties : la partie de la balise, la partie dans laquelle les nœuds accèdent au médium avec contention, notée CAP (pour *Contention Access Period*), et une autre partie avec un accès sans contention, notée CFP (pour *Contention Free Period*). Pendant la CAP, les nœuds accèdent au médium par compétition suivant l'algorithme CSMA-CA slotté.

Pendant la CFP, le coordinateur attribue des intervalles de temps garantis notés GTS (pour *Guaranteed Time Slot*), et les nœuds ayant obtenu un GTS pendant la CAP peuvent envoyer directement leurs données sans d'autres précautions. Le CPAN peut allouer jusqu'à sept GTS (à condition qu'il existe une capacité suffisante dans la supertrame) et chaque GTS peut occuper un ou plus d'un intervalle de temps.

Algorithme CSMA/CA slotté

L'algorithme CSMA/CA slotté de la norme 802.15.4 maintient trois variables pour chaque tentative de transmission : NB, CW et BE. NB (pour *Number of Backoffs*) représente le nombre de tentatives d'accès pour la trame en cours et est initialisé

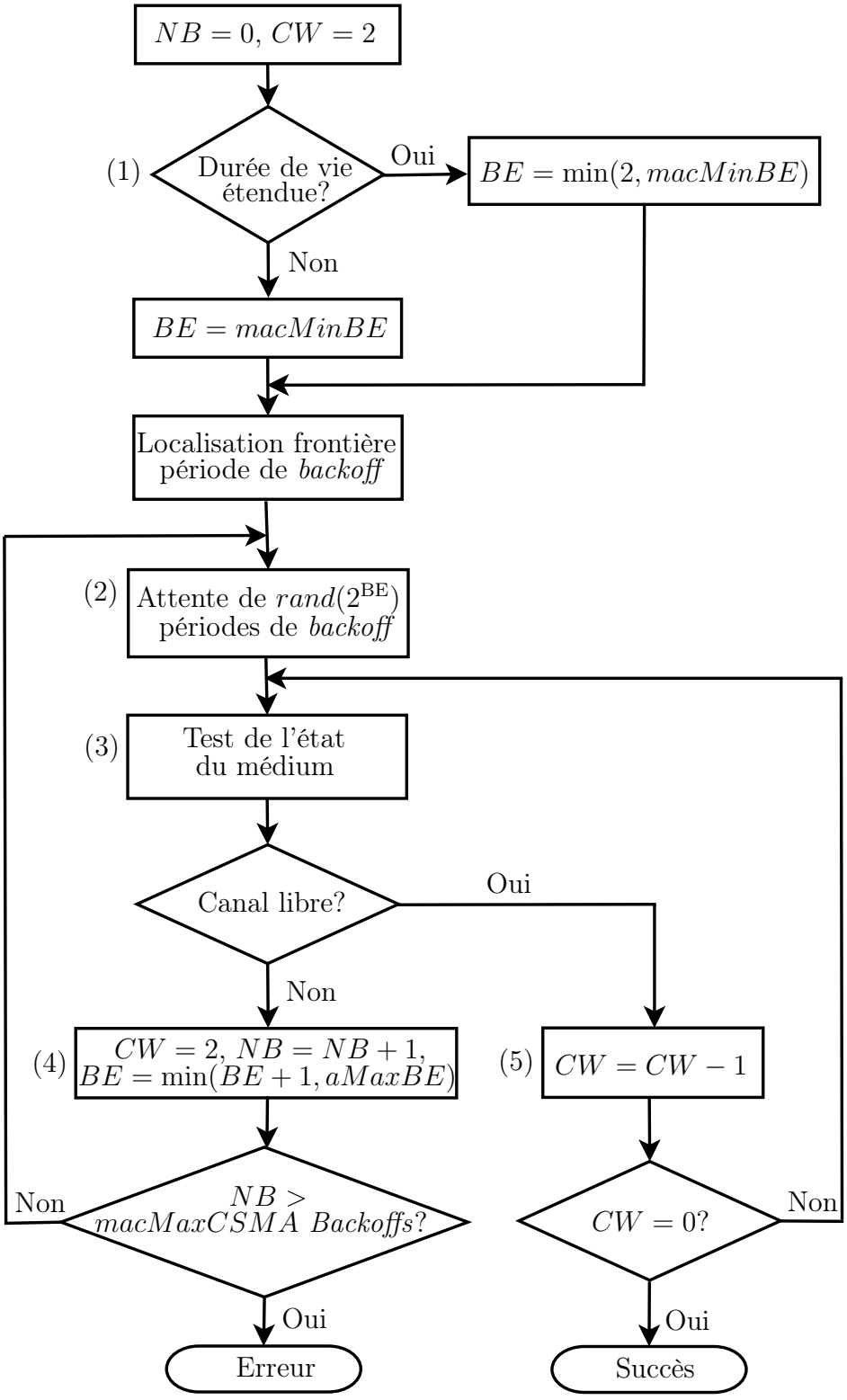


Figure 2.3 – Diagramme de l'algorithme CSMA/CA slotté.

à zéro. CW (pour *Contention Window*) représente le nombre de tests de canaux effectués pour déterminer que le canal est libre pour transmettre. CW est initialisé à 2 avant chaque tentative de transmission. BE (pour *Backoff Exponent*) permet de déterminer la fenêtre dans laquelle la durée du *backoff* est tiré aléatoirement. La figure 2.3 décrit les étapes de l'algorithme CSMA/CA slotté. Dans cet algorithme, le temps est divisé en périodes de temps appelées *backoff* de 320 μs et chaque opération est réalisée à la frontière d'un *backoff*. Après l'initialisation des paramètres CW (à 2) et NB (à 0), la sous-couche MAC vérifie si l'indicateur de l'option économie d'énergie est fixée à zéro. Si oui, BE est initialisé au minimum entre 2 et $macMinBE$, sinon, BE prend la valeur de $macMinBE$ (qui est égale à 3 par défaut). Après l'initialisation de BE, la sous-couche MAC localise la frontière du prochain *backoff* (étape (1)). Ensuite, un temps d'attente aléatoire est tirée. Il est calculé dans l'intervalle $[0; 2^{\mathrm{BE}} - 1]$. Après ce temps d'attente aléatoire, la sous-couche MAC vérifie si le temps restant dans la CAP est suffisant pour faire la transmission (étape (2)). Si ce n'est pas le cas, la sous-couche MAC interrompt le compte à rebours des *backoffs* à la fin de la CAP et le reprend au début de la CAP de la supertrame suivante. Si c'est le cas, la sous-couche MAC attend le *backoff* calculé, puis demande à la couche physique d'effectuer deux CCA. Si le canal est occupé pendant au moins un des CCA (étape (4)), la sous-couche MAC incrémente NB et BE de 1 (tout en s'assurant que BE ne soit pas supérieur à *aMaxBE* fixé à 5) et réinitialise CW à 2. Si le canal est libre les deux fois (étape (5)), la sous-couche MAC commence la transmission de la trame à la frontière de la période de *backoff* suivante. Il faut noter que si la valeur de NB est inférieure ou égale à $macMaxCSMABackoffs$ (fixé à 4), l'algorithme CSMA/CA revient à l'étape (2). Si ce n'est pas le cas, l'algorithme CSMA/CA slotté s'arrête et renvoie un échec d'accès au canal.

2.1.1.2 Autres protocoles MAC synchrones

Il existe de nombreux protocoles MAC synchrones comme par exemple S-MAC [YHE02], T-MAC [YHE04], D-MAC [LKR04], Q-MAC [VA06], R-MAC [DSJ07], LO-MAC [NJY13] et [KJK15]. Dans ces protocoles, les nœuds s'accordent sur un même instant de réveil. Le premier nœud qui diffuse son calendrier devient le synchroniseur.

Dans S-MAC (pour *sensor-MAC*) [YHE02], les nœuds commutent entre périodes d'activités et périodes d'inactivité suivant un taux d'activité fixe et se synchronisent à l'aide d'une trame SYNC. Les nœuds maintiennent une table avec les instants de réveil de leurs voisins. Lorsqu'un nœud est actif, il écoute le canal pendant un laps de temps. S'il entend une trame SYNC, il se synchronise sur le calendrier du nœud

émetteur de cette trame SYNC. S'il n'entend aucune trame après ce délai, il adopte son propre calendrier et le diffuse à son voisinage. Tous les nœuds qui suivent ce calendrier forment un *cluster* virtuel. Quand un nœud a une trame à émettre, il utilise un mécanisme de réservation de type RTS/CTS (pour *Request To Send/ Clear To Send*) et ACK (pour *Acknowledgement*) inspiré de la norme IEEE 802.11 [sc+99], comme le montre la figure 2.4. S-MAC permet aux nœuds de passer périodiquement en mode sommeil pour économiser de l'énergie. Cependant, le principal problème avec S-MAC est le nombre important de messages de contrôle générés par les trames SYNC, RTS, CTS et ACK. Un nœud peut être aussi amené à adopter plusieurs calendriers différents, donc avoir un temps d'activité plus élevé que les autres nœuds et une consommation d'énergie plus importante. De plus, la latence de bout en bout des données dans S-MAC peut être très élevée dans un réseau multi-saut car une trame ne parcourt qu'un seul saut par cycle (voir figure 2.4).

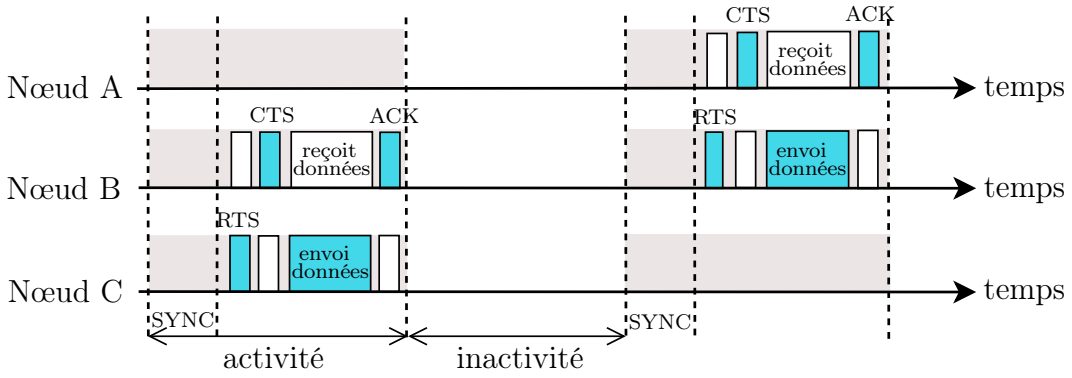


Figure 2.4 – Séquence des périodes d'activité et d'inactivité dans S-MAC.

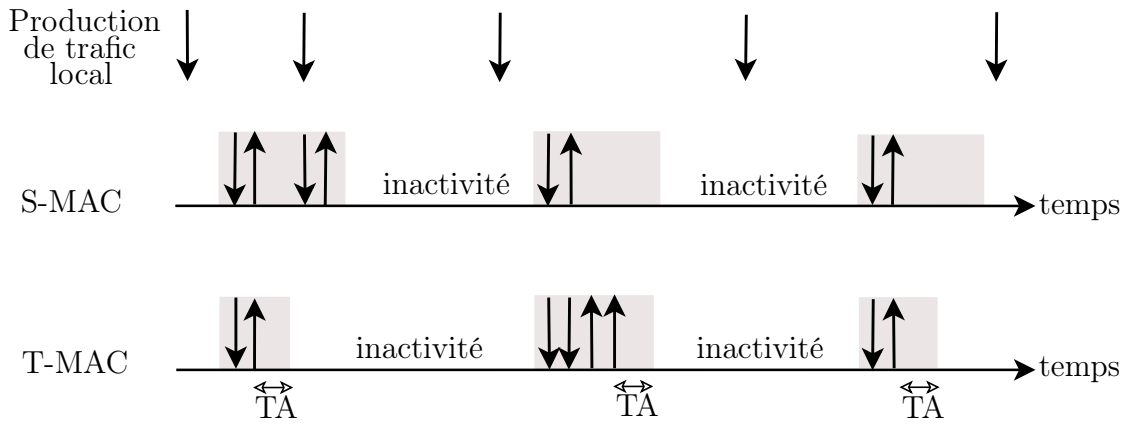


Figure 2.5 – Comparaison des périodes d'activité et de sommeil entre T-MAC et S-MAC.

T-MAC (pour *Timeout-MAC*) [YHE04], est une amélioration de S-MAC. Contrairement à S-MAC, T-MAC ne fonctionne pas à taux d'activité fixe. Il permet aux

nœuds de passer en mode sommeil s'ils ne détectent pas d'activité après un temps TA, comme le montre la figure 2.5. L'intervalle de temps TA est supérieur au temps maximum pour envoyer une trame avec contention en utilisant le mécanisme RTS/CTS. T-MAC évite que les nœuds écoutent inutilement pendant toute la période d'activité (écoute de SYNC, RTS, CTS et transmission de données) comme dans S-MAC, ce qui lui permet d'économiser de l'énergie par rapport à S-MAC.
Selon l'étude réalisée dans [DDBB14], la performance de T-MAC dans les applications de surveillance est supérieure à celle de S-MAC quand il y a une fluctuation du trafic. En revanche, l'ajustement dynamique du temps d'activité peut générer un problème d'endormissement précoce. Par exemple, sur la figure 2.5, lors du premier cycle, le nœud dans T-MAC passe en mode sommeil après l'intervalle de temps TA et juste après une trame est disponible pour lui. Il devra donc attendre le prochain cycle avant de recevoir cette trame. Ce qui augmente le temps de latence pour la livraison des données.

Dans R-MAC (pour *Routing-enhanced duty cycle MAC*) [DSJ07], un cycle de fonctionnement d'un nœud capteur est divisé en trois étapes : SYNC, DATA et SLEEP, comme le montre la figure 2.6. R-MAC suppose qu'un protocole distinct s'occupe de la synchronisation des horloges des nœuds pendant la période SYNC. Quand un nœud a une trame de données à envoyer à un destinataire qui est po-

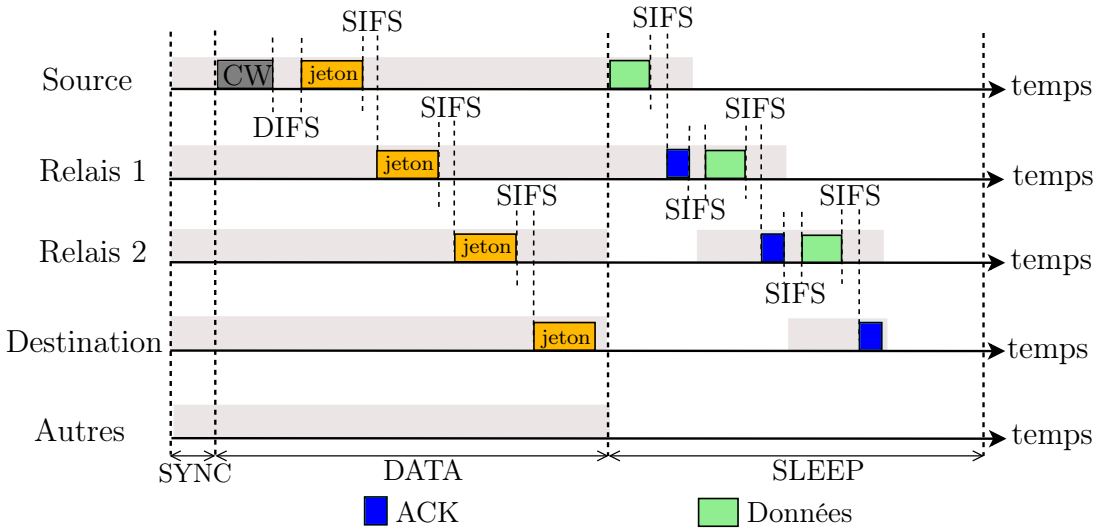


Figure 2.6 – Aperçu d'un cycle dans R-MAC.

sitionné à plusieurs sauts, une trame de contrôle spéciale appelée jeton est utilisée pendant la période DATA pour initier la communication à la place d'une paire de trame RTS/CTS (comme dans 802.11). Ce jeton est utilisé à la fois pour confirmer la réception d'un jeton du nœud en amont, et pour demander la communication avec un nœud en aval sur le chemin. Par exemple sur la figure 2.6 le nœud *Source* a une trame

de données à envoyer au nœud *Destination*, le nœud initie sa demande au début de la période DATA. Il écoute le canal pendant un temps CW (pour *Contention Window*), si le canal est libre, le nœud attend un temps DIFS (pour *Distributed Coordination Function) InterFrame Space* comme dans 802.11) après lequel il envoie une trame jeton le long du chemin de transmission de données. Ce jeton comporte tous les champs comme dans un RTS, tels que l'adresse du nœud courant, l'adresse du saut suivant, et la durée de la transmission. Lorsque *Relais1* reçoit la trame jeton, il la retransmet au *Relais2* après un temps SIFS (pour *Short InterFrame Space* comme dans 802.11) à la fois pour confirmer au nœud en amont (*Source*) de la réception du jeton, et pour demander la communication avec le prochain saut (*Relais2*). Le processus est répété jusqu'à ce que la destination finale reçoive le jeton. La période DATA est utilisée uniquement pour envoyer et recevoir des trames jeton. Les trames de données sont transmises et reçues uniquement pendant la période SLEEP. Ainsi, chaque nœud qui a envoyé ou relayé un jeton doit se réveiller à un instant spécifique de la période SLEEP pour recevoir et retransmettre une trame de données. Tous les autres nœuds éteignent leur module radio pendant la période SLEEP pour économiser de l'énergie. Le nœud *Source* reste actif jusqu'au début de la période SLEEP pour transmettre la trame de données. Le nœud suivant (*Relais1*) reste actif lui aussi pour recevoir la trame de données. Lorsque *Relais1* reçoit la trame, il envoie une trame ACK au nœud *Source* pour accusé réception. Après la réception du ACK le nœud *Source* peut passer en mode sommeil pour économiser de l'énergie. Ce processus de relais se poursuit de saut en saut jusqu'à ce que la trame atteigne la destination finale.
R-MAC peut ainsi transmettre une trame de données sur plusieurs sauts en un seul cycle. Dès lors, le nombre de sauts avec lequel une trame peut être transmise dépend de la durée du cycle. Une longue durée de cycle (pour un grand réseau) devient une source de gaspillage d'énergie lorsqu'il y a peu de trafic. Aussi, l'objectif de réduire le temps de latence peut être complètement remis en cause en cas d'erreurs de transmission de trames de données ou de collisions entre deux jetons.

LO-MAC (pour *Low Overhead* MAC) [NJY13] a été proposé pour améliorer les performances de R-MAC. LO-MAC propose d'ajouter une période CS (pour *Carrier Sensing*) juste avant la période DATA. Pendant la période CS, un nœud écoute le canal pour voir s'il y a une activité en cours. Si c'est le cas, il reste actif pour éventuellement participer à l'acheminement des données à plusieurs sauts. Sinon, le nœud passe immédiatement en mode d'inactivité. Cela permet à LO-MAC d'économiser plus d'énergie par rapport à R-MAC.

La proposition [KJK15] améliore LO-MAC et R-MAC en terme d'économie d'énergie. Dans les protocoles R-MAC et LO-MAC, les nœuds sources et relais res-

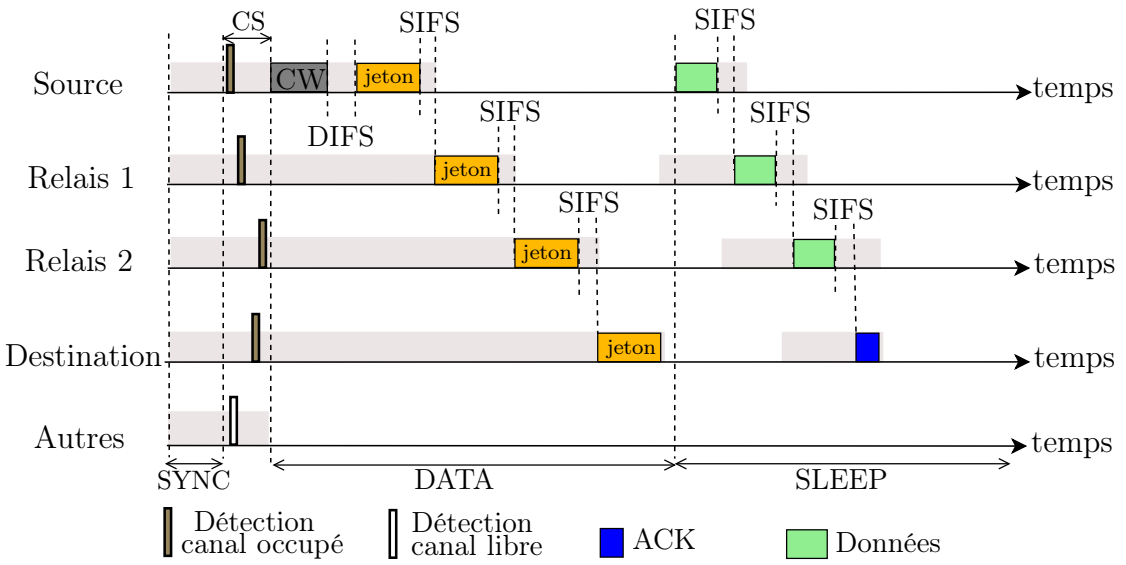


Figure 2.7 – Aperçu d'un cycle dans le protocole de [KJK15].

tent actifs inutilement pendant la période DATA après la transmission d'un jeton et sa retransmission par le nœud suivant. Les auteurs proposent que les nœuds passent en mode sommeil après la retransmission du jeton comme indiqué sur la figure 2.7. Aussi, les auteurs proposent de se servir de l'écoute de la retransmission d'une trame comme accusé de bonne réception, au lieu d'attendre un ACK. Ainsi, seul le destinataire final génère un acquittement. Ces mécanismes permettent de réduire le temps de transmission des données et la période pendant laquelle les nœuds gardent leur module radio allumé, ce qui réduit la consommation en énergie du protocole [KJK15] par rapport à LO-MAC et R-MAC.

Dans D-MAC (pour *Data-gathering MAC*) [LKR07], les nœuds se synchronisent suivant une topologie arborescente en fonction de leur position par rapport au puits. Chaque nœud décale son réveil d'un temps $d.\mu$ par rapport au calendrier du puits selon sa profondeur d dans l'arbre comme le montre la figure 2.8. Les nœuds intermédiaires ont leur intervalle de temps d'envoi immédiatement après leur intervalle de temps de réception. Le paramètre μ est une constante qui représente le temps nécessaire pour transmettre ou recevoir une trame. Le fait d'ordonnancer les périodes d'activités des nœuds fait que D-MAC conduit à un faible délai de bout en bout. Cependant, D-MAC manque de souplesse dans le sens qu'il ne peut fonctionner que pour des communications vers le puits et pour des réseaux où les positions des nœuds restent statiques, car ses performances en terme de délai peuvent être mauvaises si le réseau est dynamique. Aussi, D-MAC ne prévoit pas de périodes communes pour favoriser la diffusion.

Dans TreeMAC [SHSL09], les nœuds se synchronisent aussi suivant un arbre. Tree-

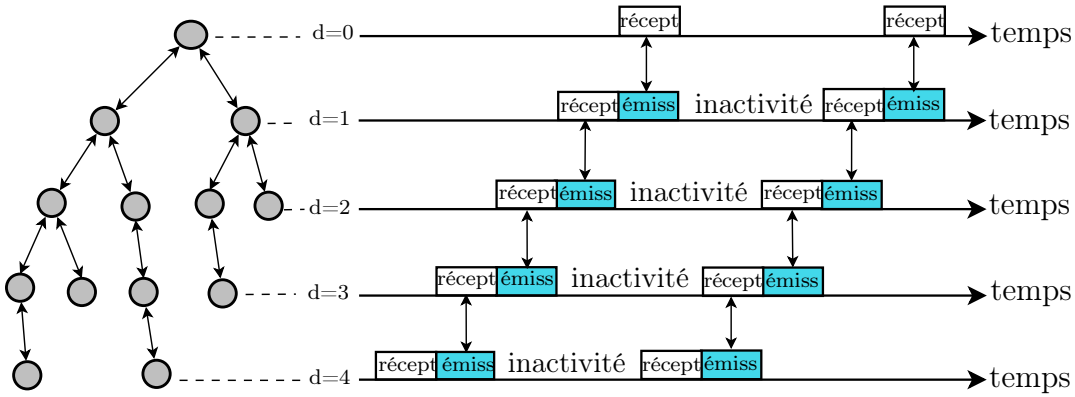


Figure 2.8 – Arbre de collecte de données dans D-MAC.

MAC propose un algorithme d'attribution d'intervalles de temps différents (appelés *slots*) aux nœuds pour éliminer les interférences dans les communications horizontales et verticales à profondeur deux dans l'arbre en vue de maximiser le débit jusqu'au puits. Le temps est divisé en cycles et chaque cycle est divisé en plusieurs intervalles de trois *slots*. Chaque nœud père dans l'arbre attribue une séquence d'intervalles à chacun de ses enfants proportionnellement à leur charge de trafic. L'emplacement du *slot* d'envoi à l'intérieur des intervalles affectés est choisi par les nœuds de telle sorte que les nœuds situés à profondeur deux en verticale n'aient pas un *slot* d'envoi identique.
L'utilisation de trois *slots* différents dans TreeMAC permet à un nœud situé à une profondeur n d'éviter les conflits d'accès au médium avec ses voisins à une profondeur $n+1$ ou $n-1$. Aussi, TreeMAC évite le gaspillage de *slots* attribués à des nœuds alors qu'ils n'ont pas de trames à transmettre. Cependant, TreeMAC nécessite une structure d'arbre relativement stable pour permettre à deux nœuds positionnés à une différence de profondeur 2 de ne jamais choisir le même numéro de *slot*. Cette rigidité limite son utilisation dans les conditions dynamiques. Une étude réalisée dans [OH12] montre que TreeMAC gaspille un nombre de *slots* de temps valant deux fois le nombre d'enfants et de petits enfants du puits.

Q-MAC (pour *Query MAC*) [VA06] propose des calendriers d'activité décalés pour faire se chevaucher le réveil des nœuds aux sauts n et $n-1$ afin d'assurer une faible latence comme dans D-MAC. Contrairement à D-MAC, le décalage de l'activité des nœuds dans Q-MAC favorise un trafic dans le sens descendant (de la racine vers les feuilles), pour permettre au puits d'envoyer des requêtes aux nœuds du réseau. Q-MAC propose également deux mécanismes pour favoriser le trafic montant (d'un nœud feuille vers la racine) en fonction de si le puits connaît la profondeur du chemin (en terme de nombre de saut) vers des sources. Si c'est le cas, un calendrier

échelonné est conçu pour permettre aux nœuds intermédiaires de se réveiller à temps pour recevoir les données et les passer aux nœuds voisins suivant la liaison montante jusqu'au puits. Sinon, chaque nœud intermédiaire reste actif une fois qu'il reçoit la requête du puits jusqu'à la transmission de la trame de données. Q-MAC réduit le temps de latence de bout en bout et permet une communication dans le sens montant et descendant. Cependant, son efficacité en terme d'économie d'énergie reste discutable lorsque les chemins ne sont pas connus à l'avance par le puits, car les nœuds intermédiaires restent longtemps actifs avant la réception et la retransmission des données.

Dans tous ces protocoles MAC, la communication entre les nœuds synchronisés est aisée car les activités des nœuds se chevauchent pendant leur période d'activité. Cependant, le fait d'avoir plusieurs nœuds actifs en même temps augmente la contention et les collisions, et donc le gaspillage d'énergie. De plus, le problème principal dans ces protocoles réside dans la complexité de la mise en œuvre de cette synchronisation quand il y a un nombre important de nœuds et dans le coût généré en terme de messages de contrôle pour mettre en œuvre et maintenir la synchronisation.

Dans la partie suivante, nous mettons en exergue le coût lié à la synchronisation en prenant pour exemple un protocole MAC synchronisé développé dans notre équipe.

2.1.1.3 Considération du coût énergétique sur un exemple

Prenons comme exemple le protocole MAC synchronisé MaCARI [CGM08], [CLG+09], [Cha09], développé dans notre équipe de recherche. La figure 2.9 montre un cycle global dans MaCARI. Un cycle dans MaCARI est divisé en trois périodes : une période de synchronisation $[T_0; T_1]$, une période d'activité $[T_1; T_3]$ et une période d'inactivité ou de sommeil $[T_3;T_0]$. Nous nous intéressons particulièrement à la période de synchronisation (c'est-à-dire à l'intervalle $[T_0; T_1]$).

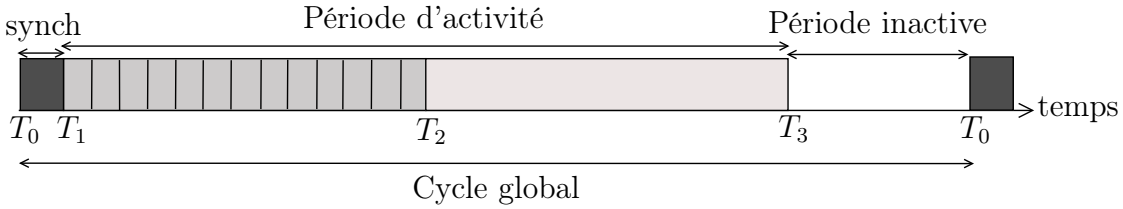


Figure 2.9 – Cycle global dans MaCARI.

La figure 2.10 illustre le mécanisme de synchronisation de MaCARI sur un réseau en arbre constitué de sept coordinateurs qui utilisent une cascade de *beacons* dans MaCARI . Par mesure de simplification, les feuilles n'ont pas été représentées sur cette

figure, mais à chaque coordinateur est associé un ensemble de feuilles (représentées par a_i, b_i, c_i, d_i, e_i, f_i et g_i).

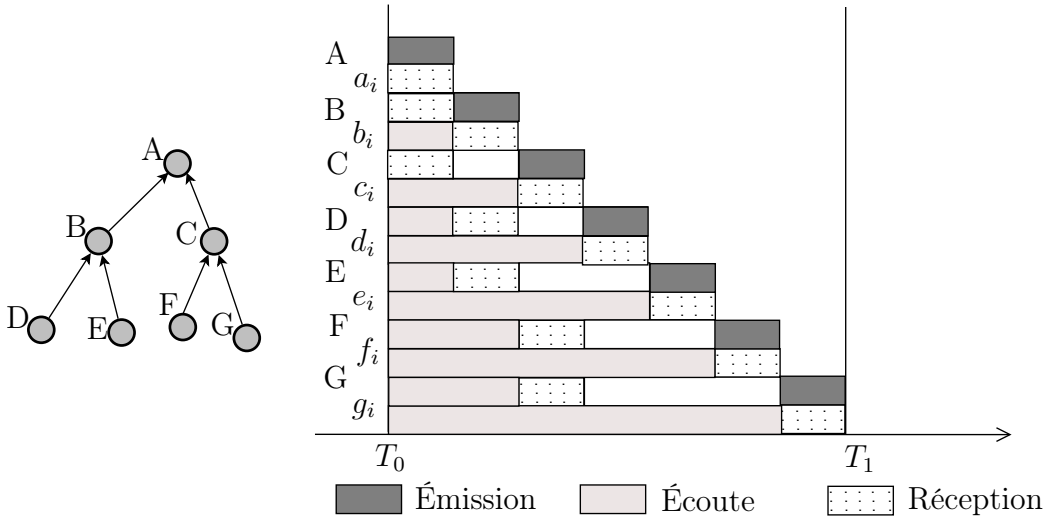


Figure 2.10 – Un exemple de cascade de *beacons*.

Le nœud A qui représente le CPAN commence la cascade et diffuse d'abord le premier *beacon* contenant l'ordre de synchronisation (B, C, D, E, F, G). Après la réception du *beacon* B sait qu'il est le suivant à transmettre le *beacon* et C sait qu'il doit attendre que B finisse la transmission. Ce processus est répété jusqu'à ce que toutes les entités du réseau reçoivent le *beacon* et soient synchronisées à partir de la date T_1.

On peut estimer le coût en terme de messages de contrôle pour mettre en œuvre la synchronisation. Ce coût est de n (avec n le nombre de coordinateurs) trames de contrôle à chaque cycle global.

Les tests réalisés sur des cartes B2400ZB-tiny [tel] avec un module radio Chipcon CC2420 [Ins06] et un microprocesseur Motorola MC9S08GT60A [mot] ont permis d'estimer le temps T_{synch} comme suit : $T_{synch} = n \times (8 + n \times 0,32)$ avec n le nombre de coordinateurs et le temps est estimé en ms. Pour l'exemple de la figure 2.10, avec uniquement 7 coordinateurs (qui forment un réseau à 2 sauts) la synchronisation va nécessiter un temps $T_{synch} = 71.68$ ms [Cha09].

Les résultats obtenus pour 30 coordinateurs et 4 feuilles actives par coordinateur, donnent une durée de synchronisation durant 528 ms pour un cycle global d'environ 6.53 secondes (avec 1.5 secondes pour la période $[T_1; T_2]$, 1.5 secondes pour $[T_2; T_3]$ et 3 secondes pour $[T_3; T_0]$). On remarque que le temps nécessaire pour réaliser la synchronisation coûte à lui seul un taux d'activité de 8.08 % sans compter le taux d'activité nécessaire pour échanger les données (cela vaut 45.95 %). Le temps de sommeil compte pour moins de la moitié du temps, soit 45,95 %.

De tels protocoles ne sont pas aptes à opérer avec des taux d'activité faibles ($\leq$1 %).

2.1.2 Protocoles MAC asynchrones

Dans les protocoles MAC asynchrones, les nœuds n'ont pas de calendrier commun. On peut classifier les protocoles MAC asynchrones économes en énergie en deux catégories suivant l'initiateur des communications : si la communication est initiée par l'émetteur (notés protocoles *sender initiated*), et si la communication est initiée par le récepteur (notés protocoles *receiver initiated*). Dans cette partie, nous décrivons tout d'abord le principal protocole MAC asynchrone de la littérature, qui est décrit dans la norme IEEE 802.15.4 en mode sans suivi de balises, puis les protocoles MAC *sender initiated*, et les protocoles MAC *receiver initiated*.

2.1.2.1 Norme IEEE 802.15.4 en mode sans suivi de balises

Dans le mode sans suivi de balises de la norme IEEE 802.15.4, les nœuds accèdent au médium en utilisant l'algorithme CSMA/CA non slotté. Dans ce mécanisme, Il n'y a pas de synchronisation entre les nœuds comme dans le mode avec suivi de balises. La période de *backoff* d'un nœud n'est pas liée à la période de *backoffs* des autres nœuds du PAN. La figure 2.11 décrit les étapes de l'algorithme CSMA/CA non slotté. Dans cet algorithme, chaque nœud maintient deux variables pour chaque tentative d'accès : BE qui permet de déterminer la fenêtre dans laquelle le *backoff* est tiré aléatoirement et NB qui représente le nombre de tentatives d'accès au médium de la trame en cours. Au départ, les paramètres BE et NB sont respectivement initialisés à $macMinBE$ et à zéro (étape 1). Ensuite, un temps d'attente aléatoire (en périodes de *backoffs*) est tiré dans l'intervalle $[0; 2^{\mathrm{BE}} - 1]$ (étape 2). Après ce temps d'attente, le nœud réalise un CCA pour vérifier si le canal est libre. Si c'est le cas, le nœud transmet ses données. Sinon (étape (3)), la sous-couche MAC incrémente NB et BE (tout en s'assurant que BE ne soit pas supérieur à $aMaxBE$). Si NB est inférieur ou égal à $macMaxCSMABakoffs$, l'algorithme CSMA-CA revient à l'étape (2). Si ce n'est pas le cas, l'algorithme CSMA-CA non slotté s'arrête et renvoie un échec d'accès au médium.
Le mode sans suivi de balises de la norme IEEE 802.15.4, ne nécessite pas de synchronisation (qui peut être une opération complexe, quand il y a un nombre important de nœuds). Cependant, ce mécanisme suppose que les coordinateurs ne passent jamais en mode sommeil, ce qui génère une consommation importante d'énergie.

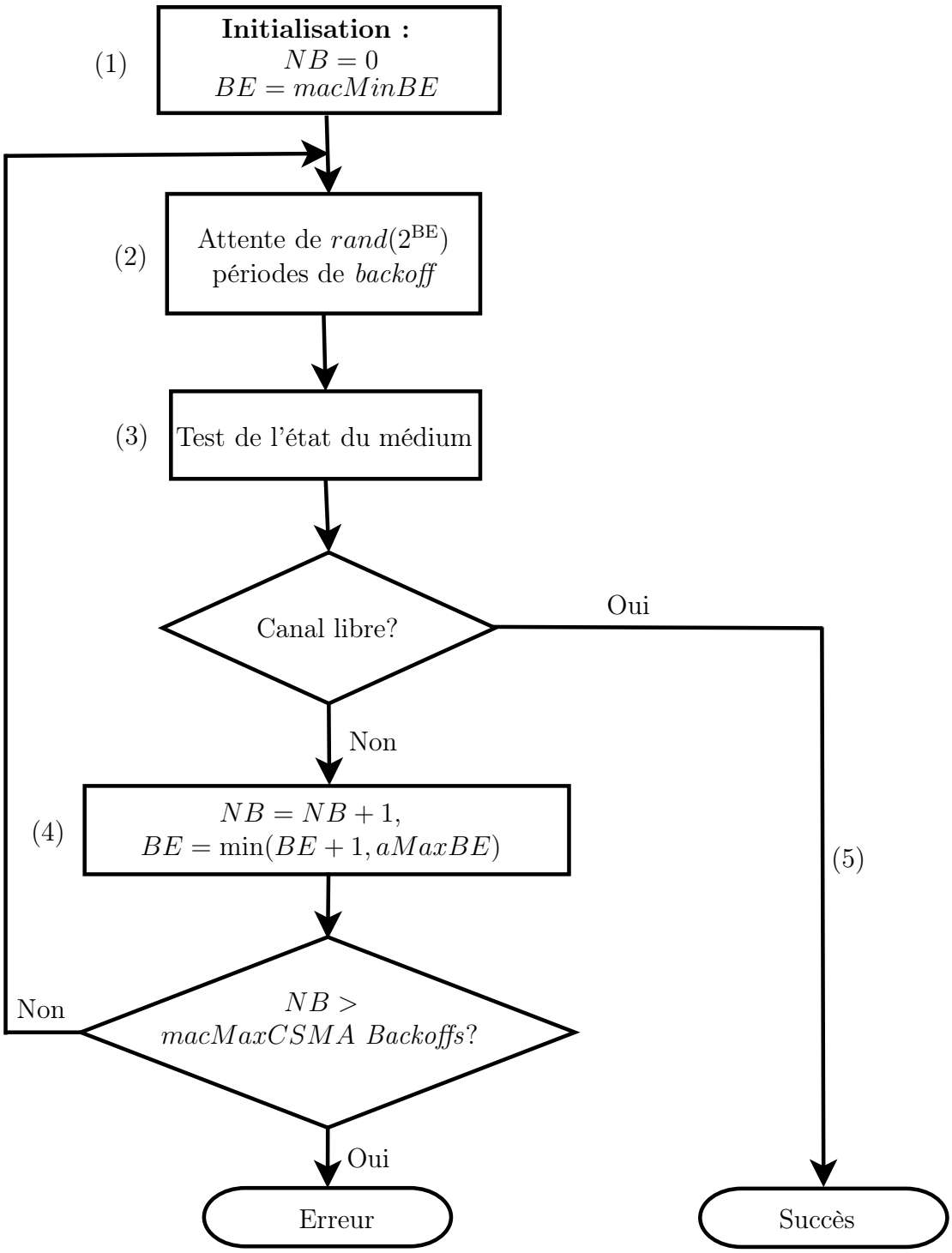


Figure 2.11 – Diagramme de l'algorithme CSMA/CA non slotté.

2.1.2.2 Protocoles *sender initiated*

Dans les protocoles *sender initiated*, l'essentiel du coût de communication est pris en charge par les émetteurs.

L'un des premiers protocoles basé sur ce principe est le protocole B-MAC (*Berkeley MAC*) [PHC04] qui est basé sur la technique LPL (pour *Low Power Listening*). Dans B-MAC, les récepteurs se réveillent périodiquement pour vérifier s'il y a une communication en cours. Lorsqu'un nœud a une trame à envoyer, il envoie un long préambule avant la transmission effective des données (voir figure 2.12). Le préambule doit être assez grand pour permettre au destinataire de se réveiller, et (ou) de détecter ce préambule. Cependant, ce mécanisme oblige tous les nœuds voisins de l'émetteur à rester actif jusqu'à la fin de la transmission des données, même si la trame ne leur est pas destinée. B-MAC souffre donc de réveils inutiles, ce qui génère une consommation inutile de l'énergie. Le long préambule avant l'envoi de chaque trame peut aussi provoquer un temps de latence de bout en bout élevé.

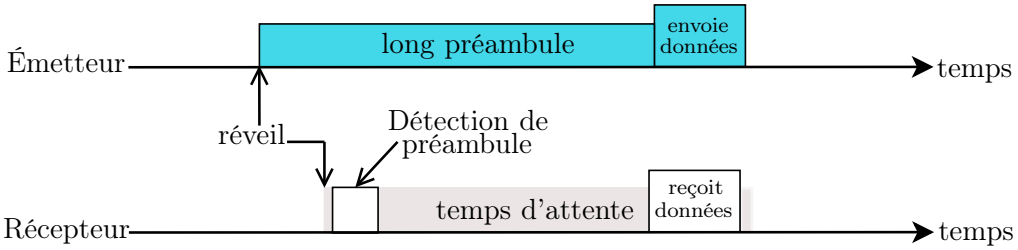


Figure 2.12 – Mécanisme de long préambule dans B-MAC.

Le protocole WiseMAC [EHDH04], réduit l'occupation du canal résultant de l'envoi d'un long préambule en incluant dans chaque trame ACK la date de prochain réveil du nœud. De cette façon, l'émetteur a connaissance du réveil de son récepteur, s'il a d'autres trames à lui envoyer il procède à l'envoi d'une petite trame préambule et débute rapidement l'émission des trames de données, comme le montre la figure 2.13.

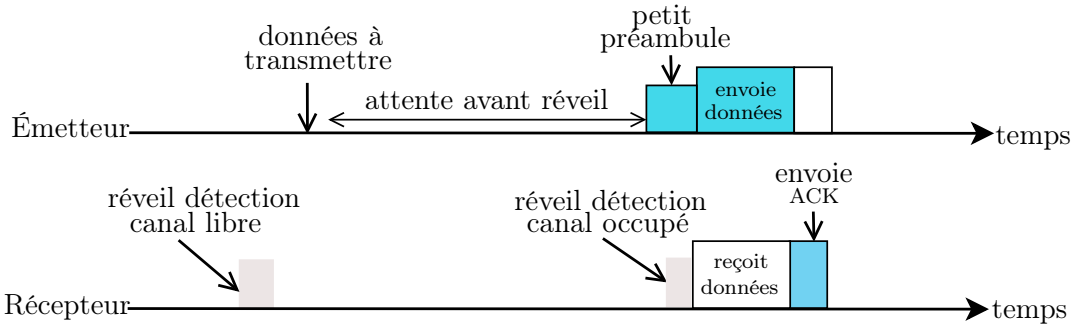


Figure 2.13 – Mécanisme de petit préambule dans WiseMAC.

Dans X-MAC [BGAH06], quand un nœud a une trame de données à émettre, il se

réveille immédiatement et écoute le canal jusqu'à ce qu'il soit libre. Lorsque le canal est libre, le nœud envoie une trame préambule avec l'adresse du récepteur et attend un accusé de réception, comme le montre la figure 2.14. Si un nœud se réveille et ne détecte aucun préambule, ou reçoit un préambule dont il n'est pas le destinataire et qu'il ne dispose d'aucune trame de données à émettre, il repasse en mode sommeil. Si un nœud qui a une trame de données à émettre entend le préambule d'un autre nœud après qu'il ait envoyé son premier préambule, il stoppe son envoi et attend l'accusé de réception du préambule en cours avant de reprendre son envoi de préambule. Lorsqu'un nœud reçoit un préambule dont il est le destinataire, il envoie un accusé de réception à l'émetteur du préambule pour lui indiquer sa disponibilité, puis reste actif pour recevoir les données. Dans X-MAC, après la réception d'une trame de données, le nœud récepteur reste encore actif après un certain temps en vu de recevoir d'autres trames d'éventuels nœuds dont il serait le récepteur. Ce qui fait que lorsqu'un nœud qui a des données à transmettre détecte un accusé de réception (dont il n'est pas le destinataire) avec comme source le récepteur qu'il attend, ce nœud fixe un temps aléatoire après lequel il envoie ses données.

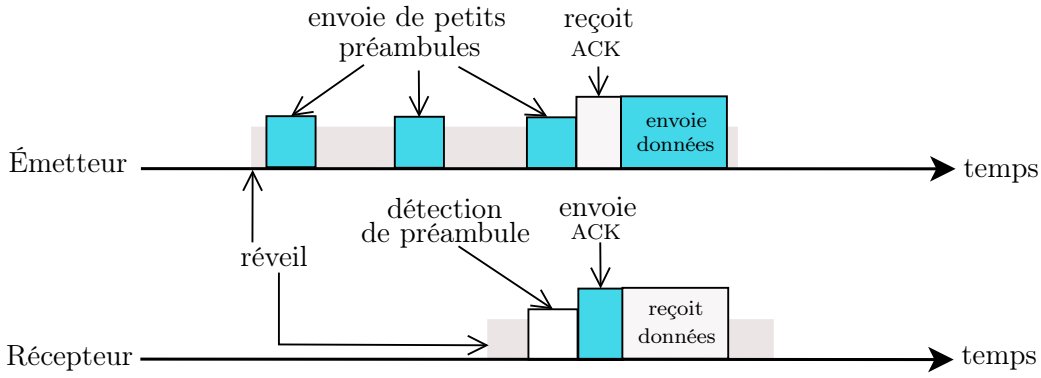


Figure 2.14 – Mécanisme de petits préambules dans X-MAC.

X-MAC assure ainsi un faible délai de bout en bout, mais génère beaucoup de collisions car l'intervalle entre deux envois de préambule peut être interprété comme un canal libre. De plus, le fait qu'il n'y ait pas d'accusé de réception pour les trames de données fait que l'émetteur n'a aucune connaissance de la bonne réception de la trame envoyée, ce qui impacte le taux de livraison des données. Enfin, tout comme la plupart des protocoles MAC asynchrones *sender initiated*, certains nœuds restent actifs beaucoup plus longtemps que d'autres, ce qui pose un problème d'équité dans la consommation énergétique, comme présenté dans [LLL14].

RA-MAC [LC08] se sert d'une succession de trames RTS comme petites trames de préambules. RA-MAC agrège les trames RTS de plusieurs expéditeurs en une seule trame : lorsqu'un nœud a une trame à émettre, il écoute le canal pendant un

temps aléatoire (soit T_1), en vue de recevoir une trame RTS avec la même adresse destinataire que son récepteur cible. S'il ne détecte aucune trame RTS après ce temps aléatoire, il diffuse une trame RTS principale (appelée mainRTS) et attend une trame RTS dérivée (appelée subRTS) similaire à une trame CTS pendant un temps T_2 (voir 2.15). Quand un nœud qui attend le même récepteur reçoit cette trame mainRTS, il la met à jour pour en faire une trame subRTS et la diffuse à son tour. Lorsque le nœud qui a diffusé la trame mainRTS reçoit la trame subRTS, il met à jour sa trame mainRTS pour en faire une trame RTS agrégée (appelée AggRTS) et la retransmet. Lorsque le récepteur se réveille et reçoit la trame AggRTS, il établit une grille de transmission pour les expéditeurs en attente, l'insère dans une trame CTS et la diffuse ensuite. Dès lors, chaque émetteur recevant cette trame CTS diffusera sa trame de données à la date qui lui est réservée pour éviter les collisions (voir 2.15).

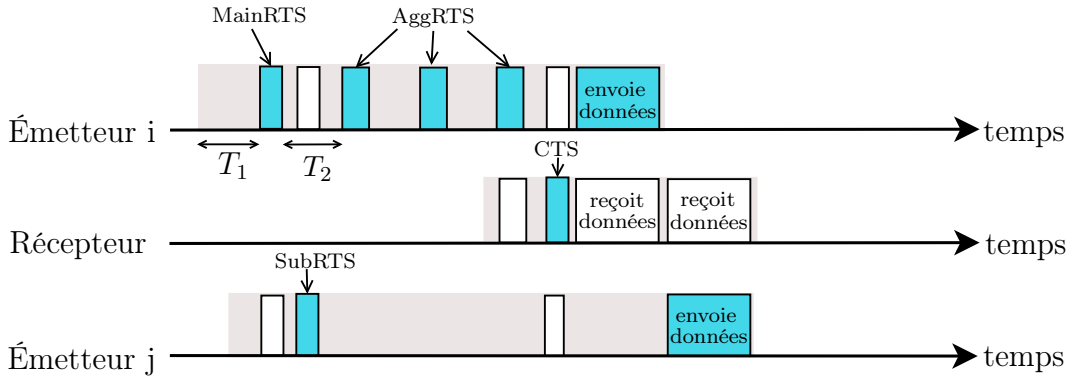


Figure 2.15 – Mécanisme d'agrégation de RTS dans RA-MAC.

RA-MAC n'implémente pas de mécanisme d'acquittement, ce qui fait qu'il est difficile de détecter une collision entre les trames subRTS de différents expéditeurs ou les collisions entre les différentes trames RTS et la trame CTS [DDB13].

Dans BoX-MAC [ML08], les auteurs proposent deux variantes de protocoles dont l'un (BoX-MAC-1) est basé sur B-MAC et l'autre (BoX-MAC-2) est basé sur X-MAC. BoX-MAC-1 transmet en continu un long préambule contenant une suite de trames de données (au lieu d'un long préambule sans données comme dans B-MAC) comme indiqué sur la figure 2.16. Lorsqu'un nœud se réveille et détecte un préambule, il attend de recevoir au moins une trame complète des suites de trames contenues dans le préambule. Il vérifie ensuite s'il est le destinataire, si ce n'est pas le cas, le nœud repasse immédiatement en mode sommeil. S'il est le nœud destinataire, il reste actif jusqu'à la fin du préambule et envoie une trame ACK à la source pour accuser réception. Ce mécanisme permet aux nœuds non destinataires d'économiser l'énergie et de réduire les délais de bout en bout par rapport à B-MAC.

BoX-MAC-2, propose de remplacer les transmissions de préambules courts par des

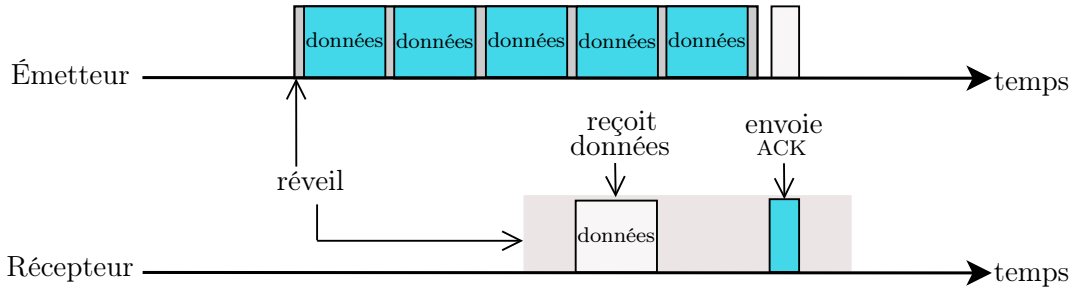


Figure 2.16 – Mécanisme de transmission dans BoX-MAC-1.

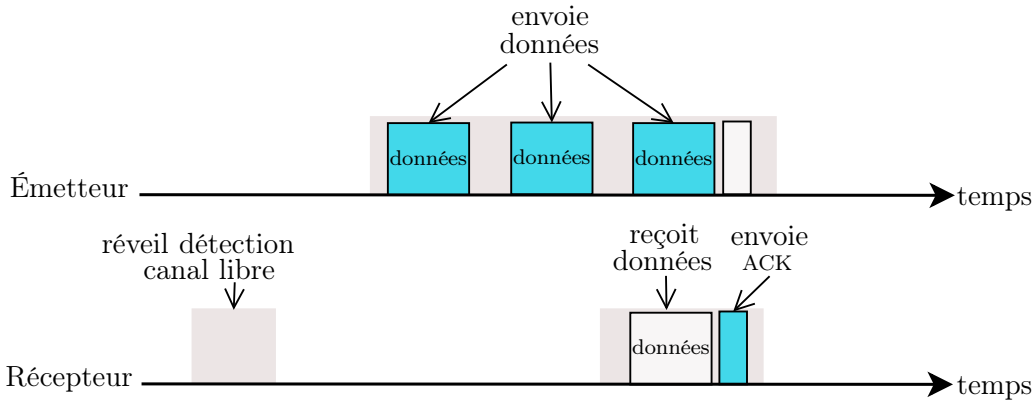


Figure 2.17 – Mécanisme de transmission dans BoX-MAC-2.

transmissions de données, comme indiqué sur la figure 2.17. Cela suppose toutefois que les trames de données sont assez courtes pour remplacer efficacement les petits préambules. De ce fait, cette approche peut fonctionner pour des applications qui génèrent de très petites trames de données, mais elle n'est pas adaptée aux applications qui génèrent des trafics volumineux.

ContikiMAC [Dun11] est basé sur deux principes pour la transmission des données. Le premier principe reprend le même schéma que dans BoXMac-2. L'auteur

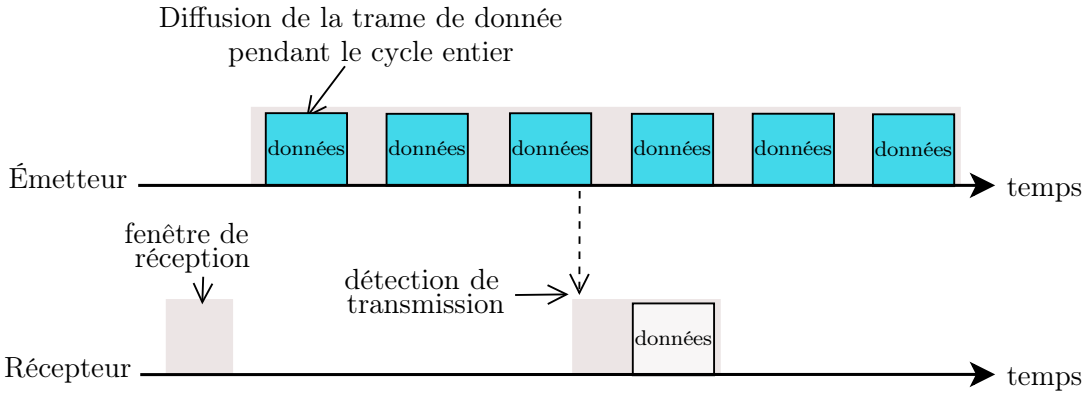


Figure 2.18 – Mécanisme de transmission par diffusion dans ContikiMAC.

propose comme optimisation de ce principe, qu'après une transmission réussie les

nœuds émetteurs stockent la date de réveil des récepteurs et de retarder les prochaines transmissions jusqu'à cette date. Ce qui permet de réduire la phase d'attente de l'émetteur et donc d'économiser de l'énergie.

Le deuxième principe de ContikiMAC opère comme indiqué sur la figure 2.18. Un émetteur répète la diffusion de la trame de données pendant le cycle entier afin de s'assurer que les autres nœuds reçoivent la trame de données au moins une fois. L'inconvénient remarquable dans ce deuxième mécanisme de ContikiMAC est le gaspillage important de l'énergie du côté de l'émetteur, car un nœud continuera d'envoyer une trame de données même si tous les destinataires attendus l'ont reçus.

De manière générale, les protocoles *sender initiated* ont l'inconvénient présenté dans la figure 2.19. Quand deux (ou plusieurs) nœuds ont une trame de données à envoyer à des destinataires différents (par exemple ici le nœud A envoie au nœud B et le nœud C envoie au nœud D) l'émission de préambules du nœud A empêche le nœud voisin C de transmettre ses données à D. Le nœud C est réveillé et détecte un préambule en cours, il ne peut donc pas envoyer de préambule pour détecter le réveil de son nœud destinataire D. Lorsque le nœud D se réveille et détecte qu'il n'est pas le destinataire du préambule en cours, il retourne immédiatement en mode sommeil alors qu'il aurait pu communiquer avec le nœud C, vu que le nœud B qui est le destinataire du préambule en cours n'est pas encore réveillé.

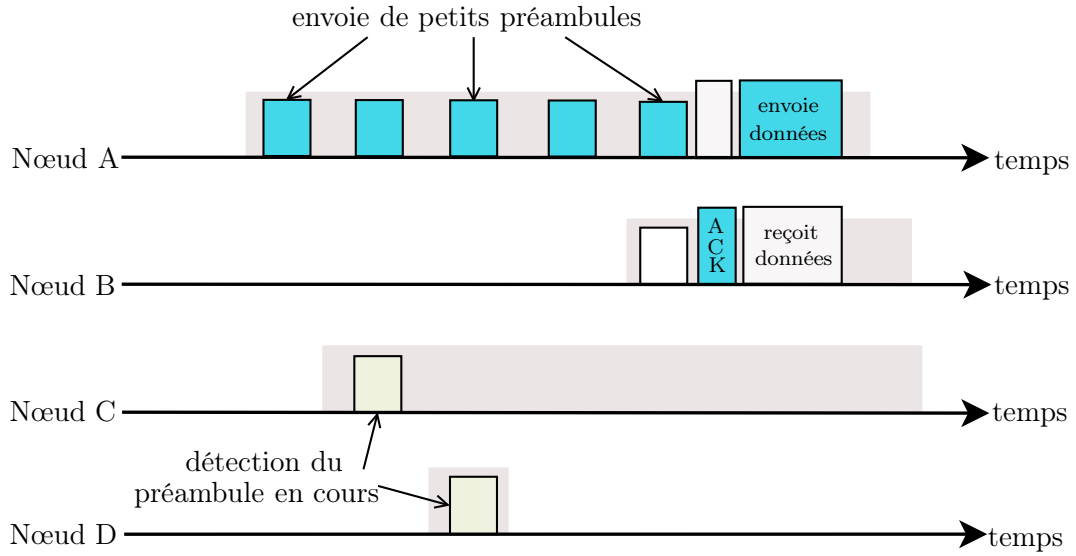


Figure 2.19 – Illustration de l'occupation du canal par les petits préambules dans les protocoles *sender initiated*.

Cela conduit à une faible utilisation des capacités du canal et à une grande latence pour la livraison des données. De plus, ces protocoles ne sont pas aptes à opérer avec

de faibles taux d'activité [VST14]. En effet, dans de telles circonstances la durée de la période d'activité du récepteur devra être très faibles par rapport à sa période d'inactivité. Ce qui conduirait à un plus important gaspillage en énergie de la part de l'émetteur (car il devra transmettre un plus gros préambules).

Pour remédier au problème d'occupation du canal par les préambules dans les protocoles *sender initiated*, d'autres protocoles appelés *receiver initiated* ont été proposés.

2.1.2.3 Protocoles *receiver initiated*

Dans les protocoles *receiver initiated*, les récepteurs se réveillent indépendamment, de façon périodique, et indiquent leur disponibilité à recevoir des données par l'envoi d'une balise. Avec ce mécanisme, la balise du récepteur n'occupe pas le canal aussi longtemps que les préambules dans les protocoles *sender initiated*.

RI-MAC [SGJ08] est le premier protocole basé sur ce mécanisme inspiré de la technique LPP (pour *Low Power Probing*) du système Koala [MELT08]. Dans RI-MAC, lorsqu'un nœud a une trame à émettre, il se réveille immédiatement et attend la balise d'un récepteur potentiel. Quand un nœud qui n'a pas de trame à émettre se réveille, envoie sa balise et ne détecte aucune trame de données pendant un temps déterminé, il retourne en mode sommeil. Dès la réception de la balise d'un récepteur, les émetteurs envoient immédiatement leurs trames (voir la figure 2.20). Ce mécanisme peut générer des collisions, mais RI-MAC effectue une nouvelle diffusion de la balise du récepteur après chaque détection de collisions.
RI-MAC réduit l'occupation du canal par rapport à X-MAC, et fournit un grand taux de livraison des données avec un faible délai de bout en bout. Cependant, certains nœuds restent plus longtemps actifs que d'autres, ce qui provoque une inéquité dans la consommation énergétique et empêche la bonne prédiction de la durée de vie du réseau. Aussi, RI-MAC ne prend pas en charge la diffusion. Certains protocoles comme ADB [SGD+09] et le protocole de [YM11] apportent une extension favorable à la diffusion.

Le protocole PW-MAC [TS+11] améliore RI-MAC en réduisant le gaspillage énergétique de l'émetteur quand il attend le réveil d'un récepteur. Pour cela, chaque nœud calcule son prochain réveil à l'aide d'un générateur pseudo-aléatoire. Lorsqu'un nœud connaît les paramètres du générateur pseudo-aléatoire du récepteur voulu et qu'il a une trame à lui envoyer, il prédit le réveil du récepteur et se réveille à temps pour lui envoyer ses trames en utilisant la formule $X_{n+1} = (aX_n + c)\ mod\ m$, avec $a = nodeID * 20$, $c = 7$ et $m = 1000$. PW-MAC intègre un mécanisme de correction d'erreur de prédiction et gère efficacement les retransmissions par rapport

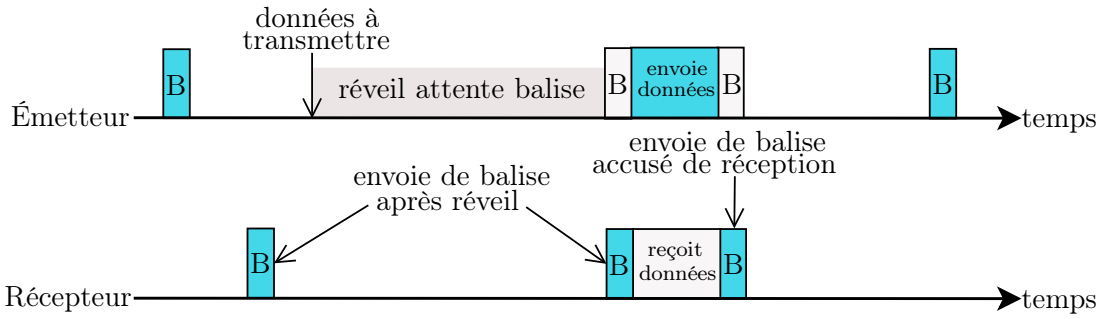


Figure 2.20 – Mécanisme de transmission dans RI-MAC.

à RI-MAC. En effet, en cas de retransmission sans succès, les nœuds ne restent pas éveillés jusqu'au prochain réveil de l'émetteur, mais font une nouvelle prédiction pour le prochain réveil du récepteur, et repassent en mode sommeil jusqu'à cet instant.

Néanmoins, PW-MAC a plusieurs inconvénients. Il suppose tout d'abord que les nœuds sont synchronisés. Il ne résout pas le problème d'inéquité dans la consommation de l'énergie. Il provoque aussi beaucoup de collisions quand plusieurs émetteurs envoient simultanément leurs trames au même récepteur. De plus, lors de l'implémentation nous avons remarqué que les paramètres du générateur pseudo-aléatoire (qui est un générateur congruentiel linéaire à 1 pas du type $X_{n+1} = (aX_n + c)\ mod\ m$) choisit par les auteurs génère des séquences de période 1. En effet, le choix de $a = nodeID*20$, $c = 7$ et $m = 1000$ ne respecte pas la propriété de période maximale de [Knu98] qui dit qu'une séquence a une période de longueur m si et seulement si : c est premier avec m, $b = a - 1$ est un multiple de p pour chaque nombre premier p divisant m, et b est un multiple de 4 si m est un multiple de 4. Par exemple, si on prend le nœud 1 avec $nodeID = 1$, on aura $a = 20*1$, $b = a-1 = 19$, et les nombres premiers 2 et 5 divisent $m = 1000$ mais ne sont pas des multiples de b. Le nœud 1 aura la séquence suivante 7, 147, 947, 947, 947, 947, etc. Un choix de $a = nodeID*21$, $c = 7$ et $m = 1000$ aurait été plus approprié.

OC-MAC [WZCZ10] améliore RI-MAC en exploitant la coopération entre les émetteurs actifs pour réduire leur temps d'attente et économiser de l'énergie. Dans OC-MAC, les émetteurs voisins actifs sont autorisés à échanger leurs trames entre eux pendant l'attente du réveil du récepteur. Ainsi, après avoir désigné un relais qui attend la disponibilité des récepteurs et lui avoir transmis leurs trames, les autres émetteurs peuvent passer en mode sommeil, ce qui réduit la consommation d'énergie des émetteurs. Un mécanisme d'établissement de liaison similaire à celui du RTS/CTS est utilisé pour la coopération entre les paires de nœuds. Quand un nœud a une trame de données à transmettre, il écoute le canal pendant un petit moment pour déterminer s'il y a une coopération en cours ou non. S'il n'y a pas de coopération en

cours, il diffuse une trame RTR (similaire à RTS) comportant son énergie résiduelle et l'adresse du destinataire. Ensuite le nœud attend pendant un temps t_w pour voir si d'autres émetteurs peuvent transmettre sa trame. Si après ce temps il ne reçoit aucune trame CTR (similaire à CTS), il conclut qu'il n'y a aucune coopération en cours, perd le droit d'initiation de coopération et continue de rester actif en silence jusqu'à ce qu'il entende un autre émetteur ou que son récepteur se réveille pour recevoir la trame (voir la figure 2.21). Quand un émetteur reçoit une trame RTR d'un autre émetteur, il compare son énergie résiduelle à celle de l'autre émetteur. S'il a plus d'énergie résiduelle, il fixe un temps aléatoire après lequel il diffuse une trame CTR. Le premier émetteur qui émet un CTR devient le relais. Une fois qu'un émetteur reçoit un CTR en réponse à un RTR diffusé, il transmet ses données au relais sélectionné (qui est la source de la trame CTR) et attend un accusé de réception après lequel il passe en mode sommeil. Avec ce mécanisme, OC-MAC réduit le temps d'attente des émetteurs (surtout dans un réseau où les nœuds ont beaucoup de voisins) pour économiser l'énergie.

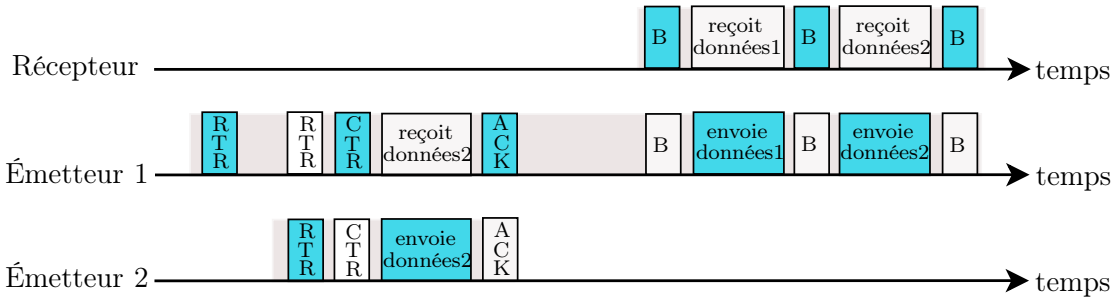


Figure 2.21 – Mécanisme de transmission dans OC-MAC.

Dans EM-MAC [TSG+11] les nœuds décident indépendamment de leur horaire de réveil et du canal d'échange en utilisant un générateur pseudo-aléatoire. Le canal de réveil n'est pas nécessairement le même que le canal d'échange de données. EM-MAC permet à l'expéditeur de se réveiller juste avant la balise du récepteur comme dans PW-MAC. Par contre, EM-MAC nécessite une phase de découverte de voisinage et maintient une table comportant des informations sur tous les voisins.

Dans HKMAC [TSLF13], chaque nœud se réveille périodiquement et diffuse une balise pour notifier qu'il est prêt à recevoir des données. Lorsqu'un nœud reçoit une balise du récepteur attendu, il transmet immédiatement ses trames de données. Si aucune trame n'est reçue après la diffusion de la balise, le nœud retourne en mode sommeil afin d'économiser de l'énergie. HKMAC utilise l'approche hybride suivante. Le temps est divisé en périodes d'activation aléatoire (RP) et en périodes d'activation planifiée (SP). Pendant la période RP, les nœuds fonctionnent comme

dans RI-MAC. Pendant la période SP, les récepteurs ajustent de façon adaptative l'instant d'envoi de leur balise pour permettre aux émetteurs de planifier leur période de réveil. L'émetteur est conscient de l'instant de réveil du récepteur, et peut donc programmer ses transmissions pour réduire le temps d'écoute avant transmission.

Dans REA-MAC [TCSL13], les nœuds utilisent la diffusion de balises pour exprimer leur disponibilité, comme dans RI-MAC. Par contre, les temps d'activité et d'inactivité des nœuds ne sont pas aléatoires comme dans RI-MAC mais sont calculés sur la base d'informations de routage. Cela conduit à ce que les trames soient transmises en multi-saut selon un *pipeline*. Dans REA-MAC, si le nombre maximal de sauts des nœuds aux puits est n, le cycle est divisé en n intervalles de temps. Les nœuds ne se réveillent pas immédiatement pour attendre le récepteur lorsqu'ils ont une trame de données à émettre, mais chaque nœud se sert de la distance du récepteur voulu au puits pour décider de sa date de réveil pour l'émission. Ainsi les nœuds qui sont plus loin du puits se réveillent plus tôt que ceux qui sont plus près du puits. Ce mécanisme de réveil progressif dans REA-MAC permet notamment de réduire le temps d'attente des émetteurs et d'économiser de l'énergie par rapport à RI-MAC. Cependant, il nécessite une connaissance globale du réseau, afin de définir le nombre maximal de sauts des nœuds au puits, qui est un paramètre primordial pour définir le cycle d'activité.

ERI-MAC [NNL+14] a été récemment proposé comme une amélioration de RI-MAC. ERI-MAC suppose que les nœuds sont capables de récupérer l'énergie de l'environnement (énergie solaire ou énergie éolienne [KHZS07] [VRS03], etc.). ERI-MAC propose un mécanisme de file d'attente pour ajuster le taux d'activité des nœuds, suivant le taux de récupération d'énergie. Ce mécanisme de communication est montré sur la figure 2.22. Le temps est divisé en cycle et les nœuds non-émetteurs diffusent

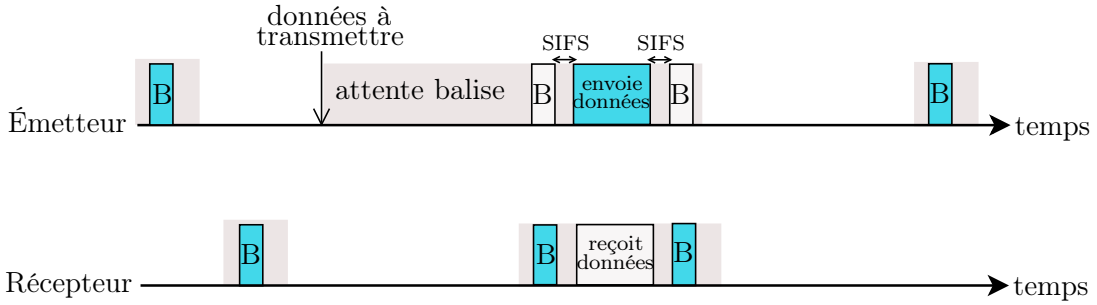


Figure 2.22 – Mécanisme de transmission dans ERI-MAC.

immédiatement une trame balise après chaque réveil, puis écoutent le canal pendant une courte période pour déterminer s'il y a une potentielle trame. En outre, les émetteurs écoutent le canal pour recevoir la balise du récepteur attendu. Lorsqu'un

nœud reçoit la balise du récepteur attendu, il envoie sa trame en attente après un court temps SIFS. Une transmission réussie est terminée lorsqu'une balise d'accusé de réception (ACK) arrive à l'expéditeur. Cette balise sert aussi de nouvelle balise d'annonce de disponibilité. ERI-MAC utilise le même mécanisme de gestion de collision que RI-MAC. Les auteurs proposent une extension de leur protocole qui consiste à concaténer les trames de données de la file d'attente pendant un intervalle de temps pour en faire une super trame (dont la taille dépend des capacités radio) avant de les envoyer. Cet intervalle de temps est dynamiquement contrôlé par les nœuds pour assurer un rapport de 1 entre l'énergie récupérée et l'énergie consommée. Cela permet d'atteindre et de maintenir un fonctionnement neutre en énergie appelé ENO [KHZS07] (pour *Energy Neutral Operation*).

De manière générale, les protocoles MAC *receiver initiated* permettent d'économiser de l'énergie du côté de l'émetteur par rapport aux protocoles *sender initiated*. Cependant, les envois périodiques de balises ajoutent une surcharge en messages de contrôle et augmentent la contention pour l'accès au médium, ce qui augmente la consommation énergétique des nœuds. De plus, ces protocoles ne sont pas efficaces pour les faibles taux d'activité ou pour les haut taux d'activité [VST14]. En effet, dans le premier cas (pour les faibles taux d'activité), les nœuds vont attendre pendant longtemps la balise du récepteur avant d'émettre leur trame de données, ce qui augmente le temps de latence de bout en bout. Dans le second cas (pour les grands taux d'activité), une transmission fréquente de balise par les récepteurs peut provoquer une forte contention et une surconsommation d'énergie. De plus, peu de protocoles proposent un mécanisme garantissant une longue durée de vie au réseau en équilibrant la consommation énergétique de chacun des nœuds.

2.1.3 Bilan sur les protocole MAC

Le tableaux 2.1 résume les principaux protocoles MAC à taux d'activité de la littérature que nous avons passés en revue, avec les avantages et les inconvénients dans chaque catégorie.

Tableau 2.1 – Tableau récapitulatif des protocoles MAC à taux d'activité

Catégories	**Protocoles**	**Avantages**	**Inconvénients**
Synchronisés / global	IEEE 802.15.4 [IEE11] mode avec suivi de balises, S-MAC [YHE02], T-MAC [YHE04], R-MAC [DSJ07], LO-MAC [NJY13], [KJK15]	communications aisées, faible latence	surcharge en messages de contrôle, forte contention, collisions, consommation importante en énergie, problème de passage à l'échelle, pas efficace à faible taux d'activité
Synchronisés / local	D-MAC [LKR04], Q-MAC [VA06], TreeMAC [SHSL09]	facile à mettre en œuvre, communications aisées	temps de latence élevé, forte contention, collisions, consommation énergétique élevée, pas efficace à faible taux d'activité
Asynchronisés / *sender initiated*	B-MAC [PHC04], WiseMAC [EHDH04], X-MAC [BGAH06], RA-MAC [LC08], BoX-MAC [ML08], ContikiMAC [Dun11]	simplicité de mise en œuvre, ne nécessite pas une connaissance globale des horaires des autres nœuds	occupation du canal par les préambules, grande inéquité (côté émetteur) dans la consommation d'énergie, n'opère pas à faible taux d'activité
Asynchronisés / *receiver initiated*	RI-MAC [SGJ08], ADB [SGD+09], [YM11], PW-MAC [TS+11], OC-MAC [WZCZ10], EM-MAC [TSG+11], HKMAC [TSLF13], REA-MAC [TCSL13], ERI-MAC [NNL+14]	facile à mettre en œuvre, ne nécessite pas de synchronisation, économise plus d'énergie (côté émetteur) par rapport aux protocoles *sender initiated*	inéquité dans la consommation d'énergie, surcharge en messages de contrôle due aux balises, collisions, n'opère pas à faible taux d'activité et pas efficace pour des taux d'activité élevés

Dans la partie qui suit nous passons en revue les protocoles de routages économes en énergie (basés sur un mécanisme de séquence de périodes d'activité et d'inactivité) proposés dans la littérature pour les réseaux de capteurs sans fil.

2.2 Protocoles de routage opportuniste pour les réseaux de capteurs sans fil

De nombreux protocoles de routage pour les RCSF ont été proposés dans la littérature. Parmi ces protocoles, certains font l'hypothèse que les nœuds gardent leur radio allumée la plupart du temps, ou que les activités entre les nœuds sont simultanées. Dans cette partie, nous nous intéressons aux protocoles de routage qui ne font pas ces hypothèses et qui profitent des rencontres opportunistes pour faire communiquer les nœuds. Les protocoles de routage opportuniste sont souvent utilisés dans le contexte des MANET (pour *Mobile AdHoc NETworks*), mais sont aussi adaptés aux RCSF pour lesquels les nœuds ont des rencontres peu fréquentes et non périodiques. Dans un routage opportuniste, un nœud qui a une trame à émettre la diffuse dans un premier temps. Ensuite, un nœud relais est choisi parmi un ensemble de relais potentiels pour retransmettre la trame. Ce processus se répète jusqu'à ce que le paquet atteigne sa destination finale. En général, ces protocoles sont basés sur le calcul d'un gradient (qui représente la distance d'un nœud au puits) pour le choix du relais potentiel ou par diffusion de plusieurs copies du même paquet jusqu'à ce que la destination soit atteinte. Dans ce qui suit, nous décrivons dans un premier temps, la norme ZigBee [Zig08] et ensuite quelques protocoles de routage opportuniste par gradient et par inondation.

2.2.1 Norme ZigBee

La norme ZigBee [Zig08], définie par la ZigBee Alliance, se base sur les couches basses de la norme IEEE 802.15.4 [IEE06] (décrite dans la partie 2.1.1) pour apporter principalement des spécifications de la couche réseau et de la couche application. Nous décrirons uniquement la couche réseau qui assure la fonctionnalité de routage.

La figure 2.23 représente les composants et les interfaces de la couche réseau de la norme ZigBee. Cette couche fournit deux types de services *via* des points d'accès aux services notés SAP (pour *Service Access Points*).
Les entités NLDE (pour *Network Layer Data Entity*) et NLME (pour *Network Layer Management Entity*) assurent respectivement l'échange de primitives de données et de contrôle entre la couche réseau et la couche application. L'entité NLDE fournit

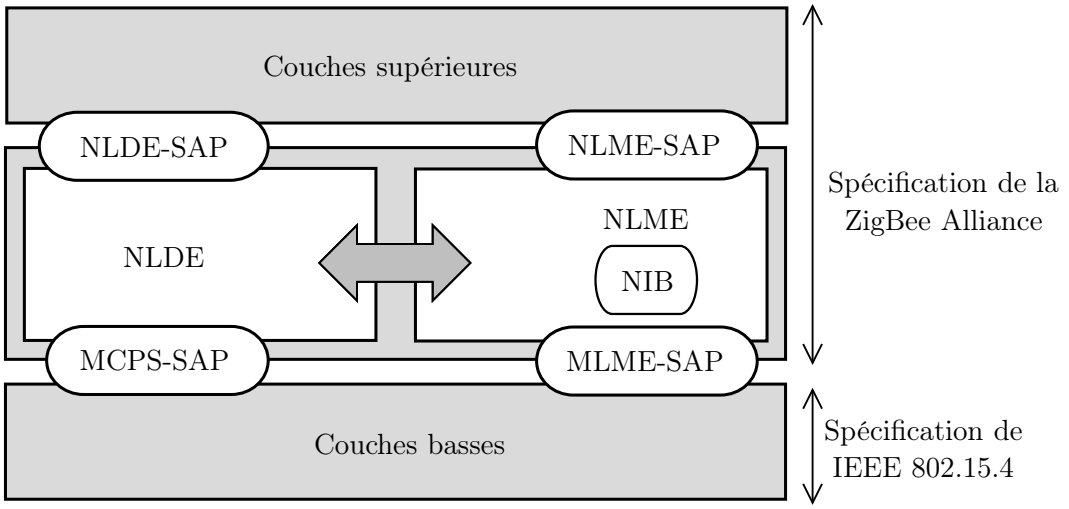


Figure 2.23 – Modèle de référence de la couche réseau ZigBee.

le service de transmission de données *via* son interface appelée NLDE-SAP. L'entité NLME assure un service de gestion *via* son interface appelée NLME-SAP.
L'entité NLME utilise l'entité NLDE pour assurer certaines de ses tâches de gestion, et maintient également une base de données des objets qu'elle gère connus sous le nom de la base d'information du réseau noté NIB (pour *Network layer Information Base*).
La double flèche représente une interface implicite entre l'entité NLME et l'entité NLDE et permet d'utiliser le service de données du réseau.

2.2.1.1 Description des topologies dans la norme ZigBee

Le standard ZigBee définit 3 types de nœuds (voir figure 2.24). Le coordinateur ZigBee, noté ZC (pour *ZigBee Coordinator*) est équivalent au coordinateur de PAN dans la norme IEEE 802.15.4. Il s'agit d'un nœud FFD qui gère l'ensemble du réseau. Le routeur ZigBee, noté ZR (pour *ZigBee Router*) est aussi une entité FFD qui a des capacités de routage. Le dispositif terminal, noté ZED (pour *ZigBee End Device*) est une entité RFD ou FFD mais ne peut pas assurer la fonction de routage.
Comme le montre la figure 2.24, la couche réseau de la norme ZigBee supporte trois types de topologies : en étoile, en arbre et maillée. Dans une topologie en étoile (voir figure 2.24 (1)), le réseau est contrôlé par un seul dispositif ZC qui est chargé d'initier et de maintenir tous les autres dispositifs (qui sont des ZED). Tous les dispositifs communiquent directement avec le ZC. Dans une topologie en arbre (voir figure 2.24 (2)) ou maillée (voir figure 2.24 (3)) le ZC est responsable du démarrage du réseau mais les communications peuvent passer par des ZR intermédiaires.

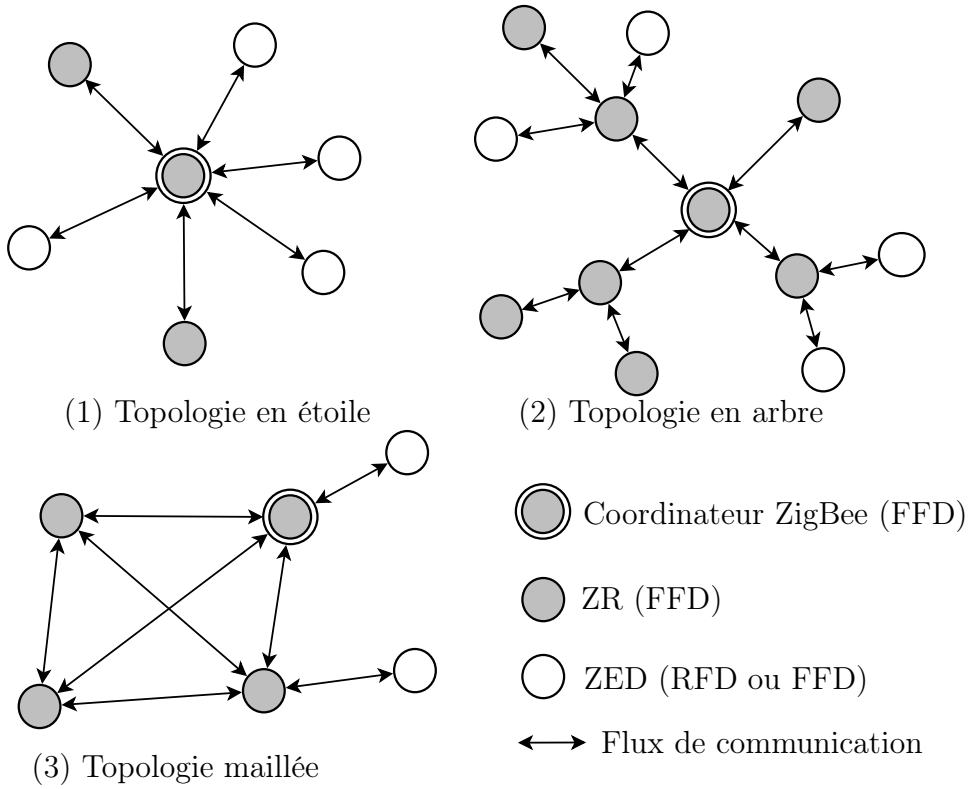


Figure 2.24 – Topologies de la norme ZigBee.

2.2.1.2 Routage dans ZigBee

Le norme ZigBee s'appuie sur deux protocoles de routage. Le protocole de routage réactif AODV [PBRD03] (pour *AdHoc On-demand Distance Vector*) est utilisé pour router les données dans une topologie maillée, et le protocole de routage hiérarchique HRP (pour *Hierarchical Routing Protocol*) pour les topologies en arbre.

AODV est un protocole de routage qui opère à la demande, c'est-à-dire ne maintient pas de routes inutilisées. Dans AODV, lorsqu'un nœud a un paquet à émettre pour une nouvelle destination, il diffuse une requête RREQ (pour *Route REQuest*) de demande de route à son voisinage. Un nœud qui reçoit cette requête et qui dispose d'un chemin dans sa table de voisinage pour cette destination, renvoie par le chemin inverse un RREP (pour *Route REPly*) au nœud demandeur. Après avoir reçu un RREP, le nœud source peut envoyer ses données.

Dans le routage hiérarchique, les communications sont réalisées suivant les liens parents-enfants de la topologie en arbre. Un mécanisme d'allocation d'adresses hiérarchiques est utilisé pour affecter une adresse à chaque nœud du réseau en fonction de sa position dans l'arbre. Ce mécanisme fournit à chaque parent potentiel un sous-bloc d'adresses à attribuer à ses descendants. Ce mécanisme s'appuie sur des paramètres fixés par le ZC lors de l'établissement du réseau : le nombre maximum de fils par rou-

teur ZigBee noté C_m, le nombre maximum de fils routeurs par routeur ZigBee noté R_m, et la profondeur maximale d'un nœud notée L_m. Une fonction appelée C_{skip} (voir équation (2.1)) est utilisée pour calculer la taille des sous-blocs d'adresses à attribuer à chaque parent situé à une profondeur d.

$$C_{skip}(d) = \begin{cases} 1 + C_m(L_m - d - 1) & \text{si } R_m = 1, \\ \frac{1+C_m-R_m-C_m*R_m^{L_m-d-1}}{1-R_m} & \text{sinon.} \end{cases} \tag{2.1}$$

Pour les dispositifs ZED, l'adresse est affectée de manière séquentielle suivant l'équation (2.2). A_n représente l'adresse du n$^{\text{ème}}$ fils d'un coordinateur (A_{parent}) qui est positionné à une profondeur d dans l'arbre.

$$A_n = A_{parent} + C_{skip}(d) * R_m + n. \tag{2.2}$$

Un nœud ZED communique uniquement avec son père et un nœud ZR maintient une table de voisinage comportant les adresses de ses descendants et celle du père auquel il est associé.

Lorsqu'un nœud n_i de profondeur d_i et d'adresse réseau A envoie un paquet à un autre nœud n_j d'adresse réseau D, le prochain saut est déterminé selon l'algorithme suivant :

n_i détermine si n_j est l'un de ses descendants en vérifiant si $A < D < A+C_{skip}(d_i-1)$.

Si n_j est un descendant de n_i, dans ce cas, il détermine si l'adresse réseau D est une feuille en vérifiant si $D > A + R_m \times C_{skip}(d_i)$.

Si, c'est une feuille le prochain saut est le nœud n_j.

Sinon, le prochain saut est son fils coordinateur dont l'adresse réseau est $N = A + 1 + \lfloor \frac{D-(A+1)}{C_{skip}(d_i)} \rfloor \times C_{skip}(d_i)$

Sinon, n_j n'est pas un descendant de n_i et le prochain saut est le père de n_i.

Notons que les deux protocoles de routage proposés pas ZigBee nécessitent une connaissance préalable du réseau et le maintien d'une table de voisinage. De plus, AODV nécessite un échange important de messages de contrôle.

2.2.2 Protocoles de routage opportuniste par gradient

Les protocoles de routage opportuniste par gradient sont basés sur le calcul d'un gradient, qui représente la distance d'un nœud au puits. La distance peut être calculée selon le nombre de sauts, l'énergie résiduelle des nœuds, la fiabilité des liens, etc. Le nœud relais va être le nœud qui propose le meilleur gradient.

ExOR [BM05] est un protocole de routage opportuniste proposé pour les réseaux sans fil maillés. ExOR vise à ce que chaque paquet soit retransmis uniquement par le meilleur relais pour éviter un nombre important de copies. Dans ExOR, les émetteurs transmettent leurs paquets par lots en insérant une liste des relais potentiels avec un ordre de priorité dépendant du coût estimé (similaire à l'estimation de métrique EXT (pour *Expected Transmission Count*) dans [DCABM03] ou à F-EXT [BCH15] (pour *Fast-ETX*)) pour atteindre la destination. Lorsqu'un récepteur reçoit avec succès un paquet, le nœud vérifie s'il est dans la liste des relais potentiels. Si oui, le nœud met le paquet en file d'attente et attend la fin de la réception du lot de paquets. Ensuite, chaque relais potentiel fixe une temporisation en fonction de sa position dans la liste des relais potentiels. Les relais avec une forte priorité diffusent en premier le lot de paquets de leur file d'attente. Les autres relais transmettent ensuite dans l'ordre en envoyant uniquement les paquets qui n'ont pas été retransmis par les nœuds de forte priorité. Le processus se poursuit jusqu'à ce que 90 % du lot de paquets atteignent la destination. Les 10% restants sont envoyés suivant le routage traditionnel (c'est-à-dire sans un ordre de priorité entre relais potentiels).
ExOR assure que chaque paquet soit transmis avec un nombre réduit de retransmissions par rapport au routage traditionnel. Toutefois, dans ExOR, la durée de planification des envois ne dépend pas du nombre de paquets à envoyer mais plutôt du nombre de relais potentiels. Dès lors, plus cette liste est longue, plus la durée de planification est longue, ce qui génère un temps important de planification même pour un nombre de paquets relativement faible. Aussi, ExOR nécessite un lot de l'ordre d'une dizaine de paquets, ce qui implique qu'il ne sera efficace que pour des applications avec des flux persistants, qui génèrent toujours au moins le nombre minimum de paquets nécessaires pour remplir un lot [BN10].

SGF [HCXT09] est un autre protocole de routage opportuniste par gradient basé sur le protocole GRAB (pour *GRAdient Broadcast*) [YZLZ05]. Dans GRAB, un paquet ADV (pour *ADVertisement*) est propagé par le puits et chaque nœud qui reçoit ce paquet détermine son coût minimum nécessaire pour atteindre le puits. Ainsi, un nœud qui a un paquet de données à émettre, insère son coût dans le paquet et le diffuse à ses voisins. Seul les voisins avec un coût plus faible que le coût contenu dans le paquet sont autorisés à le retransmettre.
SGF intègre un mécanisme d'économie d'énergie dans le protocole GRAB par une sélection opportuniste du prochain saut. Dans SGF, le gradient d'un nœud est établi sur la base de la consommation d'énergie minimum de la transmission d'un paquet du nœud jusqu'au puits. Le relais pour un nœud donné est choisi parmi plusieurs nœuds candidats suivant le gradient, l'état du canal, et l'énergie restante de ce nœud. SGF

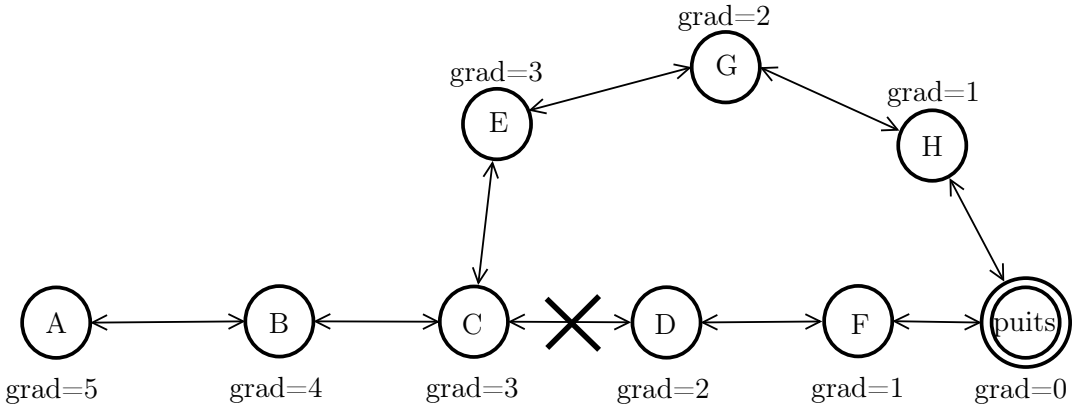


Figure 2.25 – Exemple d'échec de maintenance de gradient dans SGF.

ne nécessite pas que les nœuds maintiennent des informations sur leur voisinage ou sur la topologie du réseau, ce qui fait qu'il peut être déployé pour un réseau évolutif. Cependant, les auteurs proposent de router les paquets vers l'arrière (retourné au nœud prédécesseur) lorsqu'un nœud ne trouve pas un relais plus proche du puits pour router le paquet. Cela peut conduire à retourner inutilement plusieurs sauts en arrière et même à perdre le paquet alors qu'il peut y avoir un autre chemin vers le puits. La figure 2.25 illustre un cas de topologie mettant en œuvre ce problème [Mou13]. Un paquet provenant du nœud A arrive au nœud C après la rupture du lien entre les nœuds C et D, ce paquet doit retourner au nœud B ou au nœud A, ce qui peut conduire à une impasse, alors qu'il est possible de transmettre par exemple ce paquet de C à E.

ORW [LGDJ12, GLS+14] est un protocole de routage basé sur un mécanisme avec un séquencement d'activité et d'inactivité non synchronisé. De ce fait, il est basé sur les protocoles MAC asynchrones comme X-MAC [BGAH06] ou BoX-MAC [ML08] précédemment décrit dans la partie 2.1.2. Dans ORW, un émetteur transmet un flux de données jusqu'à ce que le destinataire prévu se réveille et le reconnaisse. Pour intégrer un routage opportuniste dans cet environnement, ORW propose que la transmission des données se fasse de la façon suivant : le premier voisin qui se réveille, reçoit avec succès un paquet et fournit une meilleure progression vers la destination, acquitte le paquet et se charge de le relayer à son tour. Notons que ce voisin n'est pas forcément connu à l'avance par l'expéditeur. Par exemple sur la figure 2.26 (b), le nœud A peut atteindre le nœud C directement *via* un lien peu fiable (car C positionné un peu plus loin de A) ou *via* B (qui est plus à portée de A). ORW propose d'intégrer le lien $A \rightarrow C$ dans le processus de routage. Par exemple, si le lien $A \rightarrow C$ est temporairement disponible et que C se réveille avant B, ORW utilise ce lien, ce qui permet de réduire la consommation d'énergie et le

délai (voir la figure 2.26 (c)). ORW propose une communication en *unicast* avec une

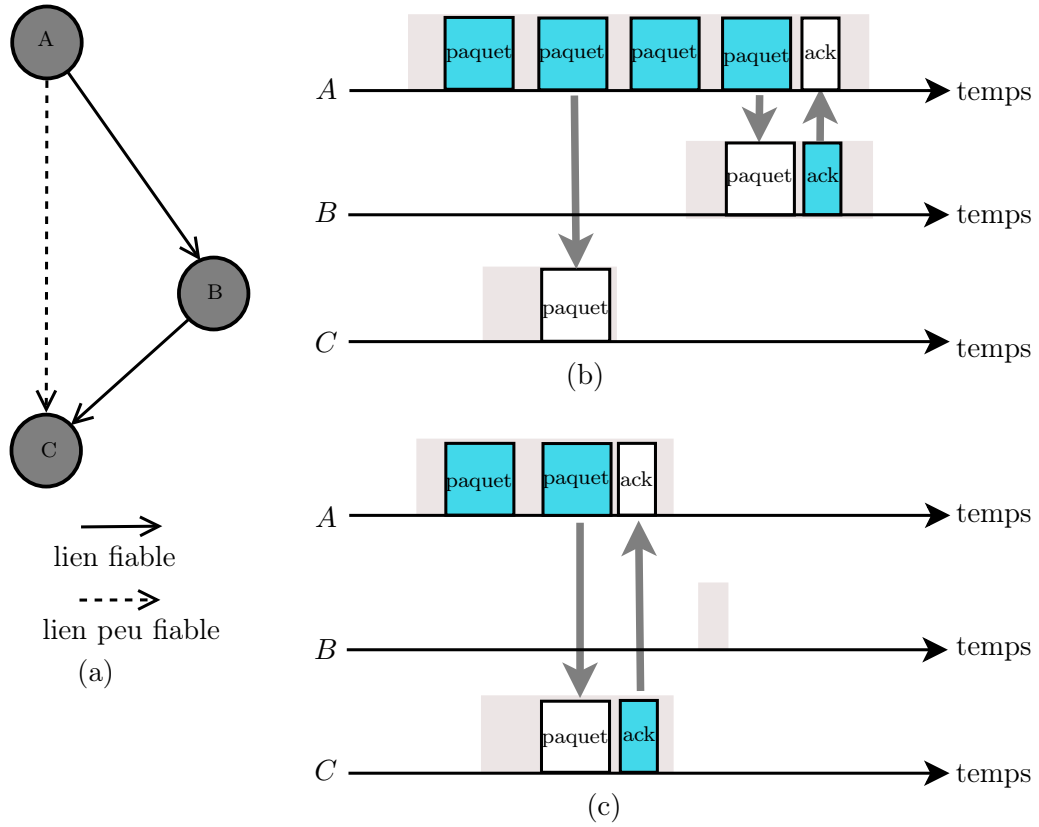


Figure 2.26 – Mécanisme de routage dans ORW.

métrique de transmission notée EDC (pour *Expected Duty Cycled wakeup*) qui est une extension de ETX [DCABM05]. EDC décrit le temps d'activité (c'est-à-dire la somme des temps d'attente avant le réveil des relais potentiels) pour qu'un paquet atteigne sa destination. Ainsi, un nœud sélectionne uniquement comme prochain saut le nœud qui fournit strictement plus de progrès que lui-même. Dès lors, les nœuds auront plusieurs choix de chemins de routage, ce qui diminue le délai de bout en bout pour qu'un paquet atteigne sa destination.

Dans FTSP [LR10, LR13] (pour *Fast Time-dependent Shortest Path algorithm*), un algorithme distribué inspiré de l'algorithme décrit dans [OR96] est proposé pour construire et maintenir des chemins avec un minimum de retard dans les réseaux de capteurs sans fil à taux d'activité asynchrones. FTSP nécessite une phase d'initialisation permettant de construire un arbre couvrant dont le puits est la racine et dans lequel chaque nœud connaît son père. Après la phase d'initialisation, si un nœud détecte une diminution ou une augmentation du coût en terme de distance signalé par un voisin, ce nœud est libre de choisir un nouveau parent. Les auteurs font une première proposition d'algorithme de routage basé sur le mécanisme d'écoute du LPL (pour *Low Power Listening*) comme dans le protocole B-MAC ou X-MAC. Chaque

nœud se réveille périodiquement pour écouter le canal et vérifier si un nœud voisin veut lui envoyer des paquets.

Dans [LR13] les mêmes auteurs proposent de baser FTSP sur le mécanisme ALPL (pour *Adaptive Low Power Listening*) décrit dans [JBL07] dans lequel chaque nœud détermine périodiquement en local son propre taux d'activité en fonction de sa charge de trafic et le diffuse à son voisinage dans le but d'atteindre une consommation énergétique plus uniforme. FTSP nécessite que chaque nœud maintienne un vecteur de distance et qu'il ait une connaissance du coût en temps des liens et les dates de réveil de chaque nœud dans son voisinage. En prenant en compte les dérives d'horloges, la mobilité, et le fait que les nœuds capteurs ont des ressources limitées, l'obtention de ces connaissances devient coûteuse.

Dans ASSORT (pour *Asynchronous Sleep-wake Schedules and Opportunistic Routing*) [HKWC14], chaque nœud possède son propre calendrier et alterne entre périodes d'activité et d'inactivité. La figure 2.27 illustre l'exemple d'un nœud émetteur u qui a des données à transmettre à un nœud récepteur v. Le nœud u suspend son inactivité et se réveille immédiatement. Ensuite, avant le réveil du récepteur attendu, le temps est divisé en petites périodes notées Δ_{reveil} appelées périodes de sondage du canal. Le nœud u diffuse une balise au début de chaque période de sondage (T_b correspond au temps d'envoi de cette balise) et attend un accusé de réception. Le nœud u commence la transmission de son paquet s'il reçoit un accusé de réception d'un relais potentiel après la diffusion de sa balise. Si aucun relais n'envoie un accusé de réception pendant un temps T_s, le nœud u retourne en mode d'inactivité le reste du temps Δ_{reveil} et se réveiller au début de la prochaine période Δ_{reveil} pour un nouveau sondage. Lorsqu'un nœud récepteur v se réveille, il écoute le canal pendant un temps Δ_{reveil} qui correspond à la période de réveil. S'il reçoit une balise d'un nœud u appartenant à son ensemble de nœuds dont il est un relais potentiel, il envoie un accusé de réception à u (T_a correspond au temps d'envoi de l'accusé de réception) et reste actif pour recevoir les données de u. En outre, pour éviter qu'un nœud récepteur se réveille et ne détecte pas de balise, l'intervalle entre la diffusion de deux balises est fixé à la période de réveil du récepteur Δ_{reveil}. Les relais potentiels pour un nœud sont sélectionnés suivant une métrique de routage appelée OECS (pour *Opportunistic Energy Cost with Sleep-wake schedules*) afin de permettre un routage opportuniste qui prolonge la durée de vie du réseau. Les auteurs proposent que la métrique OECS soit calculée lors de l'initialisation du réseau à partir du nœud puits qui commence la diffusion avec une métrique OECS égale à 0. Chaque nœud qui reçoit une métrique recalcule sa propre métrique (initialement fixée à $+\infty$) et la rediffuse à ses voisins. Dès lors, à la fin de cette phase initiale, chaque nœud connaît sa métrique et celle de

tous ses voisins. Les relais potentiels d'un nœud sont ses voisins qui offrent la plus petite métrique jusqu'au puits.

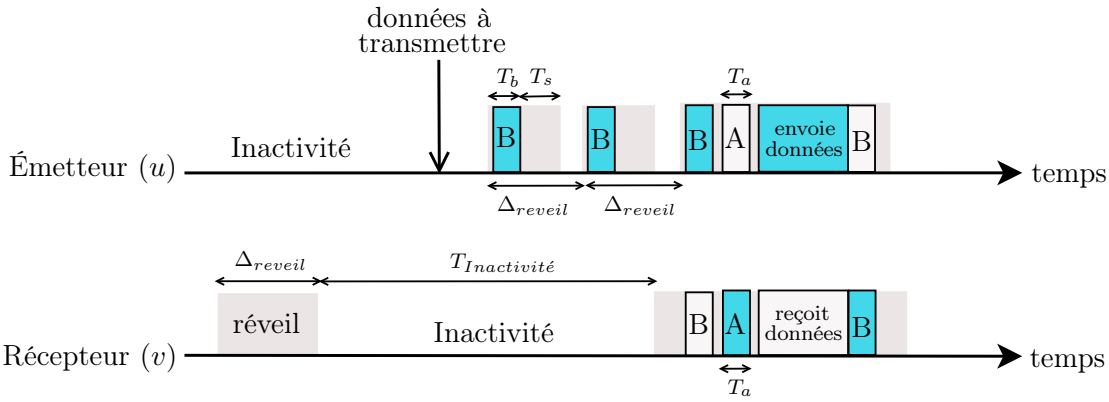


Figure 2.27 – Illustration du mécanisme d'activité et d'inactivité dans ASSORT

Le mécanisme de routage dans ASSORT vise à utiliser le plus court chemin pour transmettre les données des nœuds vers le puits. Cependant, la planification des périodes d'activité et d'inactivité dans ASSORT ne résout pas le problème d'inéquité dans la consommation énergétique des protocoles asynchrones car un nœud a besoin de se réveiller plusieurs fois (pendant un temps $T_b + T_s$ avec $T_s = 100 * T_b$, où T_b représente le temps d'envoi d'une balise) par cycle pour pouvoir acheminer ses données. Aussi, ASSORT n'est pas adapté à un fonctionnement à faible taux d'activité vu que même les récepteurs ont besoin de rester actifs pendant un temps Δ_{reveil} bien supérieur à T_s, même si aucune activité ne le concerne.

Le travail réalisée dans [WPB+09] est un protocole de routage par gradient normalisé par le groupe de travail ROLL (pour *Routing Over Low power and Lossy network*) de l'IETF (pour *Internet Engineering Task Force*) comme protocole fondamental pour la collecte des données dans les réseaux de capteurs sans fil. Il est appelé RPL (*IPv6 Routing Protocol for Low power and lossy networks*) [WT12]. RPL propose un mécanisme de collecte de données des nœuds vers le puits. Dans RPL, un gradient est construit à partir du puits et est mis à jour périodiquement. RPL nécessite une mise à jour périodique des gradients pour faire face aux changements topologiques. En effet, prévoir une période spécifique de mise à jour, suppose qu'on peut prévoir à quel moment il peut y avoir des variations dans le réseau. Aussi, pendant chaque phase de réinitialisation du gradient, la transmission des données doit être stoppée et tous les nœuds doivent être actifs, ce qui va générer une consommation supplémentaire d'énergie. [WPB+09] montre que la proposition normalisée par l'IETF ROLL est facile à mettre en œuvre, et apporte un mécanisme de découverte de voisinage au niveau de la couche MAC pour éviter la mise à jour périodique des

gradients. De ce fait, les auteurs proposent que les informations sur le gradient soient

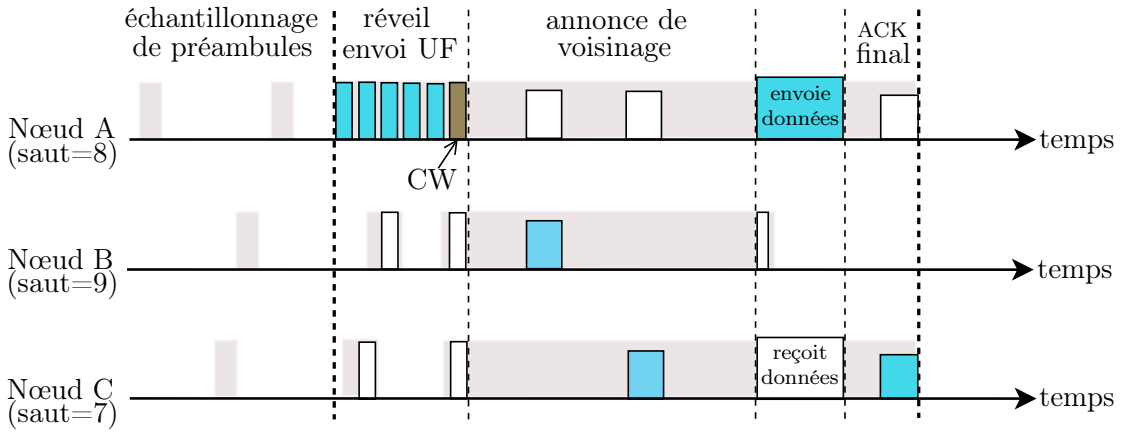


Figure 2.28 – Illustration du mécanisme de réveil et de transmission des données dans [WPB+09].

intégrées dans les paquets de données afin d'éviter les messages de contrôle et les mises à jour périodiques. Ainsi, lors du démarrage du réseau, tous les nœuds fixent leur gradient noté *height* à une valeur infinie sauf le puits qui possède une valeur de *height* égale à 0. À chaque fois qu'un nœud a des données à transmettre ou à relayer, il construit une liste de son voisinage avec leur valeur de *height* et met à jour son propre gradient au minimum de ceux de ses voisins incrémenté de 1. Si tous ses voisins ont un saut infini, son saut reste fixé à une valeur inconnue. La couche réseau se charge de mettre à jour la valeur *height* des nœuds et la couche MAC fonctionne à la demande en se servant de l'échantillonnage de préambule. La figure 2.28 représente un exemple pour une topologie à 3 nœuds. Quand un nœud veut envoyer un paquet (ici le nœud A), il commence par l'envoi d'un préambule aussi long que l'intervalle de contrôle noté CI (pour *Check Interval*) pour s'assurer que tous ses voisins entendent ce préambule. Le préambule est découpé en une série de micro-paquets notée UF, contenant chacune un compteur indiquant le nombre d'UF à venir. L'expéditeur envoie un paquet CW comportant la taille de la fenêtre d'annonce de voisins à la fin de l'envoi des UF. Lorsqu'un nœud récepteur potentiel entend un paquet UF, il éteint sa radio pour se réveiller juste après le dernier paquet UF et recevoir le paquet CW, puis choisit un temps d'attente aléatoire après lequel il envoie son annonce à travers un paquet ACK. L'expéditeur écoute les ACK pendant toute la période d'annonce et remplit sa table de voisinage (initialement vide). Après la fenêtre d'annonce de voisins, l'expéditeur met à jour sa valeur de *height* par la valeur minimale de ses voisins incrémentée de 1 et sélectionne le voisin ayant le plus petit *height* (ici C) comme relais. Ensuite, il insère ces informations dans l'entête du paquet puis le transmet. Après la réception de l'entête, les nœuds non relais (ici B) passent en mode d'inacti-

vité tandis que le relais choisit (ici C) attend pour recevoir le paquet de données. À la fin de la réception, le relais choisi répond par un paquet ACK final qui marque la reprise de l'échantillonnage de préambule.
Dans [WPB+09], le processus d'échantillonnage est répété pour chaque transmission (car la liste construite est supprimée à la fin de la transmission) et lorsqu'un nœud a des paquets à émettre, il doit attendre la fin de l'échantillonnage en cours avant d'en faire. Cela augmente le délai de bout en bout pour la livraison des données. Aussi, la longue durée de réveil lorsque les nœuds veulent effectuer une transmission fait que [WPB+09] n'est pas adapté à de faibles taux d'activité.

D'autres travaux comme [VRG+14] et [AWA11] sont basés sur des solutions exploitant l'interaction de RPL avec la couche MAC (cross-layer) pour améliorer l'efficacité du routage. Dans [VRG+14], les auteurs proposent un mécanisme de cross-layering basée sur la norme IEEE 802.15.4 en mode avec suivi de balises et le protocole de routage RPL. Leur mécanisme fournit à RPL un moyen de transmission des informations de routage à la couche 2 avant la création de la topologie dans 802.15.4 en encapsulant les messages RPL DIO dans les trames balises. Les auteurs de [AWA11] examinent l'utilisation d'un protocole MAC *receiver initiated* pour améliorer les performances de RPL.

2.2.3 Protocoles de routage opportuniste par inondation

De façon classique dans les protocoles de routage par inondation, un nœud particulier doit communiquer un élément d'information aux autres nœuds du réseau [HHL88]. Chaque fois qu'un nœud reçoit un paquet de données, il effectue une copie et le renvoie à tous ses voisins à l'exception de celui de qui il vient de le recevoir. Cela engendre une explosion ou un chevauchement de la même information. Différents protocoles ont été proposés pour faire face au problème d'explosion ou de chevauchement. Par exemple certains auteurs se servent d'une structure logique en arbre pour la diffusion des données.

Guo *et al.* ont proposé [GGJH09, GGJH14], un protocole de routage opportuniste par inondation fonctionnant avec un faible taux d'activité et des liens peu fiables noté *OppFlood*. Dans *OppFlood*, chaque nœud possède son propre calendrier pour passer en état actif et en état de sommeil, représentés respectivement par 1 et 0 sur la figure 2.29. Le temps est divisé en intervalles de temps de longueur appropriée pour envoyer un paquet de données et recevoir un ACK et chaque nœud prend un ou plusieurs intervalles pour son état actif, en conformité avec son cycle. Les calendriers sont ensuite échangés entre les voisins. La figure 2.29 montre un exemple de fonc-

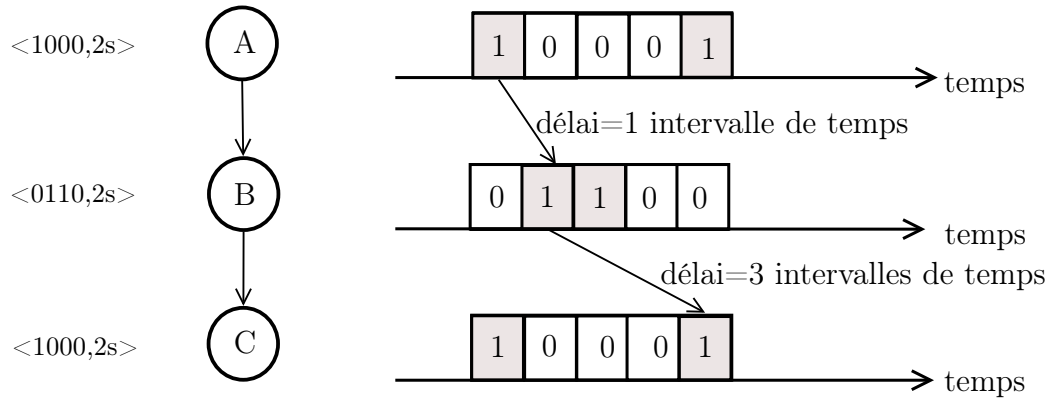


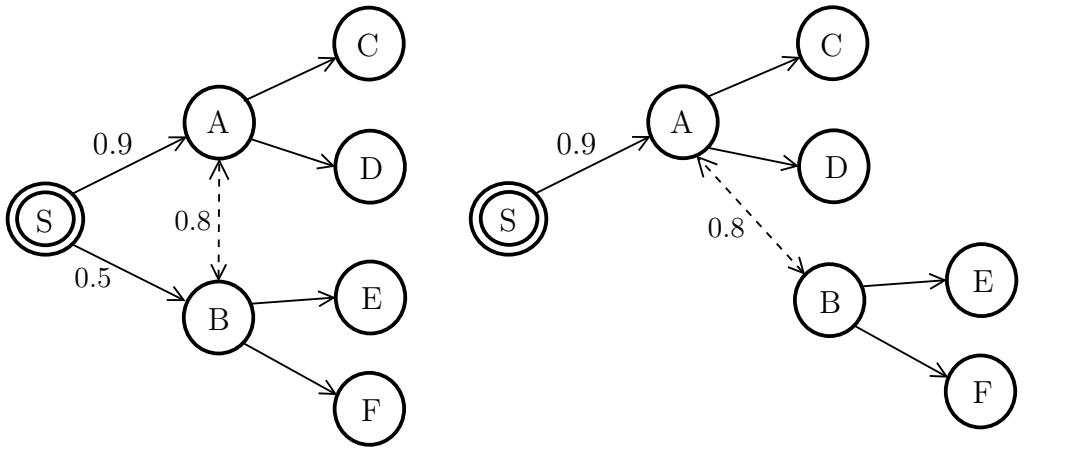
Figure 2.29 – Exemple de planification de communication dans *OppFlood*

tionnement dans *OppFlood*. La durée d'un cycle est de 8 secondes et est divisée en 4 unités de temps qui durent 2 secondes chacune. Par exemple A est actif pendant la première unité de temps et reste en inactivité les trois unités de temps restantes. A va attendre 1 intervalle de temps pour communiquer avec B qui attendra à son tour 3 intervalles de temps pour communiquer avec C si les liens entre A et B et entre B et C sont parfaits. Sinon, le nœud devra attendre le prochain cycle pour tenter de communiquer avec ce voisin. L'idée maîtresse de *OppFlood*, consiste à prendre des décisions de transfert probabilistes basées sur la distribution du délai d'attente des voisins. Le réseau est organisé comme un arbre de transmission optimal en énergie. Les auteurs supposent que les nœuds sources ont des paquets d'inondation à envoyer à travers l'ensemble du réseau et que les paquets ne sont transmis qu'à partir des nœuds plus proches du puits vers ceux qui sont plus éloignés. Les paquets sont donc transmis *via* un arbre optimal d'énergie des nœuds parents vers les nœuds fils en vue de réduire le nombre de copies des données.
Ce protocole réduit le délai d'inondation en exploitant également des liens en dehors de l'arbre optimal (liens peu fiables) pour transmettre les nouveaux paquets. Cependant, ce protocole ne fonctionne que pour des applications avec une communication du type un pour plusieurs (noté *One-to-Many*) et pour des réseaux statiques. Aussi, les auteurs supposent une synchronisation locale pour les communications, ce qui nécessite une consommation supplémentaire d'énergie.

Dans [BST11], les auteurs exploitent la corrélation des adresses pour améliorer la durée de vie du réseau. D'autres auteurs, exploitent la corrélation entre les liens pour réduire la latence et la consommation énergétique de l'inondation [ZZHZ10, GKZ+11]. En effet, les auteurs proposent que les nœuds ayant des liens fortement corrélés (à portée de transmission d'un même nœud) soient affectés à un émetteur commun et que leurs réceptions d'un paquet de diffusion soient reconnus par un seul ACK.

Dans [ZZHZ10], un algorithme probabiliste d'inondation est proposé. Cet algorithme permet de réduire la consommation énergétique pendant la transmission des données en utilisant un seul ACK implicite déduit de la corrélation entre les liens. Les auteurs de [GKZ+11] proposent de tenir compte de la corrélation entre les liens en plus de la qualité des liens dans la construction de l'arbre d'inondation, afin de rendre l'inondation plus économe en énergie.

Cheng *et al.* ont proposé DSRF [CGHN13], un protocole d'inondation pour les réseaux de capteurs à faible taux d'activité en fonction des commutations dynamiques dans une topologie en arbre. Dans DSRF, les auteurs proposent que sur un arbre d'inondation (comme l'arbre d'inondation basé sur le nombre de sauts dans [JMB01], l'arbre d'inondation dans [WL12], ou encore l'arbre d'inondation optimal en énergie de [GGJH09]) les nœuds ayant un même parent se réveillent ensemble pour recevoir simultanément les données de leur père. Aussi, ils proposent qu'en cas d'échec de réception, une décision de commutation dynamique soit prise par le récepteur afin qu'il puisse recevoir les données à partir des nœuds de la fratrie. Sur la figure 2.30 (a), le numéros représentent la qualité du lien entre les nœuds (elle varie entre l'intervalle [0,1]). Par exemple si le nœud B manque les données de son père S, au lieu d'attendre la prochaine diffusion (qui peut être longue vu le taux d'activité faible des nœuds), il peut recevoir les données à partir du nœud A comme indiqué sur la figure 2.30 (b). Ce qui permet de réduire efficacement le temps de latence de bout en bout.



(a) structure de l'arbre d'inondation initial (b) structure de l'arbre d'inondation ajusté dynamiquement

Figure 2.30 – Exemple d'un cas de commutation de liens dans DSRF.

Dans Flash [LW09], les auteurs se servent des transmissions simultanées pour permettre une inondation rapide et efficace. Flash se sert de l'effet de capture (qui est la capacité de certaines radios à recevoir un signal provenant d'un émetteur malgré des interférences provenant d'un autre émetteur, même si les puissances relatives des

deux signaux sont pratiquement les mêmes [LF76]) pour réaliser une inondation rapide dans les réseaux de capteurs sans fil. Flash suppose que les données transmises dans le réseau sont essentiellement les mêmes et veille à ce que chaque nœud reçoive le paquet d'inondation d'au moins un de ses voisins. Cependant, Flash s'appuie exclusivement sur les effets de capture, ce qui réduit considérablement les chances de recevoir correctement un paquet lorsque de nombreux nœuds transmettent en même temps.

Dans CIRF [YWW+14], une approche de routage par inondation qui ne nécessite pas que les nœuds soient synchronisés est proposée. Les auteurs exploitent l'interférence constructive des nœuds capteurs. L'interférence constructive provient du scénario dans lequel plusieurs émetteurs envoient simultanément un paquet identique à un récepteur commun. L'interférence constructive permet potentiellement de réduire la puissance de transmission, donc de réduire la consommation énergétique des transmissions, et en même temps d'augmenter la puissance de réception des données. La seule condition préalable est que le décalage entre les transmissions simultanées ne dépasse pas le seuil de $0.5\mu s$ fixé dans [WHM+12] pour les récepteurs compatibles avec la norme IEEE 802.15.4 ou le seuil définit dans [WHC+12] pour les nœuds capteurs *TMote Sky*. Les auteurs de CIRF ont choisi le mécanisme d'accès de RI-MAC [SGJ08] comme protocole MAC et supposent que la couche MAC fournit le service d'estimation de qualité de liaison comme dans [KS06]. Un mécanisme appelé AF (pour *Afore-Frame*) est ajouté à RI-MAC pour tous les nœuds. Pendant la phase de découverte de voisinage, chaque nœud définit la longueur de sa période AF en fonction de son nombre de voisins comme l'indique la figure 2.31. Dans CIRF, au

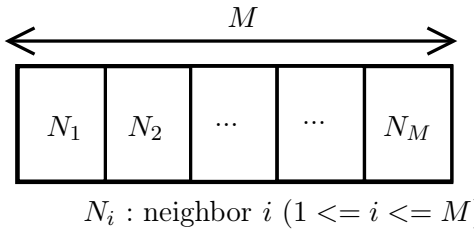


Figure 2.31 – Structure d'un AF de taille M.

lieu que les nœuds émetteurs qui attendent un même récepteur diffusent leur paquet juste après la réception de la balise du récepteur, ces nœuds attendent tous pendant la période AF du récepteur (inclus dans la balise reçue) et émettent simultanément leur paquet de données (voir figure 2.32).

La figure 2.32 montre un exemple d'interférence constructive dans CIRF. Au début, les émetteurs S_1 et S_2 attendent tous les deux le même récepteur R. Après

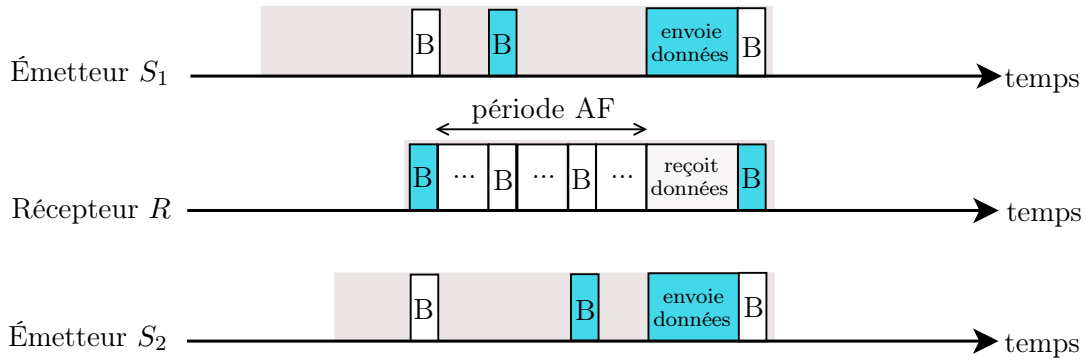


Figure 2.32 – Mécanisme de transmission dans CIRF.

le réveil et l'envoi de la balise de R, vient la période l'AF, dans laquelle S_1 et S_2 envoient chacun une balise à leur instant respectif. Une fois que la période AF se termine, S_1 et S_2 transmettent simultanément leur paquet de données. R reçoit le paquet avec une probabilité plus élevée en raison de l'interférence constructive des transmissions simultanées.

La plupart des protocoles de routage par inondation de la littérature fonctionnent uniquement pour des applications avec une communication d'une source (par exemple le puits) vers les autres nœuds du réseau *(one-to-many)*. Zhang et Fromherz ont proposé dans [ZF06] un protocole de routage par inondation pour réseaux de capteurs sans fil appelé inondation avec contrainte noté CF (pour *Constrained Flooding*), pour une communication des nœuds vers le puits *(many-to-one)*. Dans CF, un gradient appelé *cost-to-go* est calculé selon les objectifs du routage (par exemple, pour assurer le plus court chemin des nœuds vers le puits). Chaque nœud n maintient son estimation $c(n)$ du *cost-to-go* et de celui de ses voisins. À chaque fois qu'un nœud n entend un paquet d'un voisin v, qu'il soit le récepteur attendu du paquet ou non, n met à jour la valeur *cost-to-go* de ce voisin $c_n(v)$ et ré-estime son propre *cost-to-go* $c(n)$ en utilisant la formule suivante : $c(n) \leftarrow (1-\alpha)c(n) + \alpha(o(v) + \min_v c_n(v))$ où α $(0 < \alpha \leq 1)$ est le taux d'écoute et o est la fonction de coût. CF ne nécessite pas de mise à jour périodique pour ré-estimer le *cost-to-go* en cas de changement du réseau.

La figure 2.33 donne l'organigramme des règles à respecter pour le protocole CF avant la rediffusion d'un paquet reçu par un nœud n d'un voisin v.

1. Si n est le destinataire du paquet, alors le paquet est remonté à la couche supérieure. Sinon, (1) mettre à jour $c(n)$, T et $c_n(v)$. La différence entre le *cost-to-go* $c(n)$ du nœud et celui du voisin $c_n(v)$ doit être inférieure à une valeur appelée température et notée T: $c(n) - c_n(v) < T$. Cette variable température est fixée à une constante initialement grande et réduite graduellement, par exemple $T \leftarrow kT/(k+1)$, où $k = T_0$ est la température initiale.

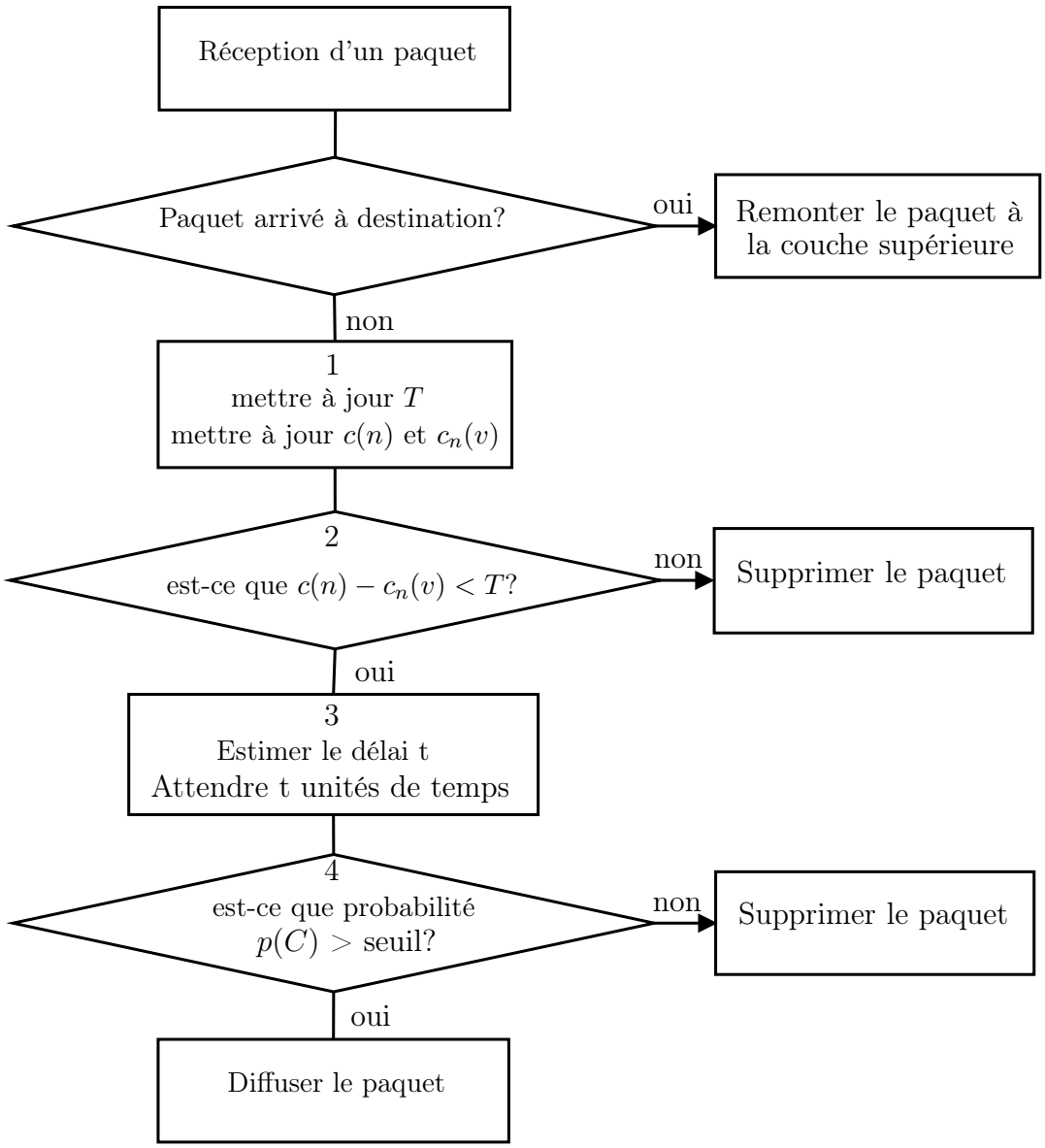


Figure 2.33 – Organigramme du protocole de routage par inondation présenté dans CF.

2. (2) La source du paquet doit vérifier si $c(n) - c_n(v) < T$. Si c'est le cas, l'algorithme passe à l'étape 3.
3. (3) Afin d'éviter les collisions, un délai est fixé avant chaque rediffusion de paquet. Ce délai dépend de la différence entre le *cost-to-go* de l'émetteur et celui du récepteur, noté Δ : $\Delta = c(n) - c_n(v)$. Plus Δ est grand, plus le délai avant la diffusion est petit. Par exemple, la fonction de délai δ peut être exprimée par $\delta(\Delta) = D/e^{\Delta}$, où D est une constante.
4. (4) Une politique probabiliste est utilisée pour décider ou non de transmettre un paquet. Plus un paquet est entendu par un nœud, moins il a de chances d'être diffusé: la probabilité de diffusion peut s'exprimer par $p(C) = 1/C^{\gamma}$, où C est le nombre de fois où le paquet a été entendu et $\gamma \leq 0$ est le compromis entre robustesse et énergie.

CF permet de diminuer le nombre de copies de paquets dans le réseau tout en assurant que les paquets des nœuds ayant un *cost-to-go* élevé aient des chances d'être retransmis par les voisins ayant un *cost-to-go* plus faible. Cependant, le fait que les nœuds changent automatiquement leur *cost-to-go* dès qu'ils entendent un autre nœud avec un meilleur *cost-to-go* peut conduire à des pertes de paquets. En effet, un nœud n avec un *cost-to-go* $c(n) = 4$ peut entendre très rarement un voisin v avec un *cost-to-go* $c_n(v) = 2$ et ainsi réduire son propre *cost-to-go* à $c(n) = 3$. Cela peut conduire à ce que la plupart des paquets de n soient perdus, car n entend rarement ses relais potentiels (voisins v avec $c_n(v) = 2$). De plus, le mécanisme de délai avant diffusion du paquet proposé dans CF n'a pas de composante aléatoire, ce qui peut augmenter le nombre de collisions. Par exemple, deux voisins qui reçoivent un paquet quelconque pour la première fois vont avoir exactement le même délai avant l'émission du paquet, D étant fixe. Comme il n'y a pas d'acquittement, la non réception des paquets n'est pas constatée par les émetteurs.

2.2.4 Bilan sur les protocole de routage

Le tableaux 2.2 résume les principaux protocoles de routage opportunistes par gradient et par inondation de la littérature que nous avons passés en revue, avec les avantages et les inconvénients dans chaque catégorie.

Tableau 2.2 – Tableau récapitulatif des protocoles de routage opportunistes à taux d'activité

Catégories	**Protocoles**	**Avantages**	**Inconvénients**
Par gradient	ExOR [BM05], SGF [HCXT09], ORW [LGDJ12, GLS+14], [LR13], ASSORT [HKWC14], RPL [WPB+09]	le chemin avec le coût minimum est utilisé pour transmettre les données, communications en *unicast* (favorise l'agrégation des données), assure une faible latence	nécessite de construire et de maintenir le gradient pour chaque nœud, surcharge en messages de contrôle pour la mise à jour périodique du gradient, adapté uniquement aux applications avec communications dirigées
Par inondation	OppFlood [GGJH09, GGJH14], [ZZHZ10, GKZ+11], DSRF [CGHN13], Flash [LW09], CIRF [YWW+14], CF [ZF06]	facile à mettre en œuvre, exploite tous les chemins, faible latence, ne nécessite pas forcement de construire et de maintenir une table de routage	nombre important de copies des données, forte charge réseau, nécessite parfois que les nœuds soient synchronisés, très souvent pas adapté aux communications *many-to-one*

Chapitre 3

Nos différentes contributions

Les contributions principales de ce travail visent la sous-couche MAC et la couche réseau du modèle OSI. Dans cette partie, nous proposons dans un premier temps, des protocoles MAC non synchronisés qui peuvent fonctionner avec de faibles taux d'activité (moins de 1 % d'activité). Dans un second temps, nous proposons des protocoles de routage adaptés aux protocoles MAC à faible taux d'activité.

3.1 Protocole MAC asynchrone à faible taux d'activité

La couche MAC est concernée par la consommation énergétique du module radio et du micro-processeur. Dès lors, pour minimiser la consommation énergétique globale dans un RCSF, il est primordial de concevoir des protocoles MAC économes en énergie, donc qui utilisent le moins possible le module radio. Dans les protocoles MAC synchrones, les nœuds se réveillent ensemble périodiquement pour communiquer et se rendorment ensuite pour économiser de l'énergie. C'est le cas avec la norme IEEE 802.15.4 [IEE11] dont le fonctionnement est représenté sur la figure 3.1. Nous notons cette approche A_{sp} (pour approche synchrone et périodique). L'approche A_{sp} permet aux nœuds de partager systématiquement une activité commune pendant toute la durée de la supertrame (SD) et d'économiser tous ensemble de l'énergie pendant $BI - SD$ unités de temps (reste de l'intervalle de balise BI). Ainsi, tous les nœuds ont une consommation énergétique fixe, ce qui donne la possibilité de prédire la durée de vie du réseau.

Cependant, la synchronisation des nœuds est difficile à réaliser et coûteuse (en

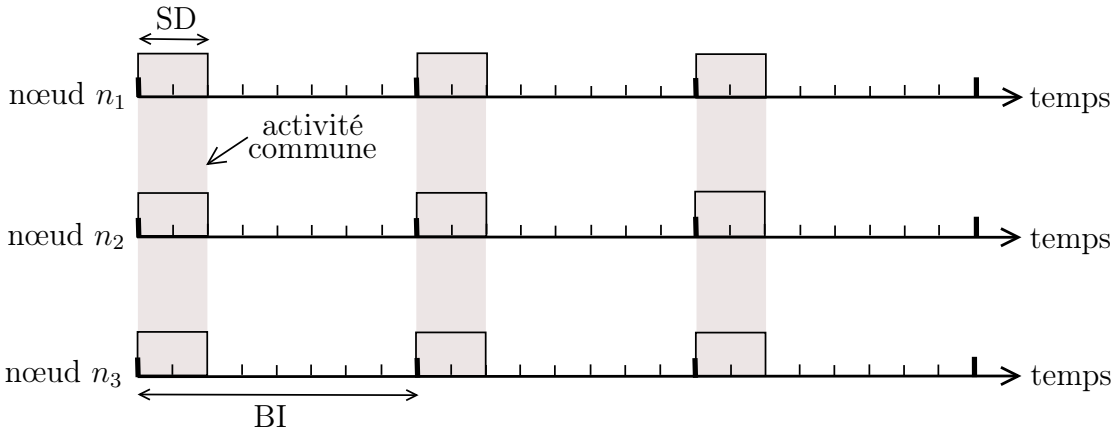


Figure 3.1 – Exemple d'activité synchronisée de trois nœuds n_1, n_2 et n_3, avec un taux d'activité de 25 % (ce grand taux d'activité de 25 % est utilisé par mesure de clarté).

termes d'énergie ou d'utilisation de la bande passante déjà limitée), notamment quand il y a un nombre important de nœuds dans le réseau. La synchronisation augmente aussi la contention pour l'accès au médium et les collisions en début d'activité [KAT06][DDB13]. De plus, des frais supplémentaires sont nécessaires pour mettre en œuvre la synchronisation, ce qui limite le fonctionnement des protocoles MAC synchronisés avec un faible taux d'activité (de l'ordre de 1 %).

Notre question fondamentale est : comment utiliser un fonctionnement proche de celui de la norme IEEE 802.15.4 avec suivi de balises sans synchronisation? ou, comment utiliser la norme IEEE 802.15.4 sans suivi de balises tout en économisant l'énergie?

Pour répondre à cette question, nous proposons tout d'abord des mécanismes pour la planification de l'activité des nœuds. Ensuite, nous proposons un protocole MAC basé sur le plus performant de ces mécanismes. Enfin, nous apportons des améliorations au protocole MAC en vue d'augmenter ses performances.

3.1.1 Mécanismes de planification de l'activité des nœuds

En éliminant la procédure de synchronisation de la norme IEEE 802.15.4 avec suivi de balises, il est possible d'envisager une approche où les activités des nœuds sont périodiques, mais non synchronisées. Nous désignons cette approche comme asynchrone périodique, avec un BI commun, et nous la notons A_{ap}. Dans ce qui suit, nous décrivons un tel mécanisme et nous montrons ses inconvénients. Ensuite, nous proposons un mécanisme avec une approche asynchrone périodique avec des BI et SD différents et nous la notons A'_{ap}. Enfin, nous proposons un autre mécanisme avec une approche asynchrone apériodique et cyclique, et nous la notons A_{aa}.

3.1.1.1 Approche asynchrone périodique avec un BI commun : A_{ap}

Dans A_{ap}, le début de l'activité de chaque nœud est indépendant comme le montre la figure 3.2. Les rencontres se font de manière aléatoire, selon les débuts des périodes des nœuds.

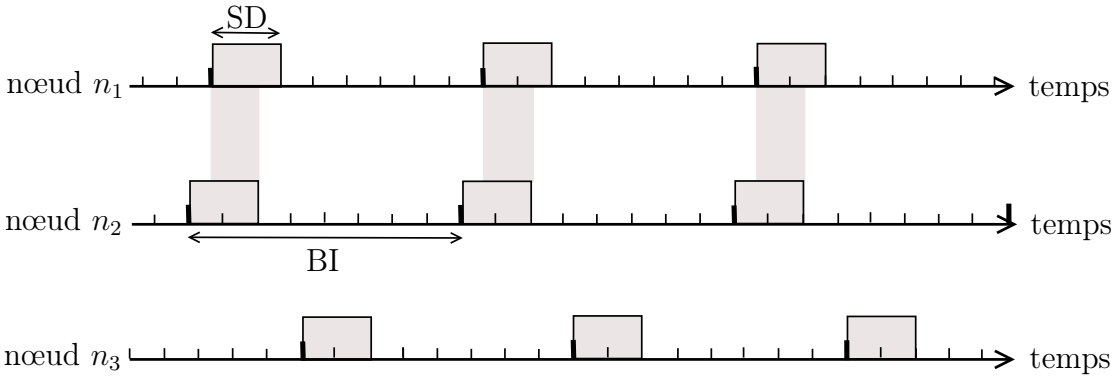


Figure 3.2 – Exemple d'activité de trois nœuds n_1, n_2 et n_3 non synchronisés, avec un taux d'activité de 25 %.

La figure 3.2 représente les activités de trois nœuds, avec un taux d'activité de 25 %. Les intervalles de balises sont désynchronisés. Pour simplifier, nous faisons l'hypothèse que $BI = 8$ intervalles de temps, comportant chacun plusieurs périodes de *backoffs*. En général, BI comprend un très grand nombre de périodes de *backoffs* : pour un BI de $2^6 \times 15.36$ ms (ce qui correspond à environ une seconde), il y a 3072 périodes de *backoffs* de 320 μs chacune comme dans la norme IEEE 802.15.4 [IEE11]. Les nœuds n_1, n_2 et n_3 ont le même BI, mais démarrent indépendamment leur activité pendant SD unités de temps (ici, 768 périodes de *backoffs*). Dans l'exemple représenté, périodiquement, les nœuds n_1 et n_2 partagent une période d'activité commune pendant une durée inférieure à SD unités de temps (à la fin de l'activité de n_1 et au début de l'activité de n_2) et n_3 ne partagent pas d'activité commune avec n_1 ou n_2.

Dans la suite, nous étudions la probabilité de rencontre entre deux nœuds et le délai moyen avant qu'un nœud ait une rencontre avec un autre nœud dans son voisinage.

Étude de la probabilité de rencontre avec A_{ap}

Nous cherchons à déterminer la probabilité pour que les activités de deux nœuds voisins soient disjointes (c'est-à-dire que les nœuds ne partagent jamais d'activité commune). Notons n_i le $i^{ème}$ nœud, BI la durée de l'intervalle de balise, et $\alpha \in]0;1]$ le taux d'activité (chaque nœud étant actif pendant $SD = \alpha \times BI$ unités de temps).

On suppose par exemple que n_1 commence son activité au début de son intervalle de balise (dans ce cas, n_1 termine son activité à $\alpha.BI$). Soit $P_{disjoint}$ la probabilité que n_1 et n_2 aient des activités disjointes. Si $\alpha > 1/2$, $P_{disjoint}$ est toujours égal à zéro, étant donné que les nœuds sont actifs pendant plus de la moitié de l'intervalle de balise. Par contre, si nous considérons $\alpha \leq 1/2$, $P_{disjoint}$ est égal à la probabilité que n_2 commence son activité après que n_1 ait terminé son activité, ou que n_2 termine son activité avant que n_1 commence. Ainsi, $P_{disjoint}$ est la probabilité que n_2 commence son activité dans l'intervalle $[\alpha.BI; BI - \alpha.BI[$. Cet intervalle existe toujours, étant donné que $\alpha \leq 1/2$ entraîne que $\alpha.BI \leq BI - \alpha.BI$. Comme nous faisons l'hypothèse que le temps de démarrage de n_2 est uniformément distribué sur $[0; BI - \alpha.BI[$, on a : $P_{disjoint} = (BI - 2\alpha.BI)/BI$, ce qui donne : $P_{disjoint} = 1 - 2 \times \alpha$. Par exemple, lorsque $\alpha = 1/4$, $P_{disjoint} = 1 - (2 \times 1/4) = 1/2$ et quand $\alpha = 1/8$, $P_{disjoint} = 1 - (2 \times 1/8) = 3/4$. Lorsque le taux d'activité est faible, la probabilité que certains nœuds ne partagent jamais une activité commune est donc élevée, ce qui est un inconvénient important.

Déterminons ensuite la probabilité P_{all} pour que les nœuds soient tous actifs à un même instant, avec n le nombre de nœuds à portée de communication. Étant donné que les activités des nœuds sont choisies indépendamment, nous pouvons calculer P_{all} comme suit :

$$P_{all} = \prod_{i \neq 1} P(n_i \text{ est actif dans } [0; \alpha BI[) = \prod_{i \neq 1} \alpha = \alpha^{n-1}.$$

Par exemple, quand $\alpha = 1/2$ et qu'il y a $n = 3$ nœuds à portée de communication, $P_{all} = 1/4$. Quand $\alpha = 1/2$ et $n = 4$, $P_{all} = 1/8$ ou si $\alpha = 1/10$ et $n = 4$, $P_{all} = 10^{-3}$. On peut voir que lorsque le taux d'activité est faible et que le nombre de nœuds n est grand, la probabilité pour que tous les nœuds partagent en même temps une activité commune est très faible, ce qui représente aussi un inconvénient.

Étude du délai moyen avant rencontre avec A_{ap}

Il est important de déterminer le délai avant que deux nœuds partagent une activité commune, dans le cas où les nœuds se rencontrent. Nous définissons le délai moyen avant une rencontre entre deux nœuds n_1 et n_2 comme la durée moyenne entre un instant où n_1 est actif, et le premier instant où n_1 et n_2 sont actifs en même temps. Il faut noter que, parfois, n_1 et n_2 ne se rencontrent jamais : ces cas ne sont pas pris en compte dans le calcul du délai moyen (car sont considérés comme des délais infinis). Cependant, la probabilité que ce cas se produise est $P_{disjoint}$. Pour calculer le délai moyen, nous examinons les deux seuls cas susceptibles de se produire : le cas 1 se

produit quand n_2 commence son activité dans l'intervalle $[0; \alpha.BI[$ de n_1, et le cas 2 se produit quand n_2 commence son activité dans l'intervalle $[BI - \alpha.BI; BI[$. Ces deux cas sont représentés sur la figure 3.3. Notons que sous l'hypothèse de rencontre, il est

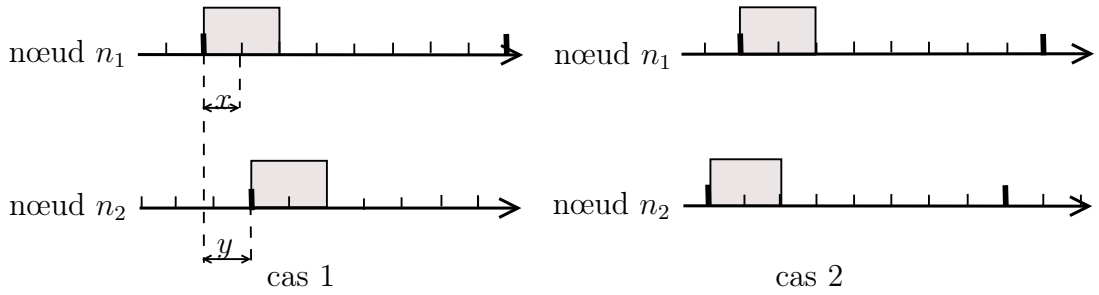


Figure 3.3 – Les deux cas possibles lorsque deux nœuds n_1 et n_2 se rencontrent dans A_{ap}.

impossible que n_2 commence son activité à d'autres moments, car cela provoquerait un délai infini. Nous considérons que le temps est discret et que la granularité du temps est par exemple, 320μs comme dans la norme IEEE 802.15.4 [IEE11].

Soit d_1 le délai moyen avant que n_1 et n_2 partagent une activité commune, pour le cas 1 de la figure 3.3. Ce délai représente toutes les attentes possibles du nœud n_1 (représentées par y) lors de sa période d'activité (représenté par $x \in [0; \alpha.BI[$). Le délai moyen pour que les deux nœuds partagent une activité commune s'exprime comme suit : $\frac{1}{\alpha^2.BI^2} \sum_{y=1}^{\alpha.BI-1} \left(\sum_{x=0}^{y-1} (y-x) + \sum_{x=y}^{\alpha.BI-1} 0 \right)$.
La somme pour x allant de 0 à $y-1$ prend en compte le temps d'attente du nœud n_1 juste avant le réveil du nœud n_2. La somme pour x allant de y jusqu'à $\alpha.BI - 1$ représente le temps d'attente (qui est égal à 0) quand les deux nœuds sont actifs simultanément. La composante $\frac{1}{\alpha^2.BI^2}$ représente la pondération par tous les instants de l'activité de n_1 (αBI) et également par ceux de n_2 (αBI). On obtient donc par calcul et par simplification [1] :

$$d_1 = \frac{(\alpha.BI - 1)(\alpha.BI + 1)}{6\alpha.BI}.$$

Soit d_2 le délai moyen avant que n_1 et n_2 partagent une activité commune dans le cas 2 de la figure 3.3. d_2 représente les attentes pour tous les instants $x \in [0; \alpha.BI[$ de l'activité de n_1 par rapport à toutes les avances possibles de l'activité de n_2 par rapport à celle de n_1. d_2 s'exprime comme suit :

1. Il faut noter que tous nos calculs ont été réalisés et simplifiés en utilisant le logiciel de calcul en ligne sur le site http://www.wolframalpha.com.

$$\frac{1}{\alpha^2.BI^2}\sum_{y=BI-\alpha.BI+1}^{BI-1}\left(\sum_{x=0}^{y+\alpha.BI-BI-1}0+\sum_{x=y+\alpha.BI-BI}^{\alpha.BI-1}(BI-x)\right).$$

La somme pour x allant de 0 jusqu'à $y+\alpha.BI-BI-1$ représente le temps d'attente (qui est égale à 0) quand les deux nœuds sont actifs simultanément. La somme de x allant de $y+\alpha.BI-BI$ à $\alpha.BI-1$ représente le délai avant la prochaine rencontre entre n_1 et n_2. On a donc :

$$d_2=-\frac{(\alpha.BI-1)(2\alpha.BI-3.BI-1)}{6\alpha.BI}.$$

Les cas 1 et 2 étant équiprobables, le délai moyen d est donc :

$$d=\frac{d_1+d_2}{2}=\frac{(\alpha.BI-1).(3.BI-\alpha.BI+2)}{12\alpha.BI}.$$

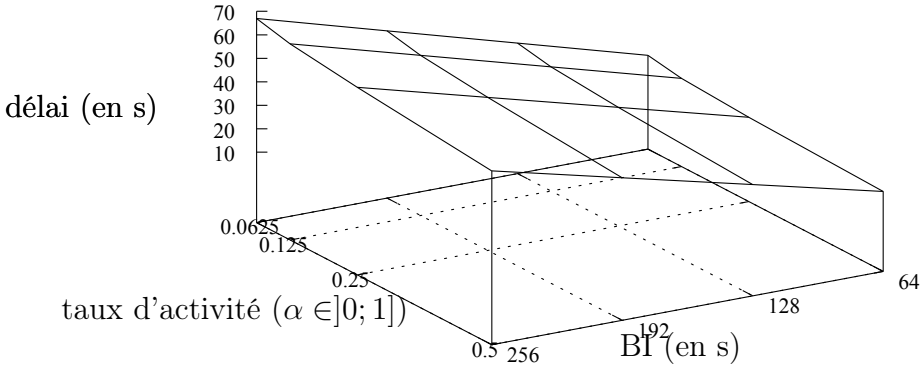


Figure 3.4 – Calcul numérique du délai d en fonction de BI et α.

La figure 3.4 montre le délai moyen d en fonction de BI et α. On peut noter que le délai moyen est relativement faible. Il atteint jusqu'à 70 s pour $BI = 256$ s et $\alpha = 6.25$ %.

L'inconvénient principal de l'approche A_{ap} est qu'elle génère une grande probabilité que des paires de nœuds dans un même voisinage ne se rencontrent jamais.

3.1.1.2 Approche asynchrone périodique avec BI et SD différents

Nous avons proposé dans [AGM13], d'utiliser des BI aléatoires pour chaque nœud, afin d'augmenter les possibilités de rencontre entre les nœuds et afin d'éviter que des paires de nœuds aient des activités systématiquement disjointes. Nous notons cette approche A'_{ap}. Dans cette partie nous décrivons l'approche A'_{ap} puis, nous proposons un mécanisme distribué à partir cette approche.

Description de l'approche A'_{ap}

L'approche A'_{ap} peut être résumée de la manière suivante :

- attribuer aux nœuds des BI différents,
- faire commencer les activités des nœuds indépendamment,
- faire en sorte que les nœuds maintiennent tous le même taux d'activité.

Le premier point permet de réduire la probabilité $P_{disjoint}$. Le deuxième point fait en sorte que tous les nœuds ne sont pas toujours actifs en même temps, ce qui réduit la contention d'accès au médium. Le troisième point permet à notre mécanisme de contrôler la durée de vie du réseau.

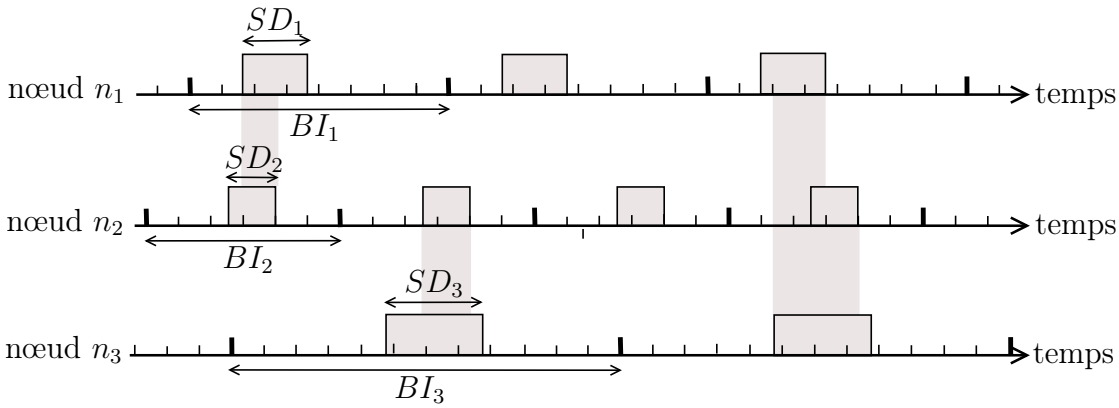


Figure 3.5 – Exemple d'activité de trois nœuds n_1, n_2 et n_3, dans l'approche A'_{ap}. Les activités sont non synchronisées, périodiques, avec des BI et SD différents pour un taux d'activité de 25 %.

La figure 3.5 montre les activités de trois nœuds n_1, n_2 et n_3 pour l'approche A'_{ap}. Chaque nœud opère avec un taux d'activité de 25 % même si chaque nœud a un BI différent. Les activités communes entre les nœuds ne sont pas périodiques : elles apparaissent en fonction du BI choisi par chaque nœud et de la date de démarrage de son activité. Par exemple, le nœud n_1 partage une activité commune avec le nœud n_2 au début de la première période d'activité de n_1, et à la fin de sa troisième période d'activité. La nœud n_3 partage également une activité commune avec le nœud n_2 au début de sa première période d'activité, et une activité commune en même temps avec les nœuds n_1 et n_2 à la fin de sa deuxième période d'activité. Toutes les possibilités d'activités communes entre les nœuds se produisent sur cet exemple.

Les principaux avantages de l'approche A'_{ap} sont les suivants :

- A'_{ap} ne nécessite pas de synchronisation entre les nœuds,
- la probabilité pour que deux paires de nœuds à portée de communication ne se rencontrent jamais ($P_{disjoint}$) est très réduite par rapport à l'approche A_{ap},

- A'_{ap} conduit à ce que peu de nœuds soient actifs en même temps (en moyenne), ce qui augmente les performances de l'accès au médium en réduisant la contention [2].

Mise en œuvre d'un mécanisme distribué à partir de A'_{ap}

Dans cette partie, nous décrivons comment construire un mécanisme distribué à partir de l'approche A'_{ap}.

L'algorithme 1 présente ce mécanisme. Pendant la phase d'initialisation, les nœuds partagent les valeurs suivantes : Δ qui représente une durée en périodes de *backoffs* (320 μs comme dans la norme IEEE 802.15.4), $minBI$ et $maxBI$ représentent respectivement la valeur minimale du BI et sa valeur maximale, et α est le taux d'activité. Nous supposons que l'activité des nœuds commencent indépendamment et aléatoirement. Chaque nœud i choisit son propre BI_i dans l'intervalle $[minBI; maxBI]$ à l'aide du calcul suivant : $4.\lfloor \text{rand}(minBI, maxBI)/4 \rfloor$ [3]. Après une durée de Δ périodes de *backoffs*, le nœud détermine s'il a rencontré un voisin. Si ce n'est pas le cas, il tire aléatoirement une nouvelle valeur pour BI_i et recommence.

```
Définir les constantes Δ, minBI, maxBI et α
neighbors ← 0
répéter
    BI_i ← 4 × ⌊rand(minBI, maxBI)/4⌋
    SD_i ← α × BI_i
    répéter
        nœud actif durant SD_i périodes de backoffs
        new ← nombre de nouveaux voisins découverts
        neighbors ← neighbors + new
        nœud inactif durant BI_i − SD_i périodes de backoffs
    jusqu'à ce que Δ périodes de backoffs soient passées
jusqu'à ce que neighbors ≠ 0
tant que batterie ok faire
    nœud actif durant SD_i périodes de backoffs
    nœud inactif durant BI_i − SD_i périodes de backoffs
fintantque
```

Algorithme 1 – Algorithme du mécanisme distribué utilisant des BI et SD indépendants.

2. Cette valeur n'a pas été évaluée mathématiquement

3. Les multiples de 4 sont utilisés pour limiter la taille du plus grand commun multiple des BI dans un voisinage.

Si c'est le cas, le nœud conserve cette valeur BI_i. Après cette phase d'initialisation, chaque nœud i maintient son propre intervalle de balise BI_i et reste actif pendant SD_i périodes de *backoffs* et inactif pendant $BI_i - SD_i$ périodes de *backoffs*.

Lorsque le premier nœud rejoint le réseau, il change plusieurs fois son BI car il n'y a aucun voisin présent. Par contre, lorsqu'un second nœud rejoint le même réseau, ces deux nœuds vont finir par se rencontrer et alors ils conservent leurs BI respectifs.

Notre mécanisme est distribué car il ne nécessite pas une connaissance sur les autres nœuds, et ne requiert pas une grande surcharge en terme de messages de contrôle. Cependant, le temps à partir duquel tous les nœuds ont un BI stables peut être long, en fonction de la valeur de la période d'attente Δ fixée et du nombre moyen de nœuds dans le voisinage de chaque nœud.

3.1.1.3 Approche asynchrone apériodique et cyclique : A_{aa}

Nous avons proposé dans [AGM14b], un autre mécanisme de planification de l'activité des nœuds que nous notons A_{aa} (pour approche asynchrone apériodique et cyclique).

Dans l'approche A_{aa}, le temps est divisé en cycles de durée c. Chaque nœud connaît c et la durée de son activité dans le cycle, notée a ($a = \alpha.c$ où α est le taux d'activité). Au cours de chaque cycle, les nœuds décident indépendamment et de façon aléatoire (dans l'intervalle $[0; c - a[$) l'instant de démarrage de leur activité.

La figure 3.6 montre le mécanisme A_{aa} pour trois nœuds n_1, n_2 et n_3, qui sont supposés être à portée de communication, et avec un taux d'activité α=25 %. On peut

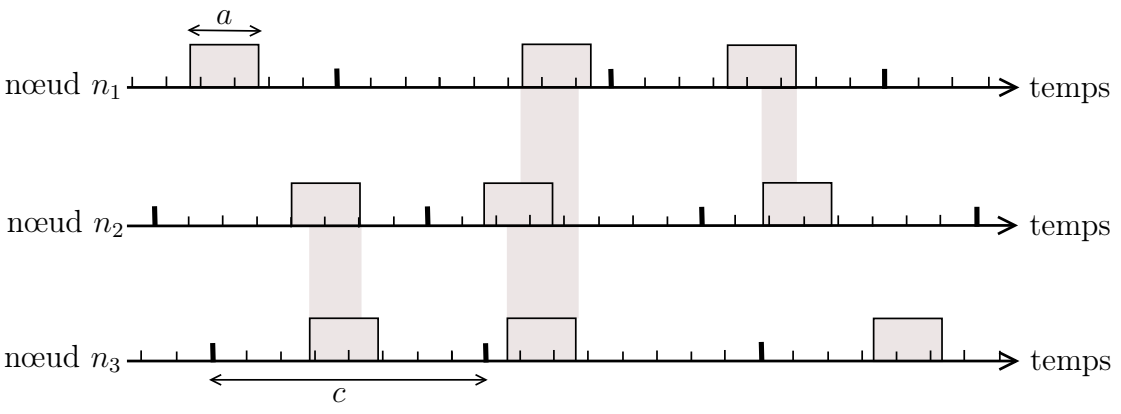


Figure 3.6 – Exemple d'activité de trois nœuds n_1, n_2 et n_3, avec un taux d'activité de 25 %. Les nœuds démarrent indépendamment et aléatoirement leur activité dans chaque cycle c.

voir sur cet exemple que toute paire de nœuds peut partager une période d'activité commune (même si cela ne se produit pas dans tous les cycles), et qu'il y a même

des possibilités que les trois nœuds partagent une activité commune.

Dans l'approche A_{aa}, le délai moyen (noté $d_t(A_{aa})$) avant que deux nœuds puissent partager une période d'activité commune d'au moins t unités de temps appelée rencontre fructueuse est une métrique importante. Nous supposons que le temps est discret et que t comporte le temps nécessaire pour détecter une rencontre, échanger au moins une trame de donnée et recevoir un ACK avec l'algorithme CSMA/CA non slotté. Une étude menée par Niek et al. [NBP+06] montre que ce temps varie entre 2 ms et 6 ms, dans le cas d'un canal parfait. Nous définissons la valeur de t à 15,36 ms (ce qui est aussi la durée minimale d'une supertrame dans la norme IEEE 802.15.4). Nous évaluons mathématiquement le délai $d_t(A_{aa})$. La rencontre fructueuse n'est possible que dans le cas où $t < a$ (la durée totale de l'activité commune t entre deux paires de nœuds est toujours inférieure à la durée totale de l'activité a). Il est important de noter que la rencontre fructueuse entre n_1 et n_2 ne se produit pas nécessairement au cours de la première activité de n_1.

La figure 3.7 montre le délai $d_t(A_{aa})$ d'un nœud n_1 avant de partager une activité fructueuse avec un nœud voisin n_2. Ce délai est évalué comme la somme de la durée entre l'activité de n_1 et la fin du premier cycle de n_1 (notée d_f), plus la durée totale des cycles sans rencontres (notée d_i^k), plus la durée passée par n_1 dans le cycle de rencontre avant le début de l'activité commune avec n_2 (notée δ).

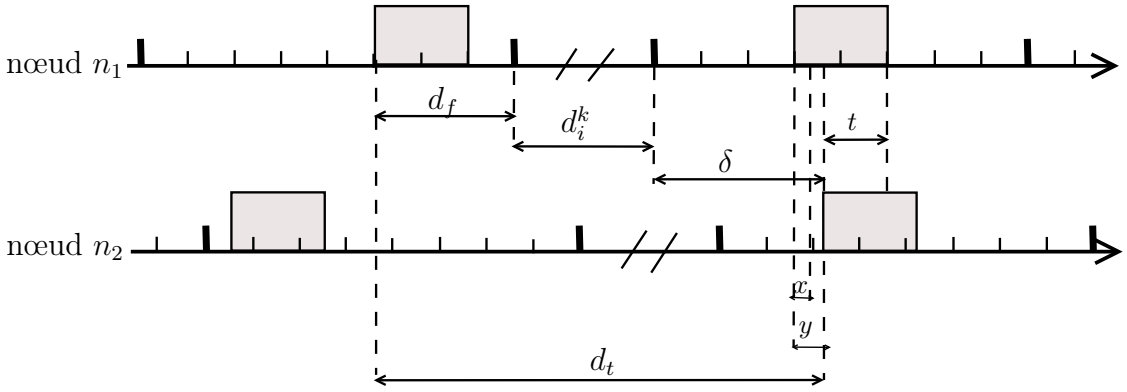


Figure 3.7 – Délai d_t d'un nœud n_1 avant de partager une activité fructueuse avec un nœud voisin n_2 pour un taux d'activité de 25 %.

Pour calculer le délai moyen $d_t(A_{aa})$, nous avons besoin de déterminer la probabilité de rencontre fructueuse entre deux nœuds dans un cycle donné. Dans un cycle de rencontre soit le nœud n_1 démarre son activité peu de temps avant celle de n_2 (voir la partie gauche de la figure 3.8). Cette situation se produit avec une probabilité $(a-t)/c$ et garantit des rendez-vous pendant au moins t unités de temps. Soit le nœud n_2 démarre son activité peu de temps avant celle de n_1 (voir la partie droite de la figure 3.8). Cette situation se produit avec une probabilité $(a-t-1)/c$ et garantit des rendez-vous

pendant au moins t unités de temps. On obtient donc par sommation que la probabi-

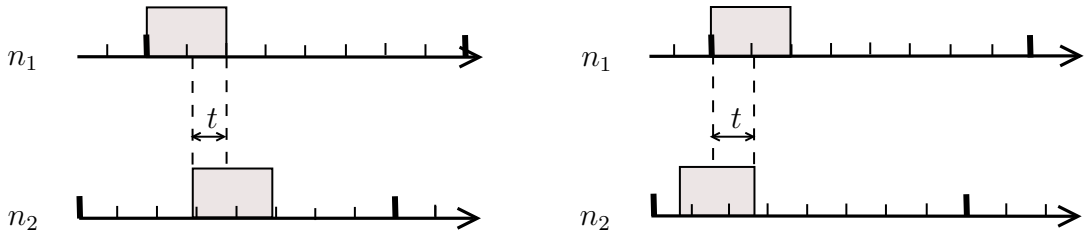


Figure 3.8 – Les deux possibilités de rencontre fructueuse entre deux nœuds voisins n_1 et n_2.

lité $p_t(A_{aa})$ pour que deux nœuds partagent une activité commune d'au moins t unités de temps au cours d'un cycle donné est : $p_t(A_{aa}) = (2(a-t)-1)/c$. Il faut dire que cette modélisation ne prend pas en compte la contention pour l'accès au canal (même si une partie de cette contention peut être intégrée dans le choix de la valeur de t). La probabilité $p_t^*(A_{aa})$ pour que deux nœuds se rencontrent à terme (c'est-à-dire dans n'importe quel cycle) et partagent une activité commune pendant au moins t unités de temps peut être calculée comme suit : $p_t^*(A_{aa}) = \sum_{i=1}^{+\infty} p_t(A_{aa})(1-p_t(A_{aa}))^{i-1}$, où i est le numéro du cycle dans lequel la rencontre fructueuse a lieu.

Le délai moyen $d_t(A_{aa})$ peut être étudié dans deux cas : le cas où la rencontre a lieu dans le premier cycle et le cas où la rencontre a lieu dans un cycle ultérieur.

Dans le cas où la rencontre entre n_1 et n_2 se produit dans le premier cycle (qui arrive avec une probabilité $p_t(A_{aa})$ et dans ce cas d_i^k et δ sont égales à 0) on observe les deux cas suivants : soit n_1 commence son activité avant n_2 (au plus $a-t$ unités de temps, voir partie gauche de la figure), le délai d_1 dans ce cas étant alors $d_1 = \sum_{x=0}^{a-1}(c-x)/a = (2.c-a+1)/2$, soit n_2 commence son activité avant n_1 (au plus $a-t$ unités de temps, voir la partie droite de la figure 3.8) dans ce cas le délai est nul.

Si n_1 et n_2 se rencontrent pendant le $k^{ème}$ cycle, avec $k \geq 2$ (qui arrive avec une probabilité $1-p_t(A_{aa})$), les composantes du délai $d_t(A_{aa})$ s'expriment comme suit : La durée $d_f = \sum_{x=0}^{a-1}(c-x)/a = (2.c-a+1)/2$, la durée d_i^k peut être simplement exprimée comme : $d_i^k = (k-2).c$, et la durée δ peut être obtenue en additionnant le temps avant le début de l'activité de n_1 dans le dernier cycle, et le temps calculé précédemment d_1, on obtient donc : $\delta = d_1 + \sum_{z=0}^{c-a-1} z/(c-a) = (c-a-1)/2 + d_1$. Par conséquent, le délai d_2 dans ce cas est :

$$d_2 = d_f + \sum_{k=2}^{+\infty}\left((1-p_t(A_{aa}))^{k-1}.p_t(A_{aa})d_i^k\right) + \delta.$$

Le délai moyen $d_t(A_{aa})$ est finalement exprimé comme suit :

$$d_t(A_{aa}) = p_t(A_{aa}) \cdot d_1 + (1 - p_t(A_{aa})) \cdot d_2.$$

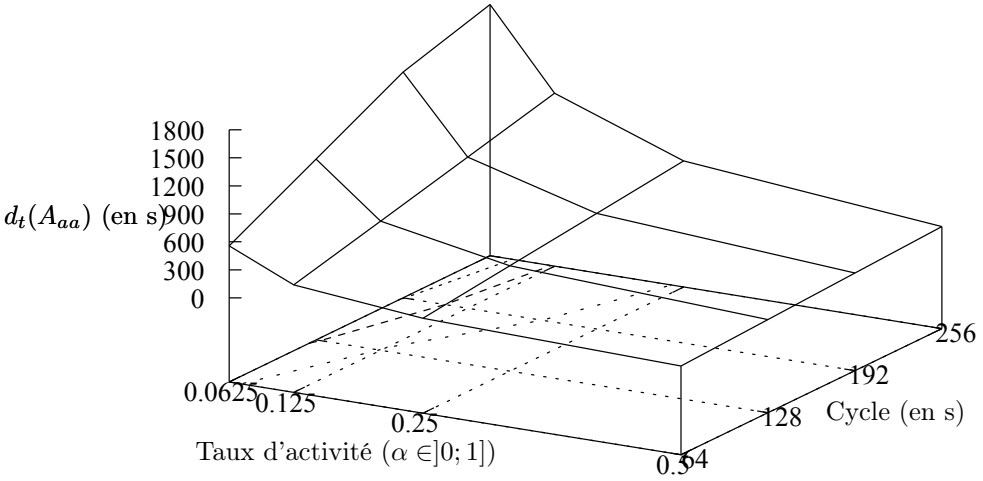


Figure 3.9 – Délai moyen $d_t(A_{aa})$ (en secondes) avant que deux nœuds établissent un contact d'une durée minimale $t = 15.36$ ms, déterminé analytiquement.

La figure 3.9 montre l'évaluation numérique du délai $d_t(A_{aa})$ en fonction de la durée du cycle c et du taux d'activité α. On peut voir que le délai augmente quand la durée du cycle c augmente ou quand le taux d'activité diminue. Quand le taux d'activité est de 6.25 % et la durée du cycle est de 256 s, il faut en moyenne 1750 s pour un $t = 15.36ms$ (soit moins de 7 cycles) pour que deux nœuds partagent une rencontre fructueuse.

L'approche A_{aa} génère un long délai avant qu'une paire de nœuds donnée partage une période d'activité commune. Notons que ce délai concerne uniquement le cas où les nœuds disposent d'un seul voisin potentiel pouvant servir de nœud intermédiaire pour transmettre les données. Ce délai peut être réduit si le nœud a plus d'un voisin plus proche de la destination que lui. Par exemple sur la figure 3.10, une source s est à portée de k nœuds voisins plus proche de la destination que lui. Chacun de ces k nœuds est à porté de transmission avec la destination d (supposé toujours actif). Ainsi, la source s a k possibilités de nœuds intermédiaires pour envoyer ses données à la destination d. Ce qui fait que, le nombre moyen de cycles pour que s communique avec un voisin plus proche de la destination diminue : ce nombre est en moyenne de 100 cycles pour un voisin et un taux d'activité de 1 %, de 50 cycles pour deux voisins, et de 33 cycles pour trois voisins. Cela réduit le délai d'attente avant que les nœuds partagent des rencontres fructueuses avec leurs voisins.

L'approche A_{aa} présente de nombreux avantages.

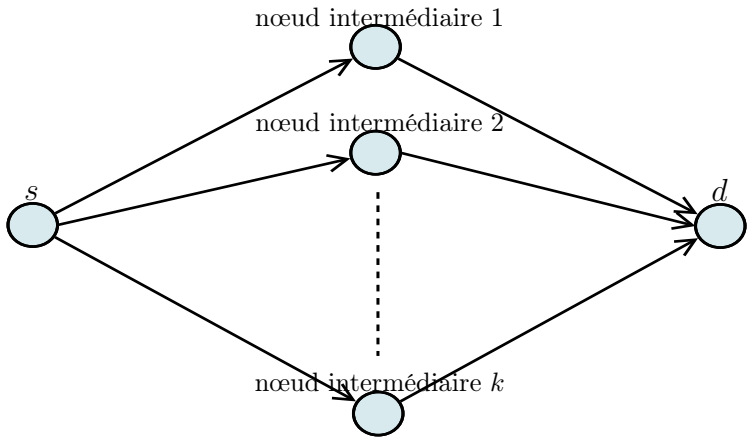


Figure 3.10 – Topologie réseau (appelée topologie en diamant) avec un nœud source s qui dispose de k possibilités de nœuds intermédiaires pour atteindre d.

- Elle ne nécessite pas de synchronisation, donc permet de développer un protocole MAC complètement asynchrone, et notamment robuste aux dérives d'horloges, qui peuvent être très importantes lorsque les nœuds sont inactifs pendant de grandes durées.
- Elle évite que l'activité de paire de nœuds à portée soit disjointe.

3.1.2 Protocole MAC à rencontres aveugles

Nous avons proposé dans [AGM14a], un protocole MAC asynchrone à rencontres aveugles basé sur le mécanisme de planification A_{aa} décrit dans la partie 3.1.1.3 que nous notons A_{aa}MAC dans la suite.

3.1.2.1 Description du protocole MAC à rencontres aveugles : A_{aa}MAC

Une rencontre aveugle se réfère à la capacité d'un nœud à être actif en même temps qu'un autre nœud sans hypothèse préalable de synchronisation ou d'échange d'informations sur les activités. Dans [MMKR13], les auteurs montrent que peu de protocoles MAC sont effectivement basés sur des rencontres aveugles. Nous proposons un protocole MAC basé sur des rencontres aveugles et où la rencontre entre les nœuds est détectée par la réception d'une trame balise.

Notre protocole MAC utilise l'algorithme CSMA/CA non slotté de la norme IEEE 802.15.4 sans suivi de balises, mais avec un mécanisme d'activation des nœuds basé sur l'approche A_{aa}. Chaque nœud connaît la durée de son cycle c et la durée a de son activité dans le cycle. Au cours de chaque cycle, les nœuds choisissent indépendamment et aléatoirement dans l'intervalle $[0; c - a[$ l'instant de démarrage de leur activité.

Quand un nœud se réveille, il diffuse une trame balise pour annoncer son réveil à son voisinage. Pour réduire la probabilité de collision, les trames balises sont envoyées en utilisant le mécanisme CSMA/CA non slotté, comme indiqué sur la figure 3.11. Il faut noter que la tentative d'envoi de la balise a un crédit illimité (au moins jusqu'à la fin de la période d'activité du nœud). En cas de tentative infructueuse d'accès au canal, elle est répétée jusqu'à ce que le nœud accède au canal et envoie sa trame balise ou jusqu'à la fin de son activité. En raison des périodes de *backoffs* aléatoires du mécanisme CSMA/CA non slotté, la trame balise peut ne pas être envoyée immédiatement après le réveil du nœud, ce qui réduit la durée de l'activité qui peut servir à l'échange des trames de données.

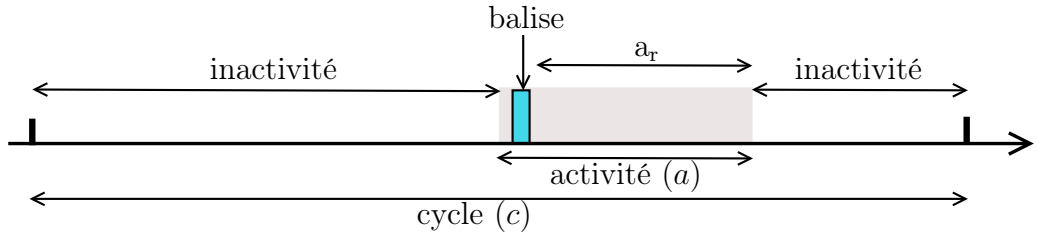


Figure 3.11 – Zoom sur l'activité d'un nœud dans un cycle.

La figure 3.12), représente la structure d'une trame balise. Le paramètre *type* représente le type de trame (qui peut être une balise, une notification, une donnée ou un ACK), d_n est un booléen qui indique si le nœud est apte à recevoir des données et r_n indique si le nœud a des données dans sa file d'attente. *Id-Source* est l'identifiant du nœud et a_r est la durée restante de la période d'activité dans le cycle en cours.

Type	Id-Source	d_n	r_n	a_r

Figure 3.12 – Structure d'une trame balise.

Toutes les trames de données transmises requièrent un accusé de réception. Après quatre envois d'une même trame de données, si aucun accusé de réception n'est reçu, la trame de données est supprimée et une notification est envoyée à la couche réseau.

Nous faisons l'hypothèse que les communications sont de type *convergecast* (c'est-à-dire des nœuds vers une seule destination finale) et nous ignorons le problème du terminal caché. Aussi, dans nos protocoles un nœud peut être à la fois émetteur et récepteur. Pour un nœud n tout voisin v plus éloigné de la destination peut se servir de n comme nœud intermédiaire, et tout voisin v plus proche que n de la destination peut être un nœud intermédiaire pour n (notons que cette décision est prise au niveau de la couche réseau).

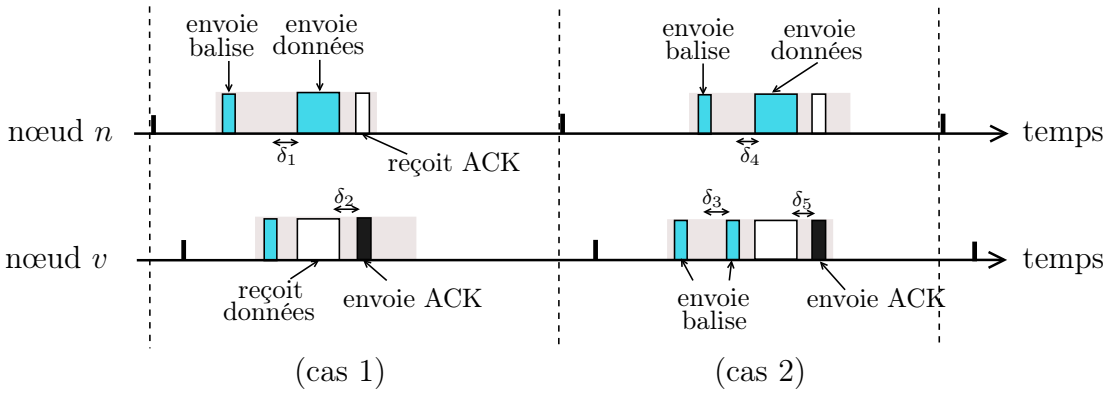


Figure 3.13 – Exemple des deux cas d'envoi et de réception de données quand les nœuds partagent une période d'activité commune avec une possibilité de nœud intermédiaire.

Lorsqu'un nœud n, avec des données dans sa file d'attente passe en période d'activité on peut observer les situations suivantes :

1. Soit le nœud ne rencontre aucun voisin (plus proche de la destination que lui) actif avant sa période d'activité et même durant sa période d'activité. On dira dans ce cas que le nœud n n'a eu aucune rencontre fructueuse, il devra attendre le prochain cycle pour voir s'il partage une activité fructueuse avec l'un de ses voisins plus proche de la destination que lui.
2. Soit le nœud réalise une seule rencontre fructueuse avec un seul voisin plus proche de la destination que lui. La figure 3.13 décrit ce cas dans lequel un nœud n a des données à envoyer à un nœud voisin v. La partie gauche (cas 1) de la figure montre le cas d'une rencontre fructueuse lorsque le nœud émetteur n commence son activité avant le nœud v. Après la réception de la balise du nœud v, le nœud n envoie sa trame de données au nœud v et attend un accusé de réception. Lorsque le nœud v reçoit la trame de données, il répond par une trame ACK (toutes les trames sont envoyées en utilisant le mécanisme CSMA/CA non slotté). La partie droite (cas 2) de la figure 3.13 montre le cas où, le nœud récepteur v commence sa période d'activité avant le nœud émetteur n. Ici, à la réception de la balise du nœud n, le nœud v envoie une deuxième balise pour signifier qu'il est aussi en activité et qu'il peut recevoir des données. Ensuite, le nœud n peut envoyer ses données. Enfin, le nœud v termine la transmission par un accusé de réception.
   Il faut noter que les durées δ_1 à δ_5 représentent le temps nécessaire pour qu'un nœud effectue le CSMA/CA et transmettre une trame. Ces durées vont dépendre du nombre de nœuds en compétition pour l'accès au médium dans le voisinage des nœuds.

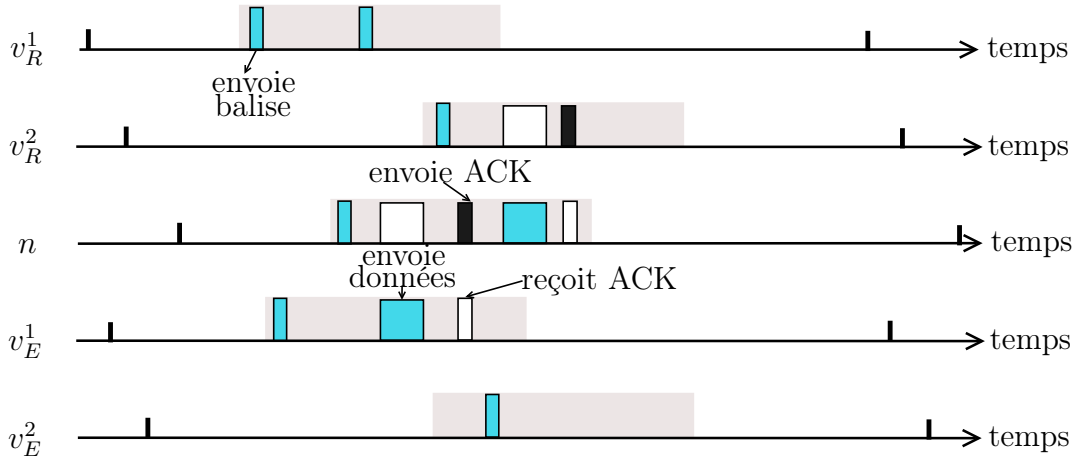


Figure 3.14 – Exemple d'envoi et de réception de données quand les nœuds partagent une période d'activité commune avec plus d'une possibilité de nœuds intermédiaires.

3. Soit le nœud rencontre plusieurs voisins (certains plus proches de la destination que lui et d'autres plus éloignés de la destination que lui). La figure 3.14 illustre un cas où un nœud n est en activité et a des données dans sa file d'attente. Dans son voisinage un ensemble de nœuds plus proche de la destination que n (ici v_R^1 et v_R^2) et d'autres plus éloignés de la destination que n (ici v_E^1 et v_E^2) vont partager une activité commune avec n.
   On suppose que tous les nœuds sont à portée de transmission de n, v_R^1 et v_R^2 sont aussi à portée l'un de l'autre tout comme v_E^1 et v_E^2, mais les nœuds v_R^1 et v_R^2 ne sont pas à portée de transmission de v_E^1 et v_E^2.
   Dans l'exemple représenté, v_R^1 et v_E^1 ont débuté leur période d'activité avant n et v_R^2 et v_E^2 débutent leur période d'activité après n. Après que n ait diffusé sa balise d'annonce d'activité, v_R^1 est le premier à annoncer à n qu'il est aussi en activité. Quand n entend la balise de v_R^1, il ne réussit pas à envoyer sa trame à v_R^1 car v_E^1 tente de lui envoyer sa trame et accède au canal en premier. n reçoit donc la trame de donnée de v_E^1 et annule sa transmission pour envoyer une trame ACK à v_R^1, mais le démarrage de l'activité de v_R^2 va retarder sa transmission. Dans le même temps les communications en cours empêchent v_E^2 de diffuser sa balise d'annonce d'activité. Dans le cas où le nœud n entend finalement la balise de v_E^2 avec une annonce de file non vide, n ne peut pas annoncer à v_E^2 qu'il est aussi en activité vu qu'il a une communication en cours. La deuxième balise d'annonce d'activité est envoyée uniquement quand le nœud n'a aucune communication en cours. Notons que lorsqu'un nœud a la possibilité de communiquer avec $k > 1$ voisins plus proches de la destination que lui, il effectue un tirage aléatoire sur k et le voisin à la position de l'indice tirer (dans

sa liste de voisins proche en activité) sera le récepteur de sa trame.
Notons que nous n'ignorons pas les possibilités de collisions. Les collisions de balises ne sont jamais constatés par l'émetteur et donc empêchent que les voisins déjà en activité aient la connaissance de l'activité annoncée. Par contre, les nœuds retransmettent leur trames de données s'ils ne reçoivent pas de trame ACK après une durée prédéfinie (au maximum 4 fois).

Le fonctionnement détaillé de notre protocole MAC est décrit dans la suite.

3.1.2.2 Algorithme de fonctionnement du protocole A_{aa}MAC

Dans chaque cycle faire
si la file d'attente du nœud n est vide **alors**
  $r_n \leftarrow faux$
sinon
  $r_n \leftarrow vrai$
finsi
si il y a de la place dans la file d'attente de trames de données de n **alors**
  $d_n \leftarrow vrai$
sinon
  $d_n \leftarrow faux$
finsi
n attend durant un temps aléatoire dans [0;c-a]
n est actif durant a périodes de *backoffs*
n envoie une notification d'activité à la couche réseau en début d'activité
n diffuse une trame balise comportant d_n, r_n et a_r
n est inactif jusqu'à la fin du cycle

Algorithme 2 – Algorithme de fonctionnement de notre protocole MAC à rencontres aveugles.

L'algorithme 2 donne un fonctionnement détaillé de notre protocole MAC. Lorsqu'un nœud n se réveille, il envoie une notification de son activité à la couche réseau, puis diffuse une trame balise pour informer son voisinage de son réveil et de son éventuelle disponibilité à recevoir des données. Chaque balise (voir la figure 3.12) comporte la durée restante de l'activité de n (notée a_r) et deux variables booléennes d_n et r_n représentant respectivement la disponibilité et la requête pour une transmission. Dans notre protocole, lorsque la file d'attente de données d'un nœud est pleine, ce nœud ignore toutes les trames qu'il reçoit. La variable d_n est utilisée pour indiquer si la file d'attente d'un nœud n est presque pleine, ce qui correspond à d_n valant faux ou non. La variable r_n est utilisée pour indiquer si le nœud n fait une requête en vue

de transmettre des données ou non (si sa file d'attente en transmission est vide, r_n est fausse, sinon r_n est vraie). Lorsqu'un nœud est en fin d'activité ou à la fin de celle d'un voisin ou lorsqu'un nœud reçoit une balise d'un voisin v avec la variable d_v égale à vraie, il envoie une notification à la couche réseau. Les trames reçues par un nœud pendant sa période d'activité sont traitées par la fonction Réception(*trame*) décrite dans l'algorithme 3. La trame reçue peut être une notification de la couche supérieure pour envoyer une trame de données, ou une balise, ou un accusé de réception ou une trame de données.

```
Entrée : Réception(trame)
si type=notification alors
    envoyer la trame de donnée en tête de file
sinon
    si type=balise alors
        envoyer une notification à la couche réseau
    sinon
        si type=ACK alors
            supprimer la trame en tête de file
            si file non vide et a_r suffisante alors
                envoyer la trame en tête de file
            finsi
        sinon
            si direction=vers couche basse alors
                mettre la trame en file d'attente
            sinon
                remonter la trame à la couche réseau
            finsi
        finsi
    finsi
finsi
```

Algorithme 3 – Fonction de réception d'une trame.

- Si la trame reçue est une notification d'émission, le nœud envoie la trame de données en tête de file à l'adresse contenue dans la notification.
- Si la trame est une balise, le nœud envoie une notification à la couche réseau pour indiquer l'activité du nœud source de la balise (*Id-Source*) et le minimum entre la durée restante de son activité et celle du voisin.
- Si c'est une trame ACK, le nœud supprime la trame en tête de file. Si la file est non vide, le nœud vérifie si son récepteur est toujours activité (avec une durée d'activité restante suffisante pour recevoir des données et transmettre un ACK), si c'est le cas, il envoie la trame de données en tête de file. Sinon,

il envoie une notification de fin d'activité du voisin en question à la couche supérieure pour trouver un autre nœud intermédiaire si nécessaire.

- Si c'est une trame de données, le nœud vérifie si elle est destinée aux couches basses. Si oui, il la met en file d'attente. Sinon, il la remonte à la couche réseau.

3.1.2.3 Avantages et inconvénients du protocole A_{aa}MAC

Les avantages de notre protocole MAC peuvent être résumés comme suit.

- Il ne nécessite pas de synchronisation entre les nœuds.
- Les nœuds sont en mesure de communiquer avec tous leurs voisins, ce qui améliore l'efficacité énergétique du protocole. Par exemple, si les nœuds sont actifs seulement 1 % du temps, le nombre moyen de cycles pour que deux nœuds partagent une communication est de 100. Toutefois, plus un nœud a de voisins, plus le nombre moyen de cycles pour qu'ils communiquent diminue.
- La contention pour l'accès au médium est généralement faible (comme le nombre moyen de nœuds actifs en même temps est faible).
- Les nœuds maintiennent le même taux d'activité, ce qui permet de maîtriser la consommation énergétique globale du réseau et donc de prédire la durée de vie du réseau.

Cependant, notre protocole MAC peut conduire à un long délai avant que les nœuds voisins se rencontrent pour communiquer (voir la figure 3.9), et la durée d'activité commune qui peut servir à l'échange des trames par rapport au protocole MAC synchrone est réduite.

3.1.3 Optimisations du protocole MAC à rencontres aveugles

Le délai dans le protocole MAC que nous avons proposé peut être grand, car une source doit attendre pour avoir une période d'activité commune avec une destination avant d'envoyer ses trames. Ce délai dépend principalement des paramètres c, a et aussi du nombre de voisins (voir [AGM14a]). Dans la partie 3.1.3.1 nous montrons comment réduire le délai d'attente avant rencontre en fragmentant c pour déterminer une valeur optimale qui permet de garantir des périodes de rencontres suffisamment longues pour les communications et aussi qui assure un faible délai d'attente avant rencontre. Nous montrons dans la partie 3.1.3.2 qu'il est aussi possible de réduire le délai d'attente (une fois que la valeur optimale de c est trouvée), si les nœuds adaptent leur activité en fonction de celle des autres nœuds.

3.1.3.1 Protocole MAC à rencontres aveugles avec périodes d'activités fragmentées

La fragmentation peut être exprimée de la façon suivante : on peut réduire c (et a) tout en gardant un taux d'activité constant. Plus c diminue, plus la durée avant rencontre diminue (mais plus la durée des rencontres diminue aussi).

Principe de base de la fragmentation d'activité

Il est possible de réduire le délai avant une activité commune dans le protocole MAC en fragmentant l'activité des nœuds en plusieurs parties : au lieu d'avoir une seule activité qui dure a unités de temps par cycle, un nœud peut être actif f fois par cycle, pour une durée de a/f unités de temps pour chaque activité. De cette façon, le taux d'activité est toujours a/c, mais les nœuds partagent des activités communes plus souvent. On dit que c a été fragmenté f fois [4].

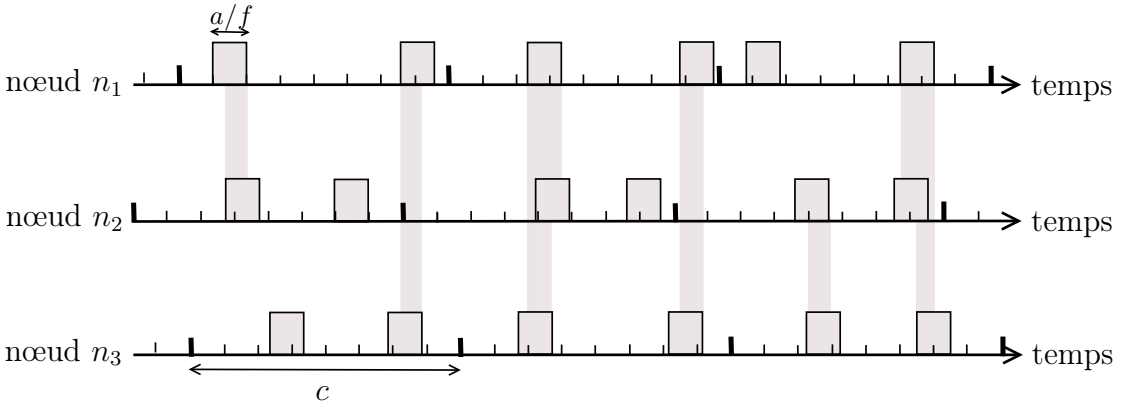


Figure 3.15 – Exemple d'activité de trois nœuds n_1, n_2 et n_3, quand l'activité est fragmentée en $f = 2$, avec un taux d'activité de 25 %. Chaque activité est aléatoirement localisée deux fois chaque c unités de temps, au lieu d'une fois chaque c unités de temps.

La figure 3.15 montre notre mécanisme de fragmentation quand le taux d'activité est de 25 %, avec $f = 2$, $c = 8$ et $a = 2$. On peut remarquer que chaque nœud dispose désormais de deux activités courtes par cycle, qui se traduisent par plus d'activités communes, ce qui réduit le temps d'attente d'un nœud avant de partager une période d'activité commune avec un voisin.

L'algorithme 4 donne un pseudo-code de la planification de l'activité d'un nœud

4. Rappelons que la consommation énergétique pour activer/désactiver le module radio est négligeable par rapport à la consommation énergétique lorsque le nœud a son module radio actif. Par exemple, sur le composant CC2420 (utilisé sur les TelosB et MicaZ) pour une puissance de transmission de 0 dbm, le composant consomme 18.8 mA en réception, 17.4 mA en émission, 20 μA pour la désactivation et une durée de 100 μs pour activer du module radio [Ins06].

avec le mécanisme de fragmentation.

$i \leftarrow 1$
tant que $i \leq f$ **faire**
  $t_i^{debut} \leftarrow \text{random}((c-a)/f)$
  le nœud attend t_i^{debut}
  le nœud active son module radio
  le nœud attend a/f unités de temps (avec le module radio activé)
  le nœud désactive son module radio
  le nœud attend $\frac{c-a}{f} - t_i^{debut}$ unités de temps
  $i \leftarrow i + 1$
fintantque

Algorithme 4 – Planification de l'activité d'un nœud dans un cycle, avec une activité fragmentée f fois.

La durée du cycle c et de l'activité a, et le nombre de la fragmentation f sont égaux pour tous les nœuds. Le nœud n se réveille à la date t_i^{debut} dans chaque fragment $i \leq f$ de cycle (c/f), $i \in [1; f]$, pendant a/f unités de temps et ensuite éteint son composant radio pendant $(c-a)/f - t_i^{debut}$ unités de temps.
Lorsqu'un nœud est actif, il fonctionne exactement suivant l'algorithme de fonctionnement du protocole MAC proposé dans la partie 3.1.2.

Le mécanisme de fragmentation réduit le délai avant que deux paires de nœuds quelconques partagent une activité commune par rapport au protocole MAC de base que nous avons proposé. Cependant, ce mécanisme réduit la durée de l'activité commune. Nous ne pouvons donc pas fragmenter indéfiniment l'activité dans un cycle c. Trouver la valeur optimale de fragmentation f de telle sorte que nous ayons non seulement un faible délai, mais aussi des durées d'activités communes assez longues pour communiquer, est un problème qui sera discuté dans la partie 4.2.2.

Extension possible avec le mécanisme de fragmentation

Il est possible de se servir du mécanisme de fragmentation pour proposer un protocole MAC à fragmentation auto-adaptative (avec un taux d'activité fixe) qui ne souffre pas du problème de rencontres courtes. Pour cela, il est possible d'utiliser la fragmentation pour détecter une rencontre et si besoin étendre cette activité, tout en maintenant fixe la durée de l'activité a dans chaque cycle, comme indiqué sur la figure 3.16. Il y a à présent au plus f activités par cycle, chacune durant au moins a/f unités de temps. Lors d'une rencontre fructueuse, l'activité est étendue tant que les communications sont effectives, dans la limite du temps d'activité restant dans

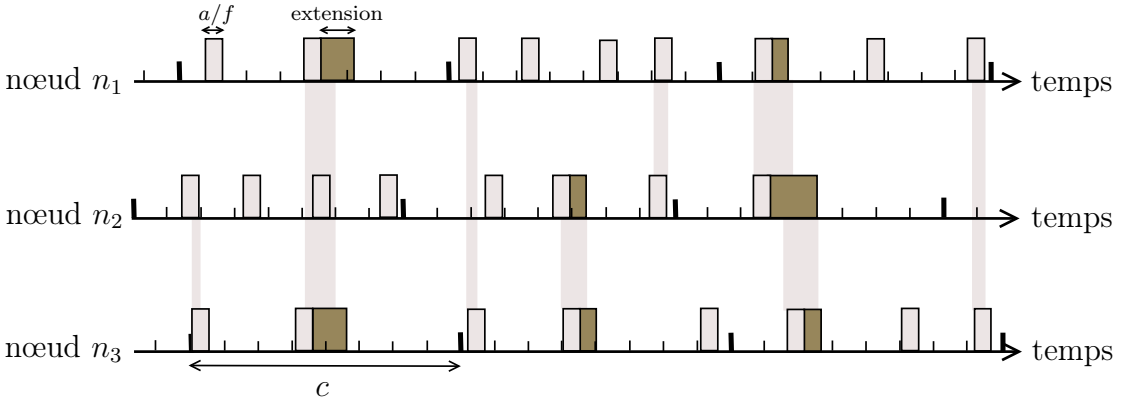


Figure 3.16 – Exemple d'activité de trois nœuds n_1, n_2 et n_3, quand l'activité est fragmentée en $f = 4$, avec possibilité d'étendre l'activité si nécessaire, pour un taux d'activité de 25 %.

le cycle. La durée totale de l'activité par cycle c n'excède donc jamais a unités de temps.

Cette extension sera exploitée dans nos travaux futurs.

3.1.3.2 Protocole MAC adaptatif

Nous avons proposé SLACK-MAC (pour *Adaptive MAC Protocol for Low Duty-Cycle Wireless Sensor Networks*) dans [AGLM15], pour que les nœuds adaptent leurs activités en fonction des activités des autres nœuds. Ce protocole s'inspire des travaux proposés dans [ADJL13], où les auteurs ont proposé le protocole SR3, qui utilise un mécanisme de réputation.

Dans SLACK-MAC, les nœuds enregistrent un historique des communications fructueuses avec leurs voisins, et se servent de cet historique pour déterminer la date de la prochaine activation de leur module radio. Cet historique augmente la probabilité de sélectionner une date de réveil qui a permis des communications fructueuses.

Description de SLACK-MAC

L'idée principale de SLACK-MAC est de maintenir un historique des temps de démarrage d'activité dans un cycle correspondant à des communications réussies avec les voisins. Dans SLACK-MAC, les nœuds ne choisissent pas toujours leur date d'activation uniformément de façon aléatoire dans chaque cycle, mais ils se servent de la connaissance des communications réussies pour calculer leur prochaine date d'activation.

La figure 3.17 décrit un exemple d'activité de trois nœuds voisins n_1, n_2 et n_3 avec SLACK-MAC. Initialement, tous les nœuds choisissent leurs temps d'activation uni-

formément de façon aléatoire dans chaque cycle. Une fois qu'un nœud choisit une date d'activation qui conduit à une communication réussie (réception ou émission d'une trame), il mémorise cette date et la probabilité de choisir cette même date augmente, comme on peut le voir vers la droite de la figure (la couleur foncée représente les rencontres basées sur l'historique de temps correspondant à une communication réussie avec un voisin).

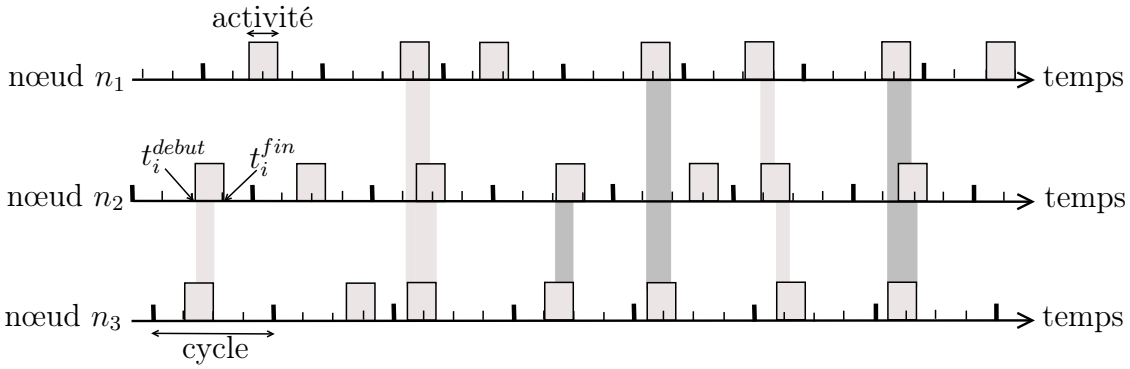


Figure 3.17 – Exemple d'activité de trois nœuds voisins n_1, n_2 et n_3, avec SLACK-MAC, pour un taux d'activité de 25 %.

Mécanisme de construction des liste dans SLACK-MAC

SLACK-MAC requiert que chaque nœud maintienne deux listes, E (pour liste d'émission) et R (pour liste de réception) qui contiennent des dates de démarrage d'activité dans les cycles.

Notons t_i^{debut} le début de l'activité dans le cycle courant i et t_i^{fin} la fin de la période d'activité dans le cycle i, comme représenté sur la figure 3.17. La date t_i^{debut} est la valeur aléatoire tirée juste en début de période d'inactivité dans le cycle $i-1$ pour déterminer à partir de quelle date la période d'activité va démarrer dans le prochain cycle i. Au cours d'une activité, un nœud peut ajouter t_i^{debut} une fois par cycle soit dans la listes E ou soit dans la liste R (ou dans les deux listes) si ce nœud envoie ou (et) reçoit une ou plusieurs trames de données dans le cycle. Chaque nœud utilise ces deux listes pour déterminer sa prochaine date de réveil. La figure 3.18 montre un exemple de l'évolution des listes E et R dans trois cas différents. Le cas 1 montre l'état des listes quand elles sont vides, c'est-à-dire au démarrage lorsque le nœud n'a réalisé aucune rencontre fructueuse (les deux listes sont vides). Le cas 2 montre l'état des listes lorsque pendant le cycle i le nœud démarre son activité à la date t_i^{debut} et reçoit avec succès au moins une trame de données, t_i^{debut} est ajouté à la liste R. Le cas 3 montre l'état des listes lorsqu'ensuite, dans le cycle j le nœud démarre son activité à la date t_j^{debut} envoie et reçoit également avec succès au moins une trame de données,

t_j^{debut} est ajouté dans les deux listes E et R. Lorsqu'une liste est pleine et qu'il y a une nouvelle entrée, l'entrée la plus ancienne de la liste est supprimée, puis les autres sont décalées afin de libérer la dernière place de la liste pour la nouvelle entrée.

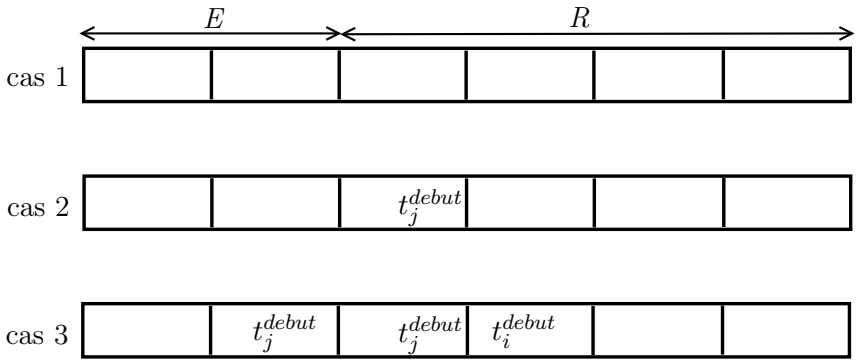


Figure 3.18 – Exemple de l'état des listes E et R dans trois cas différents.

Algorithme du fonctionnement de SLACK-MAC

L'algorithme 5 donne le pseudo-code du comportement d'un nœud quand il est en activité. Dans SLACK-MAC chaque nœud a une file d'attente des trames de données de taille fixe pour les données qui doivent être transmises, notée *file*. Si *file* est pleine et un nœud reçoit un nouveau paquet, le nœud ignore ce dernier paquet. Afin d'optimiser notre protocole, suivant l'état de *file*, chaque nœud adapte sa stratégie de sélection de sa prochaine date de réveil, selon les règles suivantes :

- Si *file* est vide (état=1), un nœud n'a pas de paquets à envoyer, il est donc inutile de sélectionner les dates de démarrage qui sont dans la liste E. La prochaine date de réveil est choisie de manière uniforme dans R avec une probabilité de $1/2$, et uniformément de façon aléatoire dans D autrement.
- Si *file* est pleine (état=2), un nœud ne peut plus accepter les trames entrantes, il est donc inutile de sélectionner des temps qui sont dans la liste R. La prochaine date de réveil est choisie de manière uniforme dans E avec une probabilité de $1/2$, et uniformément de façon aléatoire dans D autrement.
- Dans tous les autres cas (état=3), un nœud sélectionne sa prochaine date de réveil uniformément dans R avec une probabilité de $1/3$, de manière uniforme dans E avec une probabilité de $1/3$, et uniformément de façon aléatoire dans D autrement.

  Plus formellement, la probabilité qu'un nœud sélectionne la date de son prochain réveil $t \in D$ (où D désigne toutes les dates de réveil possibles dans le cycle, c'est à dire une date entre $[0; c - a]$, c étant le cycle et a l'activité) est

donnée par la formule suivante :

$$Pr[X = t] = \frac{\mathbb{1}_{|R|\neq 0}\left(\frac{|R|_t}{|R|}\right) + \mathbb{1}_{|E|\neq 0}\left(\frac{|E|_t}{|E|}\right) + \frac{1}{|D|}}{\mathbb{1}_{|R|\neq 0} + \mathbb{1}_{|E|\neq 0} + 1} \tag{3.1}$$

où $|L|$ représente le nombre d'éléments dans la liste L, $|L|_t$ représente le nombre d'occurrences de temps t_i^{debut} dans L, et $\mathbb{1}_P$ est la fonction indicateur (elle est égale à 1 si le prédicat P est vrai et 0 sinon).

```
tant que nœud n est actif faire
    si n reçoit une trame du nœud s pendant le cycle i alors
        ajouter la trame dans file
        si t_i^debut n'a pas encore été ajouté pour le cycle i alors
            ajouter t_i^debut à R
        finsi
    finsi
    si n envoie une trame à un nœud r pendant le cycle i alors
        enlever la trame de file
        si t_i^debut n'a pas encore été ajouté pour le cycle i alors
            ajouter t_i^debut à E
        finsi
    finsi
fintantque
si file est vide alors
    t ← Date-Prochain-Réveil(état=1)
sinon
    si file est pleine alors
        t ← Date-Prochain-Réveil(état=2)
    sinon
        t ← Date-Prochain-Réveil(état=3);
    finsi
finsi
programmer le prochain réveil à la date t du prochain cycle
```

Algorithme 5 – Activité d'un nœud n dans SLACK-MAC.

La fonction Date-Prochain-Réveil(état) présentée dans l'algorithme 6, détermine la date du prochain réveil suivant le contenu des deux listes et la formule (3.1). Notons que, initialement lorsque les deux listes sont vides, nous avons $Pr[X = t] = \frac{1}{|D|}$, ce qui signifie que la date du prochain réveil est choisie uniformément de façon aléatoire dans D. Si l'une des deux listes est vide, nous sélectionnons un élément de la liste non vide avec une probabilité de $1/2$, et une date uniformément de façon aléatoire dans D sinon. Si aucune des deux

listes n'est vide, nous sélectionnons un élément de la liste R avec une probabilité de $1/3$, un élément de la liste E avec une probabilité de $1/3$, et une date uniformément de façon aléatoire dans D sinon.

```
Entrée : état (de la file)
si état=1 alors
    indice ← random(2);
    si indice = 0 alors
        t ← R[random(taille(R))];
    sinon
        t ← random(c − a);
    finsi
sinon
    si état=2 alors
        choix ← random(2);
        si choix = 0 alors
            t ← E[random(taille(E))];
        sinon
            t ← random(c − a);
        finsi
    sinon
        choix ← random(3);
        selon que
            choix = 0 :
            t ← E[random(taille(E))];
            choix = 1 :
            t ← R[random(taille(R))];
            choix = 2 :
            t ← random(c − a);
        finselonque
    finsi
finsi
renvoyer t
```

Algorithme 6 – Date du prochain réveil dans un cycle i.

Il faut dire que SLACK-MAC permet aux nœuds d'adapter leur activité en fonction de celles de leur voisinage. Ce qui améliore la probabilité de communications fructueuses, qui à son tour réduit le temps d'attente avant une activité commune. SLACK-MAC se sert de l'état de la file d'attente des nœuds pour optimiser les communications, ce qui réduit le taux de perte donc augmente le taux de livraison des données. Cependant, SLACK-MAC est moins réactif face aux changements de voisinage par rapport au protocole MAC de base (proposé dans la partie 3.1.2).

3.2 Protocoles de routage pour faible taux d'activité

L'objectif des protocoles de routage est de trouver le meilleur chemin pour atteindre la destination. Un protocole de routage pour des RCSF doit faire face aux contraintes liées aux nœuds capteurs telles que la faible mémoire, la faible capacité de stockage et l'énergie limitée. Ces protocoles doivent également faire face aux contraintes des protocoles MAC sur lesquels ils sont basés. L'une des contraintes fondamentales liées aux protocoles MAC est l'activation et la désactivation du module radio des nœuds capteurs. Plus le pourcentage de temps dans lequel les nœuds laissent leur module radio actif est faible, plus longue est la durée de vie du réseau comme introduit dans [LG09]. Cependant, un faible taux d'activité se traduit soit par une forte contention pour l'accès au médium (quand le protocole MAC est synchrone), soit par une rareté des activités communes entre voisins (quand le protocole MAC est asynchrone). Or, la plupart des protocoles de routage nécessite le maintien d'une table de voisinage ou d'informations de routage (par exemple, un arbre logique, ou la connaissance d'un coût jusqu'à la destination). Cette opération de maintenance nécessite plusieurs messages de contrôle et est difficile à réaliser lorsque les activités communes entre les nœuds voisins sont rares. En conséquence, la plupart des protocoles de routage ne sont pas adaptés aux RCSF, et ceux qui sont déjà basés sur un mécanisme MAC à taux d'activité donnent de faibles performances quand le taux d'activité est faible.

Dans ce qui suit, nous proposons deux protocoles de routage qui font face à la rareté des liaisons et sont adaptés pour un nombre important de nœuds. L'un est basé sur un mécanisme de gradient et l'autre est basé sur un mécanisme d'inondation. Tous les deux utilisent le protocole A_{aa}MAC proposé dans la partie 3.1.2 comme protocole MAC.

3.2.1 Protocole de routage par gradient pour faible taux d'activité

Les protocoles de routage par gradient sont basés sur le calcul d'un coût nommé gradient, qui représente le coût minimum pour atteindre la destination finale. Ce coût peut être calculé selon le nombre de sauts par rapport au puits, l'énergie résiduelle, la fiabilité des liens, etc. Généralement, les nœuds qui sont proches du puits ont un gradient plus faible que les autres. Chaque nœud maintient sa valeur de gradient pour déterminer de façon opportuniste un nœud intermédiaire pour ses paquets de données.

Dans cette partie, nous proposons ADC-GRAB (pour *Asynchronous low Duty-Cycle GRAdient Based routing protocol*), un protocole de routage par gradient basé sur le protocole A_{aa}MAC proposé dans la partie 3.1.2.

Dans ce qui suit, nous décrivons dans un premier temps le fonctionnement de ADC-GRAB lors de la mise en place du réseau. Ensuite, nous décrivons l'algorithme de routage en tant que tel. Puis, nous décrivons le mécanisme de mise à jour du gradient. Enfin, nous faisons un résumé des avantages et des inconvénients du protocole ADC-GRAB.

3.2.1.1 Mise en place du réseau avec le protocole ADC-GRAB

ADC-GRAB nécessite un mécanisme d'estimation du gradient lors de la mise en place du réseau. Cette étape est une phase d'initialisation qui opère comme celle effectuée dans le protocole GRAB [YZLZ05].

Dans GRAB, lorsqu'un nœud reçoit un paquet ADV il estime immédiatement son gradient en incrémentant de 1 le gradient contenu dans le paquet reçu. Toutefois, ce mécanisme ne prend pas en compte l'évolution des liaisons radios qui peuvent varier en fonction de la pluie ou des changements de température. Dès lors, un nœud n peut entendre un autre nœud v qui est en réalité assez éloigné de lui et estimer son gradient en fonction de v qu'il entend assez rarement. La conséquence directe est que n aura du mal à trouver d'autres nœuds avec un gradient plus petit que son gradient à lui pour acheminer ses paquets de données.

Dans ADC-GRAB, nous fixons un seuil de robustesse pour le RSSI (pour *Received Signal Strength Indication*) des paquets *Hello* (similaire aux paquets ADV dans GRAB). Au départ, seul le puits connaît son gradient qui est fixé à 0, tous les autres nœuds ont un gradient inconnu. Le puits diffuse en premier un paquet *Hello*. Lorsqu'un nœud reçoit un paquet *Hello*, si c'est sa première réception de paquet *Hello* avec un RSSI supérieur ou égal au seuil défini [5], il fixe son gradient à celui contenu dans le paquet reçu incrémenté de 1, puis insère son gradient dans un paquet *Hello* et commence à le rediffuser. Sinon (si son gradient est déjà fixé) il incrémente de 1 le gradient contenu dans le paquet *Hello* reçu et le rediffuse. Notons qu'un nœud rediffuse au maximum les trois premiers paquets *Hello* reçus pour éviter l'explosion de l'inondation des paquets *Hello*. Chaque nœud maintient une table de ses voisins ayant un gradient inférieur, appelée table des potentiels relais dans la suite. Cette table comporte l'identifiant, le gradient ainsi que la fréquence de réception (qui re-

5. En raison de certains liens peu fiables qui sont rarement disponibles, nous prenons en compte la qualité du lien avec comme critère la puissance estimée à la réception d'un paquet qui doit être au moins égale à *seuil*

présente le nombre de fois qu'un nœud entend un voisin) de tous les voisins ayant un gradient inférieur.

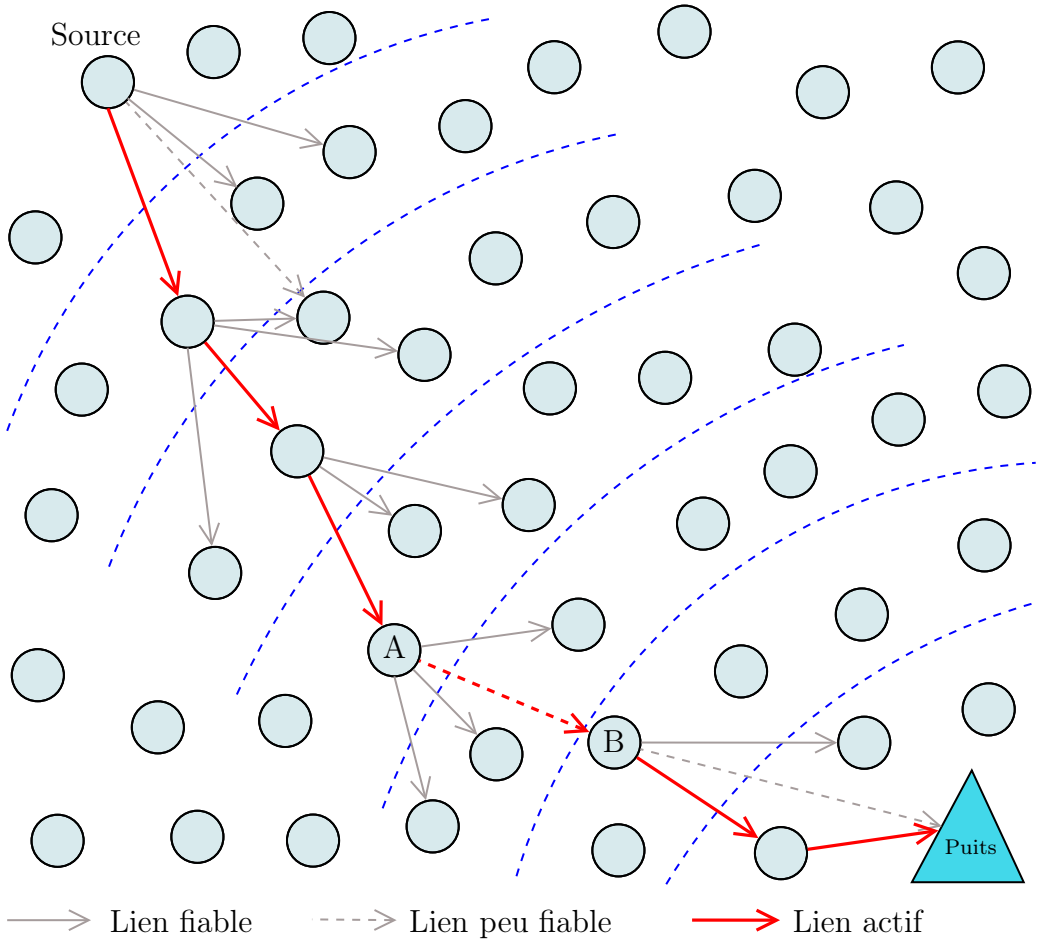


Figure 3.19 – Mécanisme de transmission des paquets de données dans ADC-GRAB, à partir d'un nœud source ayant un gradient égal à 7. Les paquets de données sont transmises en *unicast* et chaque nœud profite d'une rencontre opportuniste avec un potentiel relais pour envoyer les paquets de données.

A la fin de cette petite phase d'initialisation, chaque nœud connaît son gradient en terme de nombre de sauts (noté *grad*) pour atteindre la destination finale et peut se servir d'une rencontre opportuniste avec l'un de ses voisins ayant un gradient inférieur à son gradient comme nœud intermédiaire pour ses paquets de données. Il faut noter qu'un nœud peut changer sa valeur de gradient s'il entend un autre nœud de meilleur gradient avec au moins la même fréquence de réception que le nœud avec lequel il a fixé son gradient.

La figure 3.19 montre un exemple de transmission des paquets de données d'un nœud source positionné à un gradient égale à 7 du puits. Chaque nœud profite d'une rencontre opportuniste (en fonction des dates de réveil gérées par la sous-couche

MAC) pour envoyer ses paquets de données. Tous les paquets de données sont transférés en *unicast*. Sur cet exemple, un nœud A se sert de l'un des liens peu fiables (lien $A \rightarrow B$) pour relayer ses paquets de données, et gagne ainsi une avance.

3.2.1.2 Algorithme de routage dans le protocole ADC-GRAB

Dans ADC-GRAB, un nœud n avec un gradient $grad_n$ se sert toujours d'un voisin v avec un gradient $grad_v$ inférieur à son gradient à lui ($grad_v < grad_n$), comme nœud intermédiaire pour ses paquets de données. En dehors des paquets *Hello* de la phase d'initialisation, ADC-GRAB ne nécessite pas de paquets de contrôle supplémentaires pour mettre à jour la valeur du gradient. Par mesure d'optimisation, le format d'une trame balise générée au niveau de la sous-couche MAC (voir la figure 3.12) est légèrement modifié pour permettre l'insertion de la valeur du gradient $grad_n$ (voir la figure 3.20). Ainsi, la sous-couche MAC envoie une notification à la couche réseau chaque fois que le nœud passe en période d'activité ou d'inactivité, quand un voisin passe en période d'activité, ou encore quand elle échoue une transmission après quatre tentatives pour un même paquet de données. Cette notification comporte l'identifiant *Id-Source* du voisin v, la durée restante de son activité a_r, la valeur de son gradient $grad_v$ et le type de notification.

Type	Id-Source	d_n	r_n	a_r	$grad_n$

Figure 3.20 – Structure d'une trame balise de la couche MAC avec ADC-GRAB.

Notification venant de la sous-couche MAC
si c'est une notification d'activité d'un voisin v **alors**
  Notification(1)
sinon
  si c'est une notification d'échec de transmission **alors**
    Notification(2)
  sinon
    n supprime toutes les entrées de sa liste de nœuds intermédiaires (n est en fin d'activité)
  finsi
finsi

Algorithme 7 – Algorithme de notification de la sous-couche MAC dans ADC-GRAB.

Le mécanisme de routage dans ADC-GRAB est décrit dans l'algorithme 7 et opère

comme suit : quand un nœud n reçoit une notification venant de la sous-couche MAC, cette notification peut être soit une notification du début de l'activité du nœud lui-même, soit une notification de l'activité d'un potentiel relais (voir fonction Notification(1) décrite dans l'algorithme 8)), ou soit une notification d'un échec de transmission (voir fonction Notification(2) décrite dans l'algorithme 9), soit la fin de la période d'activité de n.

Fonction Notification(1) : un nœud n reçoit une notification d'activité d'un voisin v
si $grad_v < grad_n$ **alors**
  si la liste de nœuds intermédiaires (potentiels relais actifs) de n est vide **alors**
    n ajoute v à sa liste de nœuds intermédiaires
    si la file d'attente de n n'est pas vide **alors**
      v devient le nœud intermédiaire pour les paquets de données de n
      n envoie une notification à la sous-couche MAC pour envoyer les paquets de données *via* v
    finsi
  sinon
    si v n'est pas déjà dans la liste de nœuds intermédiaires **alors**
      n ajoute v à sa liste de nœuds intermédiaires
    finsi
  finsi
  si v fait partie de la table des potentiels relais de n **alors**
    n incrémente la fréquence de réception de v
  sinon
    n ajoute une nouvelle entrée à sa table avec une fréquence de réception de 1
  finsi
finsi

Algorithme 8 – Fonction de notification d'activité d'un voisin.

Les points suivants spécifient les actions effectuées pour chaque type de notification.

- La fonction Notification(1) décrite dans l'algorithme 8 est exécutée quand un nœud n reçoit une notification d'activité d'un potentiel relais v. Le nœud n vérifie d'abord si le gradient $grad_v$ du nœud v est inférieur à son gradient à lui. Si $grad_n < grad_v$, n ignore la notification. Sinon, n ajoute v à sa liste de nœuds intermédiaires (potentiels relais actifs), et si n n'avait pas déjà un nœud intermédiaire (c'est-à-dire si sa liste de nœuds intermédiaires était vide),

v devient le nœud intermédiaire pour les paquets de données de n. Le nœud n peut envoyer ses paquets de données *via* v si sa file d'attente n'est pas vide en envoyant une notification à la sous-couche MAC. Le nœud n met ensuite à jour sa table de potentiels relais. Il commence par vérifier si v fait partie de sa table de potentiels relais. Si c'est le cas, n incrémente de 1 la fréquence de réception de v. Sinon, n ajoute une nouvelle entrée à sa table de potentiels relais avec une fréquence de réception de 1 (cette table servira pour la mise à jour du gradient).

- Quand un nœud n reçoit une notification d'échec de transmission, la fonction Notification(2) (voir algorithme 9) est exécutée. Le nœud n supprime le nœud intermédiaire courant de sa liste et détermine une nouvelle destination, si cette liste n'est pas vide.
- Quand un nœud n reçoit la notification de fin de sa période d'activité, toutes les entrées de sa liste de nœuds intermédiaires sont supprimées.

Fonction Notification(2) : un nœud n reçoit une notification d'échec de transmission
n supprime v de sa liste de nœuds intermédiaires
si la liste de nœuds intermédiaires n'est pas vide **alors**
  n choisit un autre nœud intermédiaire pour ses paquets données
finsi

Algorithme 9 – Fonction de notification d'échec de transmission.

3.2.1.3 Maintenance de la topologie dans ADC-GRAB

Dans le protocole de routage ADC-GRAB, il n'y a pas d'échange de paquets de contrôle (en dehors des paquets *Hello* de la phase d'initialisation et des balises de la sous-couche MAC) de façon périodique pour mettre à jour la valeur du gradient. La valeur du gradient $grad_n$ d'un nœud n est insérée dans tous les paquets (y compris les balises). La valeur du gradient est mise à jour de façon dynamique grâce à une table de potentiels relais, mise à jour à chaque notification d'activité du voisinage.

Lorsqu'un nouveau nœud k rejoint le réseau, la fonction Join() décrite dans l'algorithme 10 est exécutée pour déterminer son gradient. Ce nœud écoute pendant au maximum une durée équivalente à celle de la durée d'un cycle spécifiée dans le protocole A_{aa}MAC pour s'assurer d'entendre tous ses voisins. Pendant ce temps d'écoute, il maintient une table de voisinage (notée $table_{temp}$) de ses voisins v entendus avec un RSSI supérieur au seuil fixé et de la valeur de leur gradient $grad_v$. Ensuite, il fixe

la valeur de son gradient $grad_k$ au minimum de ceux reçus incrémenté de 1. La table des potentiels relais de k comporte tous les nœuds voisins v qui ont une valeur de gradient $grad_v$ inférieur à la valeur de $grad_k$.

Fonction Join()
Pendant une durée t (équivalent à la durée d'un cycle niveau MAC), construire une table de voisinage $table_{temp}$ contenant les valeurs de gradient $grad_v$ des voisins entendus
$min \leftarrow$ minimum des gradient de $table_{temp}$
$grad_v \leftarrow min + 1$

Algorithme 10 – Fonction de détermination de la valeur du gradient $grad_k$ quand un nœud k rejoint le réseau.

L'algorithme 11 décrit le mécanisme de mise à jour du gradient dans le protocole ADC-GRAB.

Entrée : $table_1$, $table_2$ $table_3$ et $table_4$ contiennent respectivement $niveau_1$, $niveau_2$, $niveau_3$, et $niveau_4$ entrées
Après chaque P unités de temps **faire**
pour chaque entrée v de la table $table_1$ **faire**
  si v existe dans $table_2$, $table_3$ et $table_4$ **alors**
    si la fréquence de réception de v dans $table_1$ est la même que celle dans $table_2$, $table_3$, et $table_4$ **alors**
      v est retiré des quatre tables
    finsi
  finsi
finpour
si $niveau_1 > 0$ **alors**
  copie de $table_3$ dans $table_4$, de $table_2$ dans $table_3$, et de $table_1$ dans $table_2$
sinon
  exécuter la fonction Joint() décrite dans l'algorithme 10
finsi

Algorithme 11 – Algorithme de maintenance du gradient dans ADC-GRAB.

Chaque nœud maintient un historique de la table des potentiels relais avec une taille de quatre (de $table_1$ à $table_4$, avec $table_1$ la table qui est mise à jour après chaque notification d'activité d'un voisin par la sous-couche MAC). De façon périodique avec une période P (fixée par exemple à 10 fois la durée du cycle des nœuds au niveau MAC), les nœuds mettent à jour leurs différentes tables en copiant $table_3$ dans $table_4$,

$table_2$ dans $table_3$, et $table_1$ dans $table_2$. Après chaque P unités de temps, un nœud vérifie si la fréquence de réception d'un voisin est identique dans ces quatre tables. Si elle n'évolue pas, le nœud suppose que le lien entre lui et ce potentiel relais n'existe plus et supprime ce nœud de ses quatre tables. S'il existe toujours des entrées dans $table_1$, ce nœud conserve sa valeur de gradient. Sinon, ce nœud envoie à la couche MAC une notification de demande d'activité qui dure un cycle, afin de permettre au nœud d'écouter la balise de tous ses voisins et de réestimer son gradient. Ce nœud se comporte exactement comme un nouveau nœud rejoignant le réseau en exécutant la fonction Join(). De même, si un nœud entend régulièrement un voisin avec un meilleur gradient, il met à jour la valeur de son paramètre $grad$.

Il faut noter que la partie concernant l'apparition et la disparition de nœuds du protocole ADC-GRAB n'a pas été évaluée dans ce travail. Cette partie sera mise en œuvre et testée dans nos travaux futurs.

3.2.1.4 Avantages et inconvénients du protocole ADC-GRAB

Les avantages du protocole de routage ADC-GRAB peuvent se résumer comme suit.

- Il ne nécessite pas d'échange de paquets de contrôle pour mettre à jour les gradients (en dehors des paquets *Hello* de la phase d'initialisation et des balises de la sous-couche MAC).

- Il ne nécessite pas la construction d'un chemin préalable pour les transmissions, car il s'assure que tous les chemins sont exploités de façon opportuniste. En effet, un nœud se sert toujours de n'importe quel voisin ayant un gradient inférieur comme potentiel relais pour ses paquets de données.

- Le plus court chemin est utilisé pour transmettre les paquets de données, ce qui réduit le délai de bout en bout pour la livraison des paquets de données.

Le principal inconvénient de ADC-GRAB est qu'il nécessite que les nœuds maintiennent une table de leur voisinage. Aussi, ADC-GRAB nécessite que les périodes d'activités communes entre les nœuds soient assez grandes pour échanger au moins deux paquets de contrôles et un paquet de données, ce qui peut empêcher son utilisation pour des taux d'activité faibles (par exemple moins de 1 %).

3.2.2 Protocole de routage par inondation pour faible taux d'activité

Les protocoles de routage par inondation envoient les paquets dans tout le réseau, ce qui fait que le plus court chemin (parmi d'autres chemins) est utilisé pour atteindre une destination donnée. Les protocoles de routage par inondation sont généralement très simples à mettre en œuvre, mais ils sont souvent coûteux en termes de bande passante et d'augmentation de la charge du réseau, en raison du nombre important de copies des paquets de données.

Dans cette partie, nous montrons comment la combinaison d'un protocole de routage basé sur l'inondation avec un protocole MAC asynchrone assure de bonnes performances avec des taux d'activité faibles. Nous avons proposé pour cela le protocole E-ADCR (pour *Energy-efficient Asynchronous low Duty-Cycle Routing protocol*) dans [AGM15], un protocole d'inondation efficace, qui tire avantage du protocole A_{aa}MAC de la partie 3.1.2.

3.2.2.1 Objectifs principaux du protocole E-ADCR

Les objectifs du protocole E-ADCR peuvent se résumer comme suit.

- Il doit bénéficier des activités communes rares et courtes pour réduire le nombre de copies des paquets de données générées par le mécanisme d'inondation.
- Il doit assurer la réception de paquets par les récepteurs grâce à l'envoi répété des paquets au court de la période active des nœuds.
- Il doit utiliser une politique adaptée de gestion de file d'attente afin de s'assurer que les nouveaux paquets aient plus de chance d'être reçus par un récepteur et que les paquets ne restent pas indéfiniment dans les files d'attente des nœuds.

3.2.2.2 Fonctionnement du protocole E-ADCR

Dans E-ADCR, les nœuds envoient constamment leurs paquets (en utilisant le protocole MAC de la partie 3.1.2) pendant leur période active sans attendre qu'un potentiel récepteur soit détecté. Cela conduit à ce que, dès que deux nœuds se rencontrent, ils commencent à échanger des paquets de données, plutôt que de perdre du temps (au sein d'une fenêtre de temps déjà courte) à échanger des paquets de contrôle. Il est clair que cette approche augmente la concurrence pour l'accès au canal, cependant, nous considérons que cette augmentation est faible étant donné que la probabilité d'avoir beaucoup de nœuds actifs simultanément est elle aussi faible (vu le faible taux d'activité). Ainsi, l'impact sur l'énergie est limité, vu que les nœuds

capteurs consomment environ autant (voire moins) d'énergie lors de la transmission que lors de l'écoute.

Dans E-ADCR, le puits diffuse initialement un paquet *Hello* avec un paramètre *saut* (qui représente la distance en termes de nombre de sauts d'un nœud par rapport puits) fixé à 0 pour le puits. Dans E-ADCR (comme dans ADC-GRAB) nous fixons un seuil de robustesse pour le RSSI des paquets *Hello*. Tous les nœuds qui entendent un paquet *Hello* avec une puissance de réception supérieure à un certain seuil de robustesse défini (noté *seuil*), fixent leur paramètre *saut* au minimum de leur saut actuelle et de la valeur contenue dans le paquet. Puis, ils rediffusent le paquet *Hello* avec le nouveau *saut* et passent en mode sommeil jusqu'à leur prochain réveil.

Après une courte phase d'initialisation, chaque nœud a une estimation de la distance en terme de nombre de sauts qui le sépare du puits. On peut noter qu'il est possible de mettre à jour le paramètre *saut* en cas de changement de topologie ou en situation de mobilité. Pour cela, il suffit d'insérer le paramètre *saut* dans tous les paquets de données et de le ré-estimer en cas de réception d'un meilleur minimum. Ce paramètre *saut* permet à chaque nœud de fixer le paramètre TTL (pour *Time-To-Live*), qui représente le nombre total de liens que peut parcourir un paquet.

La figure 3.21 montre la structure d'un paquet dans E-ADCR. Chaque paquet possède en plus des données utiles, les paramètres suivants : le paramètre TTL est fixé par les nœuds lors de la génération de leur paquet, avec la formule $TTL = 2 \times saut$. Le facteur 2 permet de réduire le nombre de rediffusions. Le paramètre $InstantF$ représente la date de la mise en file d'attente du paquet et est mis à jour dans chaque paquet avant sa mise en file d'attente, au moment de la réception du paquet. Chaque paquet contient également un numéro unique (composé de l'identifiant du nœud l'ayant généré *Id-Source*, et du numéro de séquence *Num-Séquence*), et le paramètre *Id-Dest* représente l'identifiant du nœud puits.

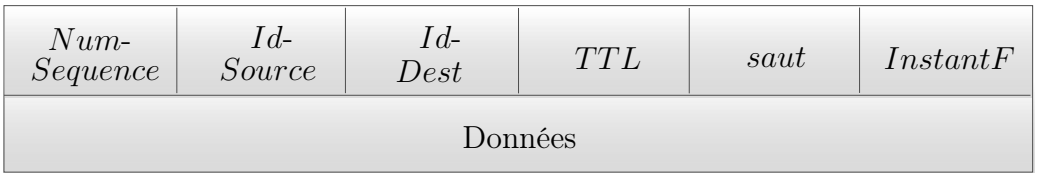


Figure 3.21 – Structure d'un paquet dans E-ADCR.

Le protocole E-ADCR utilise une file d'attente de paquets notée *queue*, représentée sur la figure 3.22. La partie (a) montre le cas où la file d'attente n'est pas pleine : les nouveaux paquets reçus sont mis en fin de file. Lorsqu'un nouveau paquet arrive et que la file d'attente est pleine, une place est crée en supprimant le plus ancien paquet de la file (voir la partie (b)). Le plus ancien paquet de la file d'attente (position 1)

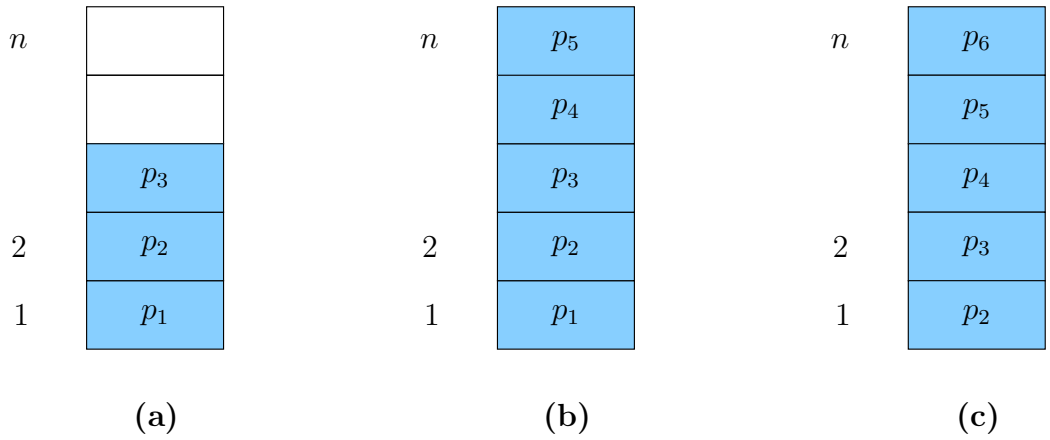


Figure 3.22 – Mécanisme de mise en file d'attente des paquets dans E-ADCR.

est retiré de la file et tous les autres paquets sont décalés vers le bas pour libérer la dernière place de la file en vue d'ajouter le nouveau paquet (voir la partie (c)).

3.2.2.3 Détails du protocole E-ADCR

Nous décrivons l'algorithme de routage et le mécanisme de maintenance de la topologie de E-ADCR dans cette partie.

Algorithme de routage dans E-ADCR

Le protocole de routage E-ADCR en lui-même est décrit dans l'algorithme 12 et opère comme suit.

- Quand un nœud reçoit un paquet, si ce paquet a atteint sa destination, ce paquet est envoyé aux couches supérieures. Sinon, le nœud vérifie s'il n'a pas déjà ce paquet dans sa file d'attente et si $TTL > 0$. Si c'est le cas, il décrémente le paramètre TTL et met le paquet en file d'attente. Sinon, le nœud ignore le paquet.
- Quand un nœud est actif et a des paquets dans sa file d'attente, il commence à diffuser ses paquets du haut de la file d'attente [6] vers le bas, jusqu'à la fin de son activité. Les paquets qui sont envoyés ne sont pas systématiquement retirés de la file d'attente car il est probable que ces paquets ne soient pas effectivement reçus (rappelons qu'un nœud envoie ses paquets sans savoir au préalable si un potentiel récepteur est actif).

6. La durée de la période d'activité (niveau MAC) étant courte, les nœuds ont un nombre limité de paquets à diffuser par activité. La priorité est donc donnée aux derniers paquets mis en file d'attente. Des tests réalisés par simulation ont confirmé que diffuser les paquets à partir du dernier paquet mis en file est plus efficace.

```
Entrée le nœud n commence son activité
si la file d'attente de n n'est pas vide alors
    niveau ← position du dernier paquet dans la file
    pour tous les paquets p de la file d'attente faire
        si dateActuelle − InstantF(p) ≥ DuréeMax alors
            supprimer le paquet p de la file d'attente
            niveau ← niveau − 1
        finsi
    finpour
finsi
k ← position du dernier paquet dans la file
tant que nœud n est actif faire
    si nœud n reçoit un paquet p alors
        si Id-Dest(p) = n alors
            envoyer p aux couches supérieures
        sinon
            si p n'est pas déjà en file et TTL(p) > 0 alors
                TTL(p) ← TTL(p) − 1
                InstantF(p) ← dateActuelle
                si file de taille j non pleine alors
                    mettre p dans la file d'attente à la position j+1
                sinon
                    pour tous les paquets dans la file d'attente (de
                    x=0 à x=fin-1) faire
                        paquet position x ← paquet position x+1
                    finpour
                    mettre p en fin de file d'attente
                finsi
            finsi
        finsi
    finsi
    si la file d'attente de n n'est pas vide alors
        n diffuse le paquet en position k de la file (dernier paquet mis en
        file d'attente)
        k ← k − 1
        si k = 0 alors
            k ←position du dernier paquet mis en la file
        finsi
    finsi
fintantque
```

Algorithme 12 – Algorithme du protocole de routage par inondation.

- Quand un nœud a envoyé tous les paquets qui sont dans sa file d'attente et qu'il est toujours en activité, il redémarre la diffusion à partir du haut de la

file d'attente. Sur l'exemple de la figure 3.22 (a), le nœud enverra p_3, p_2, p_1, p_3, p_2, etc. Ce retour au début vient du fait qu'il est possible qu'un nœud voisin commence son activité vers la fin de l'activité du nœud courant: si les paquets ne sont pas retransmis, le voisin n'aura pas la possibilité de les recevoir.

- En tout début d'activité, un nœud élimine de sa file d'attente les paquets dont la durée de séjour a expirée. Le maximum de cette durée est notée $DuréeMax$ (fixée par exemple à 10 fois la durée du cycle des nœuds au niveau MAC). Si la différence entre la date actuelle, notée $dateActuelle$, et la date de mise en file d'attente $InstantF$ est supérieure ou égale au temps $DuréeMax$, le paquet est supprimé de la file.

Maintenance de la topologie dans E-ADCR

Le mécanisme de maintenance de la topologie prend en charge l'évolution des liens, l'apparition de nouveaux nœuds, la défaillance et la mobilité de certains nœuds. Cette maintenance nécessite dans la plupart des protocoles de routage une mise à jour périodique des informations sur la topologie pour assurer efficacement le routage. Cependant, la mise à jour requiert généralement la génération d'un nombre important de messages de contrôle, donc une consommation supplémentaire en énergie. Aussi, fixer une fréquence de mise à jour suppose que l'on peut prévoir la périodicité des changements dans le réseau.

Le protocole de routage E-ADCR ne nécessite pas de mécanisme de mise à jour pour faire face aux variations de topologie. Le paramètre *saut* est simplement inséré dans tous les paquets de données diffusés par les nœuds.

La fonction Join() décrite dans l'algorithme 10 est exécutée quand un nouveau nœud k rejoint le réseau. Ce nœud écoute pendant au maximum une durée équivalente à celle de la durée d'un cycle spécifiée dans le protocole A_{aa}MAC pour s'assurer d'entendre tous ses voisins. Pendant ce temps d'écoute, il maintient une table temporaire (notée $table_{temp}$) de la valeur du paramètre *saut* de ses voisins entendus avec un RSSI supérieur au seuil fixé. Ensuite, il fixe la valeur de son paramètre *saut* au minimum de ceux reçus incrémenté de 1.

3.2.2.4 Avantages et inconvénients d'E-ADCR

Les avantages de notre protocole de routage par inondation peuvent se résumer comme suit.

- Il ne nécessite pas d'échange de paquets de contrôle avant l'envoi des paquets de données (en dehors des paquets *Hello* de la phase d'initialisation), ni de

connaissance du voisinage des nœuds pour fonctionner.

- Il tire avantage de la simplicité et de l'efficacité de toujours utiliser le plus court chemin, comme tous les protocoles de routage par inondation.
- Il n'hérite pas du nombre important de copies de paquets des protocoles d'inondation traditionnels. En effet, le protocole MAC fournit non seulement des périodes d'activités communes entre les nœuds très réduites (ce qui est dû au taux d'activité faible), mais le nombre moyen de nœuds actifs à un instant donné est toujours faible, donc il y a peu de copies des paquets de données et une faible contention pour l'accès au médium.
- Il ne nécessite pas d'échange de messages de contrôle pour mettre à jour le paramètre *saut*.
- Les paquets stockés dans les files d'attente sont détruits après un certain temps. De plus, le mécanisme de gestion de la file d'attente permet également d'éliminer les anciens paquets quand c'est nécessaire (quand la file est pleine et qu'il y a un nouveau paquet).

Le principal inconvénient du protocole E-ADCR est qu'il peut générer un nombre important de copies des paquets de données s'il opère avec un taux d'activité relativement grand (par exemple 5 %).

3.3 Résumé des différentes propositions

En résumé, nous avons proposé différents mécanismes de planification de l'activité des nœuds et choisi le meilleur pour développer le protocole A_{aa}MAC à rencontre aveugle dans lequel les nœuds se réveillent indépendamment et aléatoirement dans chaque cycle. Nous avons utilisé un mécanisme de fragmentation et un mécanisme de construction d'historique des communications réussies pour réduire le délai et augmenter la probabilité des rencontres avec succès.

Ensuite, nous avons proposé un protocole de routage par gradient ADC-GRAB dans lequel tous les paquets de données sont transférés en *unicast*. Un nœud se sert toujours d'un voisin avec un gradient inférieur à son gradient à lui, comme nœud intermédiaire pour ses paquets de données. ADC-GRAB exploite de façon opportuniste tous les chemins disponibles et augmente la fiabilité dans la livraison des données car tous les paquets de données envoyés requièrent un accusé de réception. ADC-GRAB réalise de très bonnes performances avec un taux d'activité supérieur ou égal à 1 %.

Enfin, nous avons proposé un protocole de routage par inondation E-ADCR dans lequel les nœuds envoient constamment leurs paquets (en utilisant le protocole A_{aa}MAC) pendant leur période active sans attendre qu'un potentiel récepteur soit détecté. E-ADCR est capable de réaliser de bonnes performances avec un taux d'activité faible inférieur à 1 %, mais peut générer un nombre important de copies des paquets de données si le taux d'activité est grand.

Chapitre 4

Résultats

Dans ce chapitre, nous définissons d'abord notre environnement de simulation dans la partie 4.1. Dans la partie 4.2, nous présentons les résultats concernant notre protocole MAC et ses différentes améliorations, puis nous les comparons avec les principaux protocoles MAC à taux d'activités existants. Dans la partie 4.3, nous présentons les résultats liés aux protocoles de routage et nous les comparons à l'existant. Dans la partie 4.4, nous présentons les résultats obtenus par expérimentation de nos deux protocoles de routage sur des nœuds capteurs réels.

4.1 Environnement de simulation

Dans cette partie, nous spécifions notre modèle de simulation, les paramètres utilisés pour réaliser nos simulations et les métriques pour cette évaluation retenue.

4.1.1 Modèle de simulation

Nos simulations ont été réalisées avec la version 2.31 du simulateur réseau NS2 [ns202]. NS2 est un simulateur à événements discrets qui cible la communauté de recherche en réseaux. Il offre un large support pour la simulation des protocoles filaires et sans fil pour toutes les couches du modèle OSI. Le simulateur NS2 est écrit en C++ et utilise des scripts OTCL (pour *Object Tool Command Language*) pour la création des commandes et pour la configuration des interfaces. Nous analysons les résultats de simulation en exécutant des scripts PERL. Notons que NS2 a été enrichi par les travaux de notre équipe de recherche pour lui accorder une couche physique IEEE 802.15.4 à la place de la couche physique 802.11 de base.

4.1.2 Paramètres de simulation

Sauf spécifications contraires de notre part, les paramètres communs pour toutes les simulations sont contenus dans le tableau 4.1.

Tableau 4.1 – Paramètres de simulation

Paramètres	**Valeurs**
Zone de déploiement des nœuds	170 m x 170 m
Nombre de nœuds	100
Distance maximale entre les nœuds	30 m
Distance minimale entre les nœuds	1 m
Seuil de réception	-90 dBm
Seuil de détection de la porteuse	-92 dBm
Puissance de transmission	-1 dBm
Modèle de propagation : shadowing	path loss exponent = 2.74
	shadowing deviation = 2.0 dBm
Taille des paquets	30 octets
Taille maximale de la file d'attente	20 paquets
Durée d'une simulation	3600 secondes

Il est important de noter que la taille de la zone de déploiement des nœuds a été adaptée en vu d'obtenir une topologie connexe avec une distance minimale de 1 m et une distance maximale de 30 m (il n'est pas possible d'avoir une topologie connexe au delà d'une zone 170 m x 170 m pour 100 nœuds étant donné que toutes les positions des nœuds sont tirées de façon aléatoire avant la vérification de la connexité avec l'algorithme de **Prim**). Nous supposons que les liens entre les nœuds sont avec pertes (d'où l'utilisation du modèle de propagation shadowing) et nous tenons compte de la congestion (Un nœud peut stocker au maximum 20 paquets en file d'attente, au delà les nouveaux paquets qui arrivent sont ignorés). Les paramètres du modèle de propagation shadowing ont été adaptés pour permettre de couvrir une portée maximale de 30 m entre les nœuds.

4.1.3 Métriques évaluées

Dans nos simulations, nous évaluons spécifiquement le taux de livraison, le délai moyen de livraison des paquets de données, le taux d'activité et la consommation énergétique.

Le taux de livraison des paquets de données (noté T_L) représente le rapport entre le nombre de paquets effectivement reçus par le puits (noté N_R) sur le nombre total

de paquets générés (noté N_G) par les nœuds sources du réseau : $T_L = \frac{N_R}{N_G} \times 100$.

Le délai moyen (ou le temps de latence noté D_L) des paquets délivrés représente le temps moyen entre la date de génération des paquets (noté D_G) par la source et la date de réception effective par le puits (noté D_R). Les principales causes de retard d'un paquet de données sont : le délai de propagation (très faible), le délai de transmission, le délai de traitement et le délai passé en file d'attente. Le délai moyen de livraison est calculé comme suit : $D_L = \frac{1}{N_R} \times (\sum_{x=1}^{N_R} (D_R - D_G))$

Le taux d'activité représente la proportion du temps pendant laquelle les nœuds ont leur composant radio allumé. Par exemple un taux d'activité de 1 % signifie que les nœuds gardent leur radio éteinte 99 % du temps et l'allument uniquement 1 % du temps.

La consommation en énergie des nœuds représente la quantité d'énergie consommée par les nœuds. La formule pour le calcul de l'énergie (E) est : $E = P \times \Delta t = U \times I \times \Delta t$, où P représente la puissance (en Watt), U représente la différence de potentiel (en Volt), I représente l'intensité du courant (Ampère) et Δt représente le temps (en seconde).
Nous calculons la consommation énergétique (en joule) des nœuds comme suit :
$[TxTime \times TxIntensity + ListenTime \times RxIntensity + WakeUpTime \times WakeUpIntensity + IdleTime \times IdleIntensity] \times tension$
où $TxIntensity$, $RxIntensity$, $WakeUpIntensity$ et $IdleIntensity$ représentent respectivement l'intensité du courant consommée en mode transmission, en mode réception, au démarrage de la radio et en mode sommeil. Ces valeurs sont définies pour les composants CC2420 dans [Ins06]. $TxTime$, $ListenTime$, $WakeupTime$ et $IdleTime$ correspondent aux temps passés respectivement en état de transmission, de réception, d'activation et de désactivation du composant radio, et en état de sommeil. Enfin, *tension* représente le voltage de la batterie (égal ici à 1.2 volts par piles).

4.2 Résultats sur les protocoles MAC

Dans cette partie, nous évaluons dans un premier temps les performances du protocole A_{aa}MAC (présenté dans la partie 3.1.2). Dans un second temps, nous appliquons le mécanisme de fragmentation à A_{aa}MAC. Ensuite, nous testons les performances du protocole MAC adaptatif proposé dans la partie 3.1.3 en le comparant avec A_{aa}MAC. Enfin, nous comparons les protocoles A_{aa}MAC et SLACK-MAC avec d'autres protocoles MAC de la littérature.

Chaque point sur les figures suivantes représente une moyenne de 100 répétions

et chaque répétition dure 3600 secondes.

4.2.1 Résultats sur le protocole A_{aa}MAC

Pour évaluer les performances du protocole A_{aa}MAC, nous avons réalisé les simulations dans deux scénarios différents. Dans le premier scénario, nous évaluons le protocole A_{aa}MAC sur une topologie dans lequel un nœud s a k possibilités de nœuds intermédiaires pour atteindre une destination d donnée (voir la figure 4.1). Dans le deuxième scénario, nous évaluons l'impact du mécanisme de fragmentation sur les performances du protocole A_{aa}MAC.

4.2.1.1 Scénario 1 : Évaluation de l'impact de k sur A_{aa}MAC

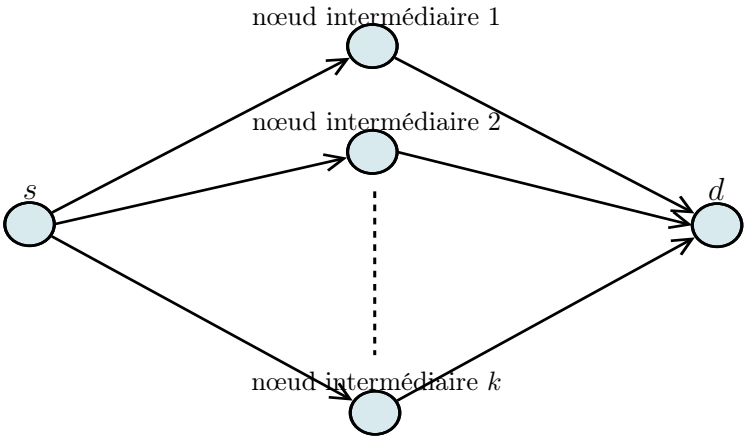


Figure 4.1 – Topologie réseau (appelée topologie en diamant) pour les scénarios 1 et 2 avec un nœud s qui dispose de k possibilités de nœuds intermédiaires pour atteindre d.

Dans ce scénario, nous évaluons la performance de A_{aa}MAC avec un routage par gradient fixé. Pour ce faire, nous considérons une source s à portée de k nœuds intermédiaires. Chacun de ces k nœuds intermédiaires est à portée de la destination d. Ainsi, la source s a k potentiels nœuds intermédiaires pour envoyer ses trames de données à la destination d (considéré comme le puits et supposé toujours en activité), comme présenté sur la figure 4.1. Nous nous référons à cette topologie en diamant (avec k nœuds intermédiaires) et nous déterminons le taux de livraison et le délai de livraison des trames de données de la source à la destination. Sauf indication contraire, la durée du cycle est de 5 s, le taux d'activité est de 5 %, et la source s génère une trame de données toutes les 5 s.

La figure 4.2 montre le taux de livraison des trames de données en fonction de k. On observe que lorsque k varie entre 2 et 6, le taux de livraison des trames de

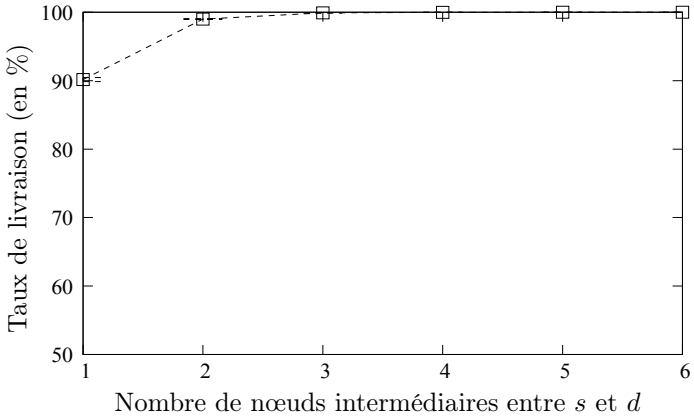


Figure 4.2 – Taux de livraison des trames de données en fonction de k (le nombre de possibilité de nœuds intermédiaires sur la topologie en diamant), avec une période de génération de 5 s et un taux d'activité de 5 %.

données est d'environ 100 %. Lorsque $k = 1$, le taux de livraison est égal à 90 % pour une période de génération d'une trame de données toutes les 5 s. En effet, lorsque le nombre de possibilités de nœuds intermédiaires dans la topologie est faible ($k = 1$), le taux de livraison décroît. Cela est dû au fait que la source a peu de possibilités pour envoyer ses trames de données : les trames remplissent rapidement les files d'attente du nœud, et sont finalement supprimées.

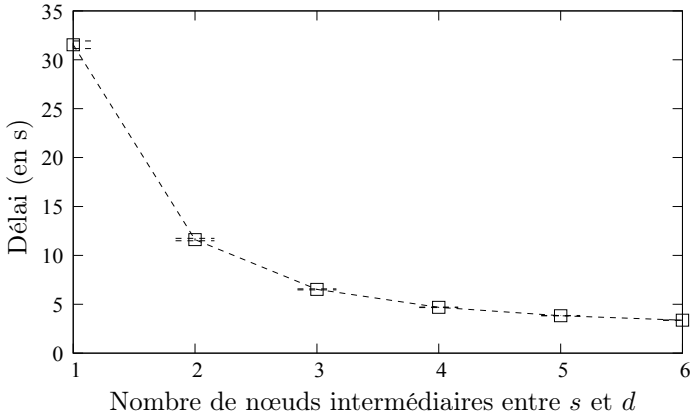


Figure 4.3 – Délai moyen de livraison des trames de données en fonction de k, avec une période de génération de 5 s et un taux d'activité de 5 %.

La figure 4.3 montre le délai moyen de livraison des trames de données en fonction de k. Le délai diminue quand k augmente. En effet, quand k augmente, la source a une plus grande probabilité d'avoir une activité commune avec l'un de ses nœuds intermédiaires, ce qui diminue le délai de livraison des trames de données.

4.2.1.2 Scénario 2 : Évaluation de l'impact de la fragmentation sur A_{aa}MAC

Dans cette partie, nous évaluons l'impact de la fragmentation sur les performances du protocole A_{aa}MAC. Nous utilisons ce scénario pour calculer le taux de livraison et le délai entre la génération des trames de données par la source s et leur réception par la destination d pour différentes valeurs de k. Notre objectif est de trouver la valeur optimale pour la fragmentation f qui est un paramètre important de notre protocole.

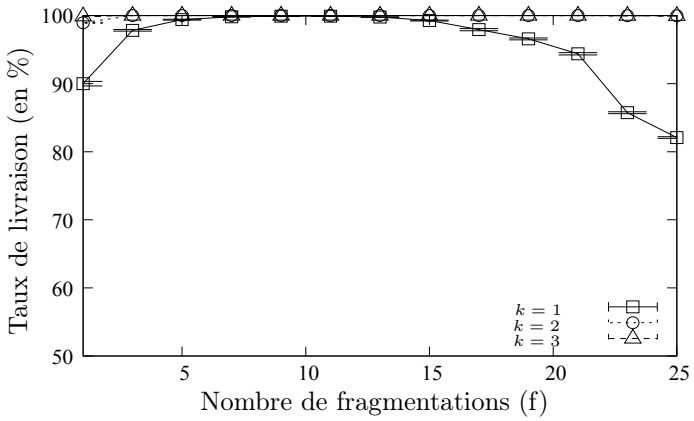


Figure 4.4 – Taux de livraison des trames de données en fonction de la fragmentation f, pour différents k avec un taux d'activité de 5 % et une période de génération de 5 s.

La figure 4.4 montre le taux de livraison des trames de données en fonction du nombre de fragmentation f. Pour $k > 1$ le taux de livraison est de 100 % quand $f \in [5; 15]$ et d'environ 99 % lorsque $f < 5$ ou $f > 15$. pour $k = 1$ le taux de livraison est d'environ 100 % quand $f \in [5; 15]$, est d'environ 90 % quand $f < 5$ et moins de 90 % quand $f > 15$. Ceci peut être expliqué comme suit. Lorsque le nombre de fragmentation est faible ($f = 1$), le temps nécessaire pour que la source ait une activité commune avec un voisin est très grande (elle est supérieure à 25 s, qui correspond à 5 cycles). Ce qui fait que les trames s'accumulent dans la file d'attente, et certaines sont finalement supprimées une fois que la file est pleine. Lorsque le nombre de fragmentation est élevé ($f \geq 15$), le taux de livraison diminue car les nœuds sont actifs pendant de très courtes périodes ($14,7$ ms pour $f = 17$), ce qui donne des activités communes qui ne durent pas toujours assez longtemps pour que la source puisse détecter son voisin et envoyer une trame de données. Cela montre que la valeur de f doit être limitée.

La figure 4.5 montre le délai moyen de livraison des trames de données en fonction du nombre de fragmentation f. Pour les différentes valeurs de k, lorsque le nombre de

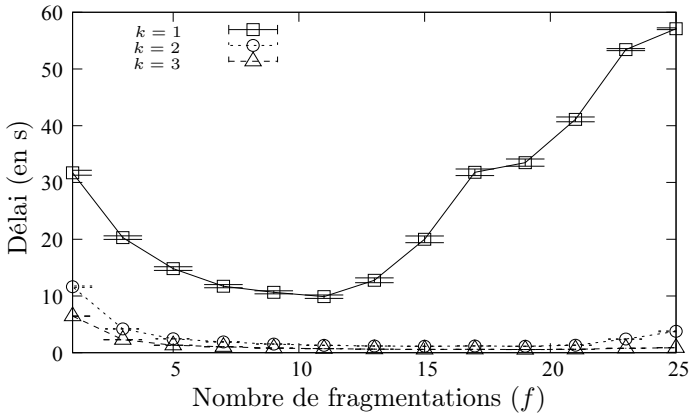


Figure 4.5 – Délai moyen de livraison des trames de données en fonction de la fragmentation f, pour différents k avec un taux d'activité de 5 % et une période de génération de 5 s.

fragmentation est élevé ($f \geq 15$), le délai augmente en raison du fait que les nœuds se rencontrent souvent pour une durée qui est trop courte pour envoyer des trames. Lorsque le nombre de fragmentation est faible ($f \leq 5$), le délai est élevé en raison du temps nécessaire à la source pour rencontrer un voisin.

Avec ces paramètres, le nombre de fragmentation f peut être réglé entre 5 et 15, ce qui correspond à la fois à un taux de livraison grand et un faible délai. Chaque nœud a une activité entre environ 17 ms et 50 ms toutes les 5 s.

4.2.2 Résultats sur le protocole SLACK-MAC

Pour évaluer les performances du protocole SLACK-MAC présenté dans la partie 3.1.3.2, nous avons réalisé des simulations comparant SLACK-MAC au protocole MAC à rencontre aveugle A_{aa}MAC. Les simulations ont été effectuées avec 10 topologies aléatoires de 100 nœuds (ici la densité est de 8 : chaque nœud a en moyenne 8 voisins dans un rayon de 30 m), dont un puits positionné à un coin de la zone. 30 nœuds sources, aléatoirement localisés dans le réseau, effectuent des mesures périodiques (de période P) pour les transmettre de saut en saut au puits (en se servant d'un protocole de routage par gradient pour estimer le gradient des nœuds). Les nœuds ont un taux d'activité de 1 % et le cycle global est de 5 s (ce qui fait que les nœuds sont actifs pendant 50 ms chaque 5 s). Nous déterminons dans un premier temps la valeur optimale des deux paramètres E et R du protocole SLACK-MAC. Ensuite, nous évaluons les performances de SLACK-MAC par rapport à A_{aa}MAC.

4.2.2.1 Méthodologie pour le choix des paramètres dans SLACK-MAC

Afin de déterminer la taille optimale des deux listes E et R de SLACK-MAC. Nous réalisons les tests avec trois valeurs différentes de période de génération de trafic P, et nous faisons varier la taille des listes. Nous supposons que dans les applications que nous visons (du type *convergecast*, c'est-à-dire avec des communications des nœuds vers une seule destination finale), si le puits est positionné à l'extrémité ou au centre, un nœud (en dehors des nœuds de bordure), a toujours plus de nœuds fils (voisins plus loin du puits) que de nœuds pères (voisins plus près du puits). Nous avons estimé cette valeur à environ deux. Ce qui signifie que la taille maximale de la liste R est estimée à deux fois celle de la taille maximale de la liste E, d'où le fait que nous fixons $|R| = 2|E|$. Chaque point, sur les figures qui suivront sont une moyenne de 100 répétitions par topologie.

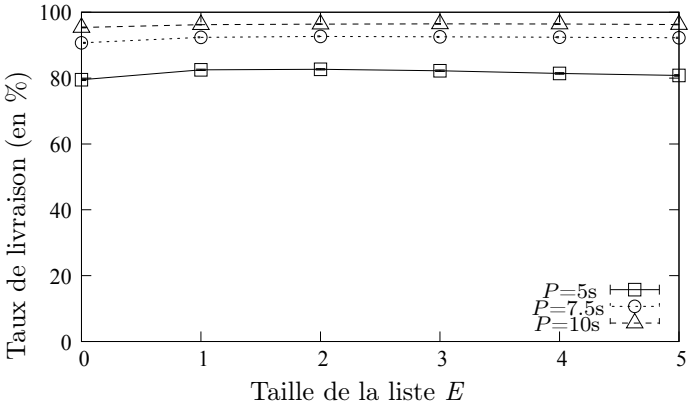


Figure 4.6 – Impact de la taille de la liste E sur le taux de livraison, avec $|R| = 2|E|$ et un taux d'activité de 1 %, pour différentes périodes de générations de trafic P.

Les figures 4.6, et 4.7 montrent respectivement le taux de livraison et le délai moyen de livraison des trames de données, en fonction de la taille de la liste E, où $|R| = 2|E|$. On observe sur ces figures que, indépendamment de la période de générations de trafic P, une taille liste E égale à deux est un bon choix, donc quatre pour la liste R. Ces paramètres sont utilisés dans la suite. Notons qu'une taille totale d'historique de six entrées est réaliste pour les nœuds capteurs (avec une capacité de mémoire limitée). Aussi, nous avons également observé dans nos expériences qu'en moyenne, il faut environ 12 cycles de 5 secondes (60 secondes) pour que les nœuds remplissent leur liste E et environ 50 cycles (250 secondes) pour que les nœuds remplissent leur liste R. Cela montre que la convergence des listes est rapide et négligeable comparée à la durée de vie d'un nœud.

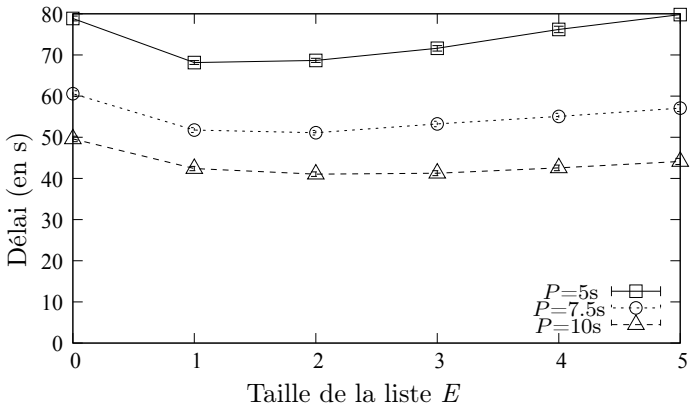


Figure 4.7 – Impact de la liste E de SLACK-MAC sur le délai moyen de livraison, avec $|R| = 2|E|$ et un taux d'activité de 1 %, pour différentes périodes de générations de trafic P.

4.2.2.2 Performances de SLACK-MAC par rapport à A_{aa}MAC

La figure 4.8 montre le taux de livraison des trames de données en fonction de leur période de génération (entre 5 secondes et 20 secondes), avec un taux d'activité de 1 %, pour les protocoles A_{aa}MAC et SLACK-MAC.

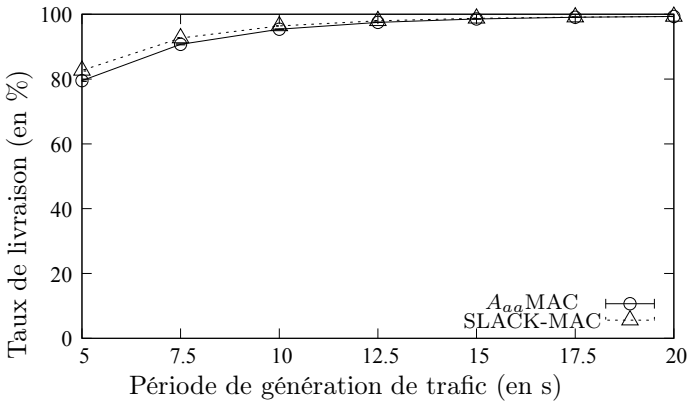


Figure 4.8 – Taux de livraison des trames de données en fonction de leur période de génération, avec un taux d'activité de 1 %, pour les protocoles A_{aa}MAC et SLACK-MAC.

Pour le protocole A_{aa}MAC, le taux de livraison des trames de données augmente de 79 % à 99 % quand la période de génération augmente de 5 secondes à 20 secondes. Le taux de livraison des trames de données est élevé. En effet, lorsque les nœuds se rencontrent, ils peuvent bénéficier pleinement de leur temps de rencontre, étant donné qu'il y a peu de nœuds actifs simultanément.

Pour le protocole SLACK-MAC, taux de livraison des trames de données augmente d'environ 83 % à 99 % quand la période de génération augmente de 5 secondes à 20 secondes. En effet, SLACK-MAC profite d'un mécanisme similaire au protocole

A_{aa}MAC, et permet plus d'activités communes entre les nœuds grâce au mécanisme d'historique implémenté par les listes E et R. Le protocole SLACK-MAC reste également dynamique avec une probabilité de 1/3 de permettre aux nœuds de choisir un mécanisme aléatoire.

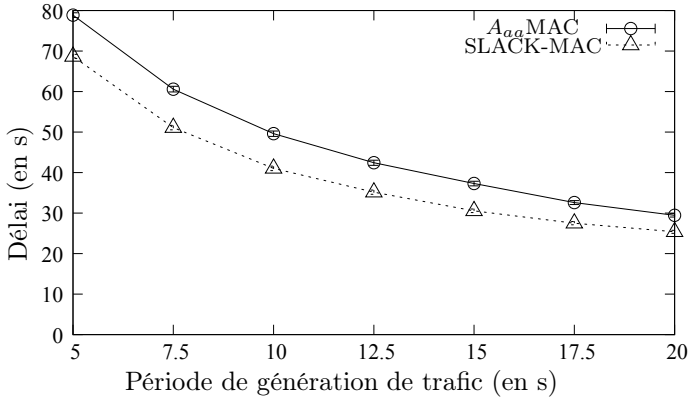


Figure 4.9 – Délai moyen de livraison des trames de données en fonction de leur période de génération, avec un taux d'activité de 1 %, pour les protocoles A_{aa}MAC et SLACK-MAC.

La figure 4.9 montre le délai moyen de livraison des trames de données en fonction de leur période de génération (de 5 secondes à 20 secondes). On peut noter que SLACK-MAC fournit un délai de bout en bout inférieur à celui du protocole A_{aa}MAC. Ce délai décroît de 68 secondes à 25 secondes pour SLACK-MAC et décroît d'environ 79 secondes à 29 secondes pour A_{aa}MAC.

Les résultats montrent que :

- en terme de taux de livraison des trames de données, pour la période de génération de trafic de 5 secondes à 20 secondes, SLACK-MAC fournit un gain allant jusqu'à 3,98 % .
- en terme de délai, SLACK-MAC offre un gain par rapport au protocole A_{aa}MAC allant de 12.90 % à 13.87 %.

4.2.3 Comparaison avec des protocoles MAC de l'état de l'art

Dans cette partie, nous montrons dans un premier temps que les protocoles MAC synchrones ne sont pas adaptés aux faibles taux d'activité. Nous comparons nos protocoles (A_{aa}MAC [AGM14a] et SLACK-MAC [AGLM15]) avec le protocole MAC synchrone de référence qui est le standard IEEE 802.15.4 [IEE11]. Nous comparons dans un deuxième temps nos protocoles avec les protocoles MAC asynchrones X-MAC [BGAH06], RI-MAC [SGJ08] et PW-MAC [TS+11], qui sont des protocoles

MAC à taux d'activité asynchrones représentatifs. Il convient de noter qu'en dehors du standard IEEE 802.15.4 sur lequel est basé ZigBee [Zig08] (qui intègre un protocole de routage en arbre), le même protocole de routage par gradient est utilisé dans nos simulations par tous les protocoles MAC pour router les paquets de données de saut en saut jusqu'au puits. Les simulations ont été effectuées avec 10 topologies aléatoires de 100 nœuds, dont un puits positionné à un coin de la zone. La densité du réseau est de 8 (dans un rayon de 30 m) et le nombre maximum de sauts pour atteindre le puits est de 7. Nous avons 30 nœuds sources aléatoirement localisés dans le réseau, et effectuent des mesures périodiques de période P. Chaque point, sur les figures suivantes est une moyenne de 10 répétitions par topologie.

Il est important de noter que dans nos simulations, les résultats pour le standard IEEE 802.15.4 ne prennent pas en compte le coût de la synchronisation (tous les noeuds sont nativement et parfaitement synchronisés).

Les figures 4.10 et 4.11 montrent respectivement le taux de livraison et le délai moyen de livraison des trames de données, en fonction de leur période de génération, pour les protocoles X-MAC, RI-MAC, PW-MAC, le standard IEEE 802.15.4, A_{aa}MAC et SLACK-MAC. La période de génération des trames varie de 5 s (ce qui correspond à une génération de trafic élevé pour un taux d'activité de 1 %) à 30 s (ce qui correspond à une génération de trafic relativement faible pour ce taux d'activité).

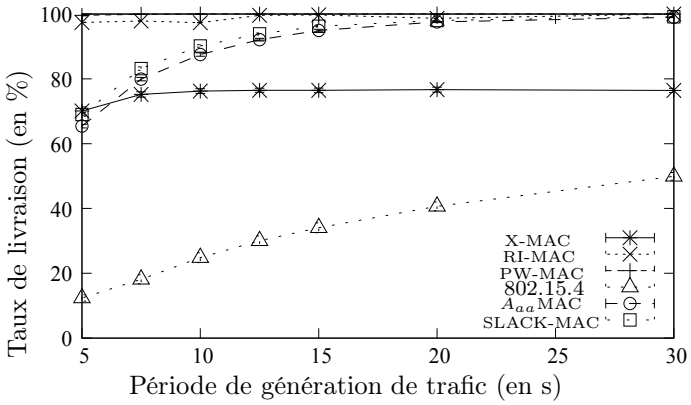


Figure 4.10 – Taux de livraison des trames de données en fonction de leur période de génération, avec un taux d'activité fixe de 1 % pour IEEE 802.15.4, A_{aa}MAC et SLACK-MAC, et avec un taux d'activité variable pour X-MAC, RI-MAC et PW-MAC.

Pour le standard IEEE 802.15.4, les valeurs de BI et SD sont fixés et sont respectivement égales à 5 secondes et 50 millisecondes. Le taux de livraison des trames de données augmente de 12 % à 50 % et le délai moyen de bout en bout pour la livraison des trames de données décroît de 163 s à 44 s. Ce faible taux de livraison des

trames de données est dû à la forte contention, car les nœuds sont tous synchronisés pendant de courtes périodes de temps. Cette forte contention génère de nombreuses collisions et provoque des débordements dans les files d'attente des nœuds, causant de nombreuses pertes des trames de données.

Pour le protocole X-MAC, le taux de livraison des trames de données augmente de 70 % à 76 % et le délai moyen de livraison des trames de données est d'environ 2 s, lorsque la période de génération de trafic varie de 5 s à 30 s. La perte des trames de données se produit principalement en raison du nombre relativement élevé de trames préambules, et aussi parce que l'émetteur n'a pas connaissance de la réception réussie des trames de données par le récepteur. Le faible délai de 2 s est dû au fait que X-MAC ne prévoit pas un taux d'activité fixe pour chaque nœud : quand un nœud a des trames de données à transmettre, il reste actif pour les envoyer.

Pour le protocole RI-MAC, le taux de livraison des trames de données augmente de 97 % à 100 % et le délai moyen de bout en bout pour la livraison des trames de données est d'environ 1 s, lorsque la période de génération de trafic varie de 5 s à 30 s. Le taux de livraison élevé et le très faible délai de livraison sont dûs au fait que RI-MAC ne prévoit pas de taux d'activité fixe pour chaque nœud. RI-MAC réduit l'occupation du canal par rapport à X-MAC, générant ainsi moins de collisions. De plus, les émetteurs dans RI-MAC sont informés de la réception réussie des trames de données par des accusés de réception.

Pour le protocole PW-MAC, le taux de livraison des trames de données est de 100 % et le délai moyen de bout en bout pour la livraison des trames de données diminue de 3 s à 2 s, lorsque la période de génération de trafic varie de 5 s à 30 s. PW-MAC ne prévoit pas de taux d'activité fixe pour chaque nœud. Le délai avec PW-MAC est légèrement supérieur par rapport au délai de RI-MAC parce que les nœuds émetteurs ne restent pas actifs jusqu'à ce que le récepteur se réveille. En effet, les émetteurs prédisent le réveil des récepteurs. Cela permet de réduire la consommation d'énergie, mais conduit à une augmentation du délai moyen à chaque fois qu'il y a un échec de prédiction.

Pour le protocole A_{aa}MAC, le taux de livraison des trames de données augmente de 65 % à 99 % et le délai moyen de bout en bout pour la livraison des données diminue de 156 s à 43 s. Le taux de livraison des trames de données est faible lorsque le trafic est élevé, parce que la durée des activités communes ne suffit pas pour supporter une charge de trafic élevée. En revanche, le protocole A_{aa}MAC réalise un grand taux de livraison des trames de données lorsque la charge de trafic est faible, au détriment d'un délai plus grand.

Pour le protocole SLACK-MAC, le taux de livraison des trames de données aug-

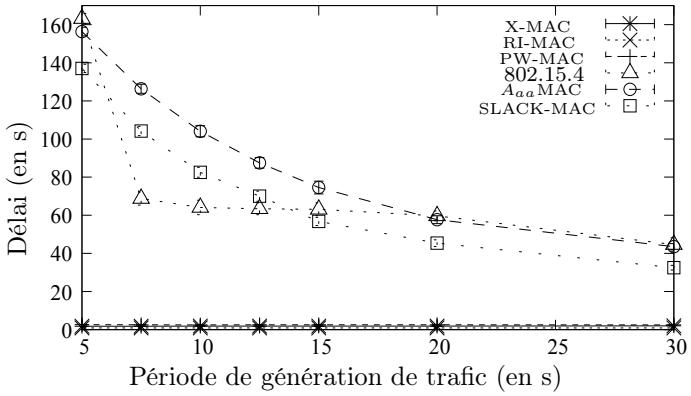


Figure 4.11 – Délai moyen de livraison des trames de données en fonction de leur période de génération, avec un taux d'activité fixe de 1 % pour IEEE 802.15.4, A_{aa}MAC et SLACK-MAC, et avec un taux d'activité variable pour X-MAC, RI-MAC et PW-MAC.

mente de 68 % à un peu plus de 99 % et le délai moyen de bout en bout pour la livraison des données diminue de 137 s à 32 s, lorsque la période de génération de trafic varie de 5 s à 30 s. Le taux de livraison des trames de données est relativement élevé globalement et le délai est relativement faible par rapport A_{aa}MAC, car comme nous l'avons montré dans la partie 4.2.2, SLACK-MAC profite d'une mécanisme similaire au protocole A_{aa}MAC, et permet plus d'activités communes entre les nœuds grâce au mécanisme d'historique réalisé par les listes E et R.

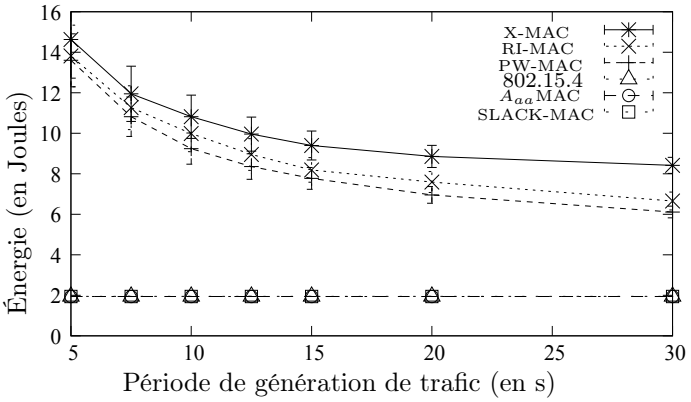


Figure 4.12 – Énergie moyenne consommée par heure par nœud en fonction de la période de génération de trafic, avec un taux d'activité fixe de 1 % pour IEEE 802.15.4, A_{aa}MAC et SLACK-MAC et, avec un taux d'activité variable pour X-MAC, RI-MAC et PW-MAC.

La figure 4.12 montre la consommation d'énergie moyenne par nœud en Joules pour une période de 1 heure, en fonction de la période de génération de trafic (de 5 s à 30 s), pour les protocoles X-MAC, RI-MAC, PW-MAC, le standard IEEE 802.15.4, A_{aa}MAC et SLACK-MAC.

Pour le standard IEEE 802.15.4, le protocole A_{aa}MAC et SLACK-MAC, la consommation d'énergie moyenne est toujours en dessous de 2 J (car tous les nœuds ont un taux d'activité fixe de 1 %). Pour X-MAC, la consommation d'énergie diminue de 15 J à 8 J. Pour RI-MAC, elle diminue de 14 J à 7 J. Pour PW-MAC, elle diminue de 14 J à 6 J. La consommation d'énergie dans X-MAC, RI-MAC et PW-MAC est élevée parce que les nœuds doivent être actifs plus de 1 % du temps quand ils ont des trames de données à envoyer, tandis que les nœuds dans IEEE 802.15.4, A_{aa}MAC et SLACK-MAC, gardent un taux d'activité fixe.

4.2.4 Bilan de comparaison avec les protocoles MAC de l'état de l'art

Les résultats de comparaison montrent que le protocole MAC synchrone n'est pas adaptés à des taux d'activité faibles. De même, les protocoles MAC asynchrones qui ne fonctionnent pas à taux d'activité fixe pour tous les nœuds ne sont pas adaptés pour opérer avec des taux d'activité faibles, étant donné qu'il nécessitent une consommation importante d'énergie. Il est également important de noter que dans ces protocoles la consommation d'énergie n'est pas équitablement répartie entre les nœuds. Par exemple, la consommation d'énergie maximale pour un nœud avec une petite période de génération de trafic de 5 s est de 60 J pour X-MAC, de 61 J pour RI-MAC et de 45 J pour PW-MAC. La consommation d'énergie maximale pour une grande période de génération de trafic de 30 s est de 18 J pour X-MAC, de 23 J pour RI-MAC et de 13 J pour PW-MAC. En résumé, on remarque que A_{aa}MAC et SLACK-MAC fournissent le meilleur compromis en termes de consommation d'énergie, de taux de livraison des trames de données et de délai moyen de livraison pour les applications de surveillance de l'environnement qui nécessitent une longue durée de vie du réseau.

4.3 Résultats sur les protocoles de routage

Dans cette partie, nous évaluons les performances des protocoles de routage E-ADCR et ADC-GRAB, et nous les comparons avec un protocole de routage par inondation de la littérature CF [ZF06] (décrit dans la partie 2.2.3) qui est adapté aux communications de type *convergecast*. Nous les évaluons dans trois scénarios différents en analysant l'impact des faibles taux d'activité, l'impact des taux d'activité important, et enfin l'impact de la densité du réseau sur les performances des différents protocoles. Dans les deux premiers scénarios, les simulations sont effectuées avec 10

topologies aléatoires de 100 nœuds, dont un puits positionné à un coin de la zone. La densité du réseau est de 8 (dans un rayon de 30 m) et le nombre maximum saut pour atteindre le puits est de 7. Trente nœuds sources sont aléatoirement localisés dans le réseau, et effectuent des mesures périodiques de période P. Dans le troisième scénario, nous faisons évoluer le nombre de nœuds du réseau. Chaque point, sur les figures qui suivront est une moyenne de 10 répétitions par topologie.

4.3.1 Impact des faibles taux d'activité

Dans le premier scénario de simulation, nous étudions l'impact des taux d'activité faibles sur les performances des trois protocoles de routage. Nous considérons des taux d'activité qui sont en dessous de 1 % (entre 0.25 % et 1 %).

La figure 4.13 montre le taux de livraison des paquets de données en fonction du taux d'activité (qui varie entre 0.25 % et 1 %), avec à gauche une période de génération de trafic de 5 s (ce qui correspond à une génération de trafic élevée compte tenu du taux d'activité faible) et à droite une période de 60 s (ce qui correspond à une génération de trafic relativement faible). Le taux de livraison augmente avec le taux d'activité pour les trois protocoles.

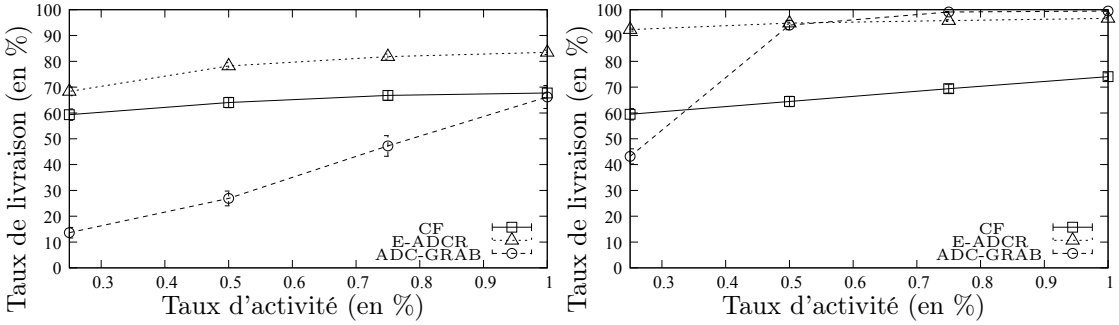


Figure 4.13 – Taux de livraison des données en fonction du taux d'activité, avec une période de génération de trafic de 5 s (à gauche), et une période de génération de trafic de 60 s (à droite) pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Pour le protocole CF, le taux de livraison des paquets de données varie entre 59 % et 68 % lorsque la période de génération de trafic est de 5 s, et entre 60 % et 74 % lorsque la période de génération de trafic est de 60 s. Le taux de livraison des paquets de données ne varie pas beaucoup avec l'augmentation du taux d'activité ou de la charge de trafic. Le taux de livraison reste faible, en raison du mécanisme de mise à jour du *cost-to-go* (représentant le coût pour atteindre le puits), qui est un paramètre crucial dans le protocole CF. En effet, quand un nœud n entend un nœud voisin v ayant un plus faible coût $c(v)$, que n soit le récepteur attendu du paquet ou

non, n met à jour sa valeur $c_n(v)$ en conséquence. Ce qui fait que les anciens voisins de n qui lui servaient de relais pour ses paquets de données vont ignorer les paquets de n par la suite. Ce comportement réduit les performances du protocole si le lien entre n et v est avec perte, ce qui est probable si v est assez loin de n et ne peut l'entendre que rarement.

Pour le protocole ADC-GRAB, le taux de livraison augmente de 13 % à 66 % lorsque la période de trafic est de 5 s, et de 43 % à un peu plus de 99 % lorsque la période de trafic est de 60 s. Le taux de livraison est très faible lorsque la charge de trafic est élevée (pour une période de 5 secondes), car les nœuds ne se rencontrent pas pendant une durée suffisamment longue (vu le faible taux d'activité) pour supporter les coûts dus aux balises et échanger les paquets de données.

Pour E-ADCR, le taux de livraison augmente de 68 % à 84 % lorsque la période de trafic est de 5 s, et de 92 % à 97 % lorsque la période de trafic est de 60 s. E-ADCR montre de bonnes performances pour des taux d'activité faibles : il est bénéfique de réduire la surcharge des paquets de contrôle en envoyant fréquemment les paquets de données, plutôt que d'attendre un voisin qui est plus proche de la destination.

La figure 4.14 montre le délai moyen de livraison des paquets de données en fonction du taux d'activité, avec à gauche une période de génération de trafic de 5 s et à droite une période de 60 s.

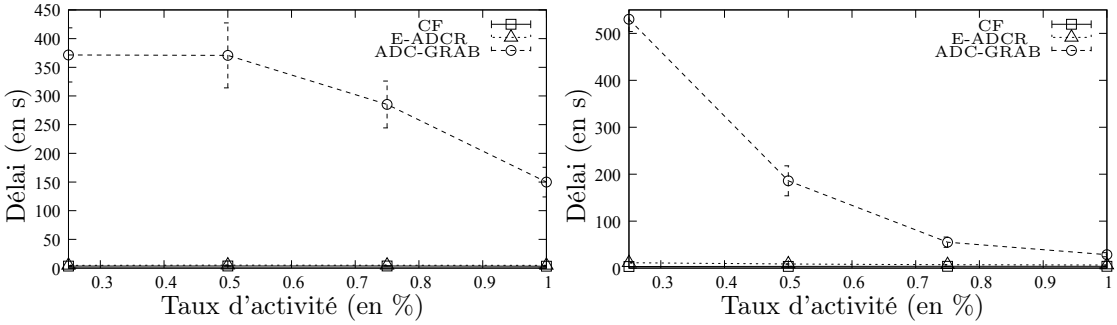


Figure 4.14 – Délai moyen de livraison des données en fonction du taux d'activité, avec à gauche une période de génération de trafic de 5 s et à droite une période de 60 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Pour le protocole CF, le délai moyen de livraison des paquets de données varie entre 3 s et 4 s lorsque le taux d'activité augmente, indépendamment de la charge de trafic. Ceci peut être expliqué par le fait que dans CF les paquets dont le *cost-to-go* est supérieur sont rediffusés après leur réception qu'il y ait ou pas un voisin actif pour le recevoir. Dès lors, les paquets qui ont pu être entendus ne passent pas beaucoup de temps en file d'attente, d'où le faible délai.

Pour le protocole ADC-GRAB, le délai moyen de livraison des paquets de données

diminue rapidement lorsque le taux d'activité augmente, mais reste toujours élevé (il varie de 371 s à 150 s avec une période de génération de trafic de 5 s, et de 530 s à 28 s avec une période de génération de trafic de 60 s). En effet, comme le taux d'activité augmente et le trafic est relativement faible, il y a plus d'occasions de rencontre entre les voisins avec une durée suffisante pour échanger les données. Cela évite que les paquets restent longtemps en file d'attente avant d'être transmis.

Pour E-ADCR, le délai moyen de livraison des paquets de données est faible (il varie entre 4 s et 5 s avec une période de génération de trafic de 5 s, et entre 6 s et 11 s avec une période de génération de trafic de 60 s). Cela montre que l'envoi des paquets à tous les voisins possibles permet au protocole E-ADCR de trouver l'itinéraire le plus rapide pour les paquets (parmi beaucoup d'autres routes empruntés en parallèles). Il est important de noter que le délai est calculé uniquement à partir des paquets reçus (voir la figure 4.13), et ce délai prend en compte le temps de la première réception du paquet par la destination.

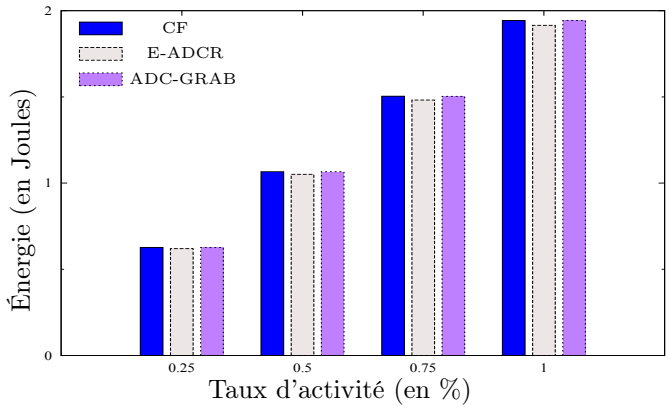


Figure 4.15 – Énergie moyenne consommée par heure par nœud en fonction du taux d'activité, avec une période de génération de trafic de 5 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

La figure 4.15 montre la consommation d'énergie en Joules pendant 1 heure en fonction du taux d'activité. La consommation d'énergie augmente pour tous les protocoles de 0,62 Joules à 1,94 Joules avec le taux d'activité. La consommation d'énergie est pratiquement similaire pour les trois protocoles à cause du fait qu'ils sont tous les trois basés sur le même protocole MAC qui opère avec un taux d'activité fixe pour tous les nœuds. Cela confirme que le taux d'activité est un bon indicateur pour estimer la durée de vie du réseau.

4.3.2 Impact des taux d'activité important

Dans ce deuxième scénario de simulation, nous étudions l'impact des grands taux d'activité sur les performances des trois protocoles de routage. Nous considérons des taux d'activité qui sont supérieurs de 1 %.

La figure 4.16 montre le taux de livraison des paquets de données en fonction du taux d'activité (qui varie entre 1 % et 10 %), avec à gauche une période de génération de trafic de 5 s et à droite une période de 60 s.

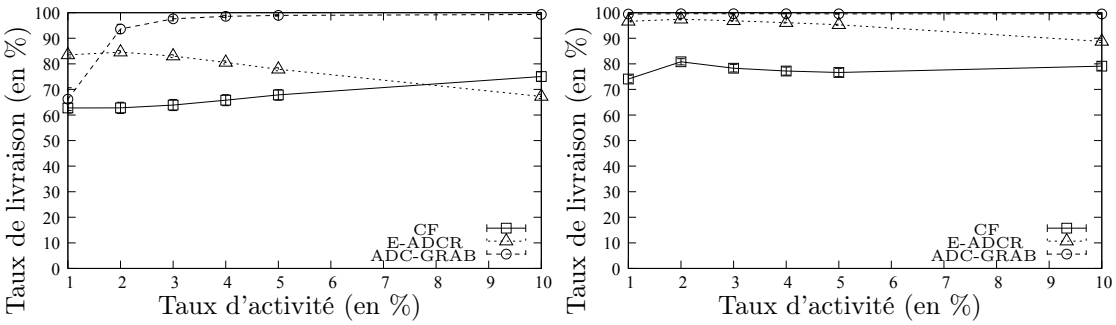


Figure 4.16 – Taux de livraison des paquets de données en fonction du taux d'activité, avec à gauche une période de génération de trafic de 5 s et à droite une période de 60 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Pour le protocole CF, les résultats montrent qu'il n'y a pas de variation significative du taux de livraison qui augmente de 63 % à 75 % avec une période de génération de trafic de 5 s, et de 74 % à 79 % avec une période de génération de trafic de 60 s, même lorsque le taux d'activité est est de 10 %.

Pour le protocole ADC-GRAB, le taux livraison des paquets augmente de manière significative avec le taux d'activité entre 66 % et 97 % avec une période de génération de trafic de 5 s et reste autour de 100 % pour une période de 60 s.

Pour le protocole E-ADCR, le taux de livraison de paquets atteint un maximum avec un taux d'activité de 2 %. Ceci s'explique par le fait que lorsque les nœuds ont une longue durée d'activité qui fait que les activités communes avec les voisins se produisent plus fréquemment. Cela se traduit par une augmentation du nombre de collisions qui réduit le taux de livraison des paquets de données.

La figure 4.17 montre le délai moyen de bout en bout pour la livraison des paquets de données en fonction du taux d'activité, avec à gauche une période de génération de trafic de 5 s et à droite une périodes de 60 s.

Pour le protocole CF, le délai moyen de bout en bout pour la livraison des paquets de données varie entre 2 s et 3 s pour les deux périodes de génération de trafic, lorsque le taux d'activité augmente de 1 % à 10 %.

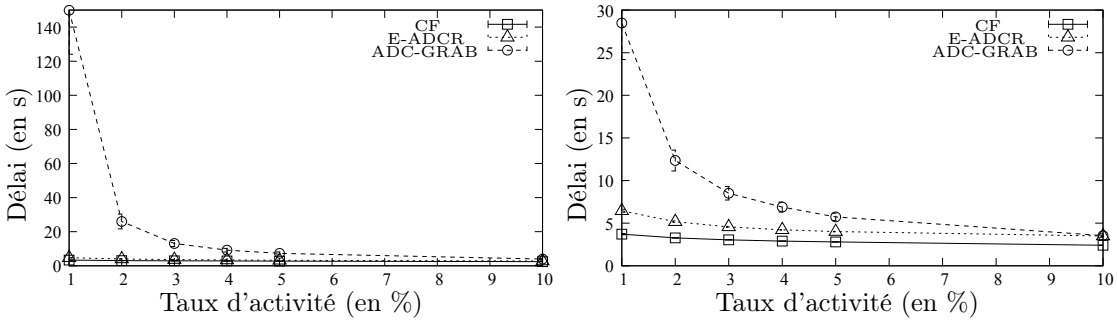


Figure 4.17 – Délai moyen de livraison des paquets de données en fonction du taux d'activité, avec à gauche une période de génération de trafic de 5 s et à droite une périodes de 60 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Pour le protocole ADC-GRAB, le délai moyen de bout en bout pour la livraison des paquets de données diminue de manière significative (de 150 s à 3 s avec une période de génération de trafic de 5 s, et de 28 s à 3 s pour une période de 60 s), lorsque le taux d'activité augmente de 1 % à 10 %. Cette augmentation de performance est due au fait qu'avec un grand taux d'activité, les nœuds se rencontrent plus souvent et pour de grandes durées, et n'ont pas besoin d'attendre longtemps avant de rencontrer un nœud voisin qui est plus près du puits.

Pour le protocole E-ADCR, le délai moyen de bout en bout pour la livraison des paquets de données reste faible (entre 3 s et 6 s pour les deux périodes de génération de trafic), même avec un taux de livraison des paquets de données élevé (voir la figure 4.16).

La figure 4.18 montre la consommation d'énergie en Joules pendant 1 heure en fonction du taux d'activité (qui varie entre 1 % et 10 %). Comme on s'y attend, la

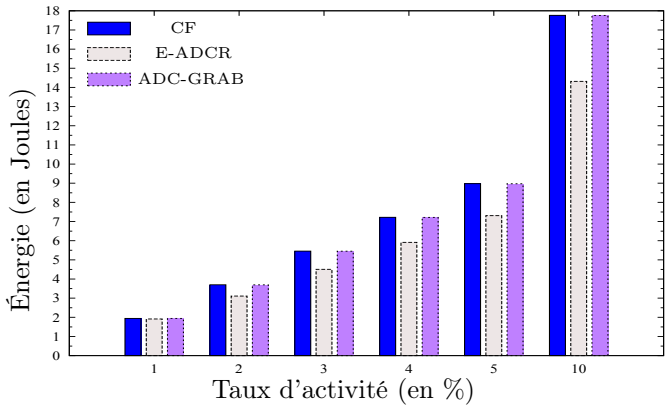


Figure 4.18 – Énergie moyenne consommée par heure par nœud en fonction du taux d'activité, avec une période de génération de trafic de 5 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

consommation d'énergie augmente avec le taux d'activité. La consommation d'énergie

est pratiquement similaire pour les protocoles CF et ADC-GRAB (de 1,94 Joules à 17,76 Joules) et légèrement inférieur pour E-ADCR (de 1,91 Joules à 14,31 Joules). La consommation énergétique du protocole E-ADCR est de plus en plus inférieure à celle de CF et ADC-GRAB quand le taux d'activité augmente, à cause du fait que les nœuds passent la plupart du temps en mode transmission (qui consomme un peu moins d'énergie que le mode réception).

4.3.3 Impact de la densité du réseau

Dans ce troisième scénario de simulation, nous étudions l'impact de la densité du réseau sur la performance des trois protocoles de routage. Nous considérons dans un premier temps des taux d'activité inférieurs à ou égaux à 1 % et dans un second temps des taux d'activité supérieurs à 1 %. Nous faisons varier le nombre de nœuds de 100 à 200 sur la même zone de 170 × 170 avec une distance maximale de 30 m et une distance minimale de 1 m. Les simulations ont été effectuées avec 10 topologies aléatoires de 100 nœuds (avec une densité de 8 dans un rayon de 30 m), 10 topologies aléatoires de 150 nœuds (avec une densité de 12 dans un rayon de 30 m) et 10 topologies aléatoires de 200 nœuds (avec une densité de 22 dans un rayon de 30 m), dont un puits positionné au coin de la zone. Le nombre maximum de sauts dans la topologie de 100 nœuds est de 7, et est de 8 pour les deux autres topologies. 30 % des nœuds aléatoirement localisés dans le réseau effectuent des mesures périodiques (de période 5 secondes ou de 60 secondes) pour les transmettre de saut en saut au puits.

4.3.3.1 Cas des taux d'activité $\leq$ 1 %

Dans cette partie nous évaluons les performances des trois protocoles de routage E-ADCR, ADC-GRAB et CF avec un taux d'activité de 0.5 % et de 1 %, dans le cas d'un trafic relativement élevé (avec une période de génération de trafic de 5 s) et dans le cas d'un trafic relativement faible (avec une période de génération de trafic de 60 s).

Cas d'une période de génération de trafic de 5 s

Les figures 4.19 et 4.20 montrent respectivement le taux de livraison des paquets de données et le délai moyen de bout en bout en fonction du nombre de nœuds (qui varie entre 100 et 200), avec à gauche un taux d'activité de 0.5 % et à droite un taux d'activité 1 % pour les trois protocoles lorsque la période de génération de trafic est de 5 s.

Pour le protocole CF, le taux de livraison diminue de 64 % à 61 % pour un taux d'activité de 0.5 %, et de 67 % à 57 % pour un taux d'activité de 1 %. Le délai moyen de bout en bout pour la livraison des paquets de données est environ 4 s avec un taux d'activité de 0.5 % et reste autour de 3 s avec un taux d'activité de 1 % indépendamment de la densité du réseau. Le taux de livraison de CF diminue lorsque le nombre de nœuds augmente car le nombre moyen de voisins pour chaque nœud augmente. Cela augmente le nombre de copies des paquets de données.

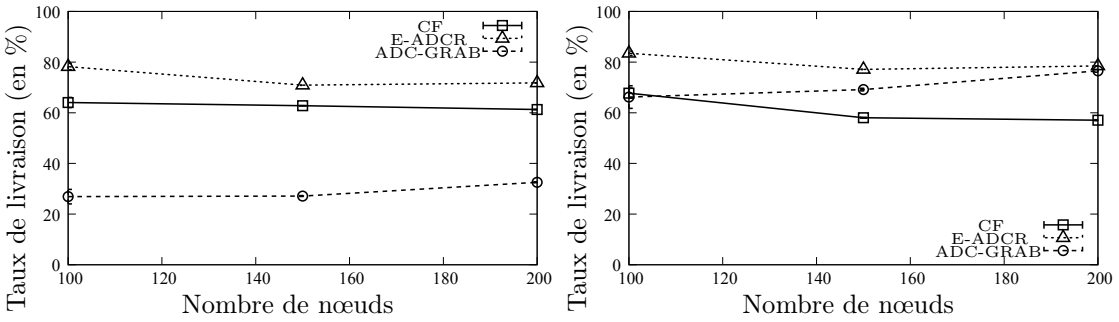


Figure 4.19 – Taux de livraison des paquets de données en fonction du nombre de nœuds, avec un taux d'activité de 0.5 % (à gauche) et un taux d'activité de 1 % (à droite) lorsque la période de génération de trafic est de 5 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Pour E-ADCR, le taux de livraison diminue de 78 % à 71 % pour un taux d'activité de 0.5 %, et de 83 % à 78 % avec un taux d'activité de 1 %. Le délai moyen de bout en bout pour la livraison des paquets de données reste autour de 5 s pour un taux d'activité de 0.5 % et augmente légèrement d'environ 5 s à 6 s pour un taux d'activité de 1 % lorsque la densité du réseau augmente. Le fait que le taux de livraison dans E-ADCR diminue lorsque le nombre de nœuds dans le réseau augmente, s'explique par le remplissage rapide des files d'attente causé par le nombre élevé de voisins et la charge de trafic importante. Cela fait qu'un paquet passe moins de temps dans la file d'attente et a donc moins de possibilités d'être rediffusé plusieurs fois pour avoir plus de chance d'être entendu par un voisin plus proche du puits (car lorsque la file d'attente est pleine, et qu'il y a un nouveau paquet, le paquet en tête de file est supprimé puis, les autres paquets sont décalés de sorte à libérer la dernière place de la file pour le nouveau paquet). Aussi, il y a une forte probabilité que des voisins de même distance au puits rediffusent plusieurs fois un même paquet et qu'il finissent par être supprimé (vu qu'à chaque rediffusion le paramètre TTL est décrémenté et lorsqu'il atteint 0 le paquet est supprimé).

Pour le protocole ADC-GRAB, le taux livraison des paquets augmente lorsque la densité du réseau augmente, mais reste globalement faible (environ 27 % à 32 % pour un taux d'activité de 0.5 %, et de 66 % à 76 % pour un taux d'activité de 1 %).

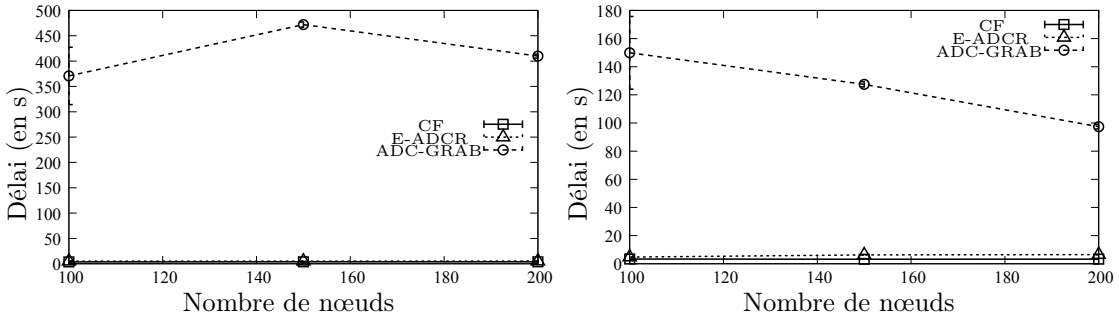


Figure 4.20 – Délai moyen de livraison des paquets de données en fonction du nombre de nœuds, avec un taux d'activité de 0.5 % (à gauche) et un taux d'activité de 1 % (à droite) lorsque la période de génération de trafic est de 5 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Le délai moyen de bout en bout pour la livraison des paquets de données augmente avec la densité du réseau de 370 s à 409 s pour un taux d'activité de 0.5 % et diminue significativement (de 149 s à 97 s) pour un taux d'activité de 1 %, lorsque la densité du réseau augmente. L'augmentation du délai lorsque le taux d'activité est égal à 0.5 % est due à la légère augmentation du taux de livraison qui est très faible lorsque la charge de trafic est élevée et que le taux d'activité est faible. Le taux de livraison dans ADC-GRAB augmente de façon significative et le délai diminue lorsque le taux d'activité est de 1 % car la durée de l'activité est un peu plus longue. De plus, chaque nœud a plus de possibilités de rencontrer un nœud voisin qui est plus près du puits. Le taux de livraison est faible et est en dessous de celui de CF et de E-ADCR car les rencontres sont plus fréquentes mais ne sont pas suffisamment longues (vu le faible taux d'activité) pour supporter les coûts dus aux balises et à la charge de trafic élevée.

Cas d'une période de génération de trafic de 60 s

Les figures 4.21 et 4.22 montrent respectivement le taux de livraison des paquets de données et le délai moyen de bout en bout en fonction du nombre de nœuds, avec à gauche un taux d'activité de 0.5 % et à droite un taux d'activité 1 % pour les trois protocoles lorsque la période de génération de trafic est de 60 s.

Pour le protocole CF, le taux de livraison augmente de 64 % à 69 % avec un taux d'activité de 0.5 %, et diminue de 74 % à 68 % avec un taux d'activité de 1 %. Le délai moyen de bout en bout pour la livraison des paquets de données augmente de 3 s à 4 s avec un taux d'activité de 0.5 % et reste autour de 4 s avec un taux d'activité de 1 % lorsque la densité du réseau augmente. Le taux de livraison des paquets et le délai moyen de bout en bout dans CF augmentent légèrement avec la densité du réseau lorsque le taux d'activité est de 0.5 %. Cela s'explique par le fait que lorsque

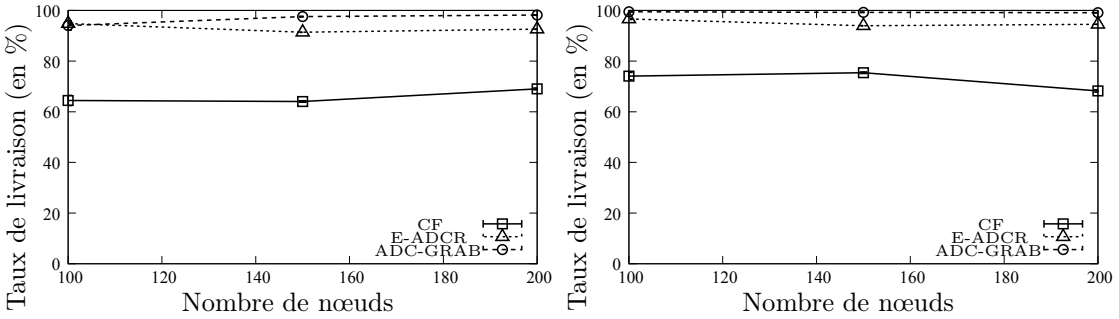


Figure 4.21 – Taux de livraison des paquets de données en fonction du nombre de nœuds, avec un taux d'activité de 0.5 % (à gauche) et un taux d'activité de 1 % (à droite) lorsque la période de génération de trafic est de 60 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

le taux d'activité est faible et la charge de trafic est moins importante, il y a une faible contention même si le nombre de voisins augmente.

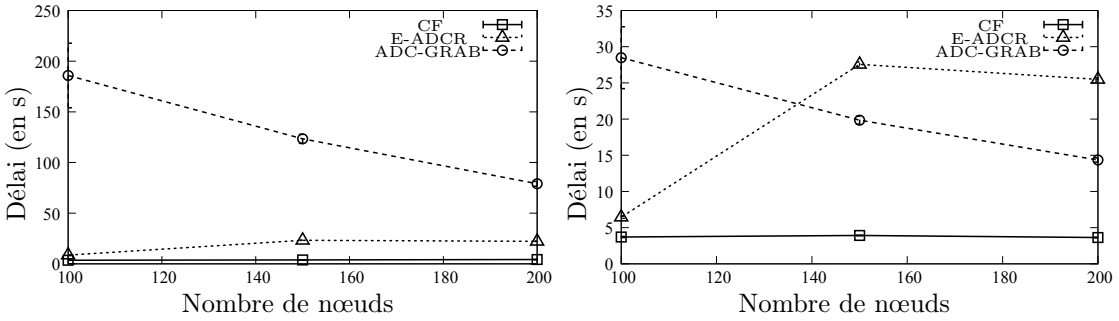


Figure 4.22 – Délai moyen de livraison des paquets de données en fonction du nombre de nœuds, avec un taux d'activité de 0.5 % (à gauche) et un taux d'activité de 1 % (à droite) lorsque la période de génération de trafic est de 60 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Pour E-ADCR, le taux de livraison diminue d'environ 95 % à 91 % avec un taux d'activité de 0.5 % et d'environ 97 % à 94 % avec un taux d'activité de 1 % lorsque la densité du réseau augmente. Le délai moyen de bout en bout pour la livraison des paquets de données augmente de 8 s à 23 s avec un taux d'activité de 0.5 % et augmente d'environ 6 s à 27 s avec un taux d'activité de 1 % lorsque la densité du réseau augmente. Le taux de livraison dans E-ADCR diminue et le délai augmente lorsque la densité du réseau augmente à cause du fait que plusieurs nœuds dans un même voisinage peuvent être actifs en même temps. Cela génère un nombre important de copies, une augmentation des retransmissions dues aux collisions, et une suppression des paquets avant même qu'une copie atteigne la destination finale (vu que le TTL des paquets est décrémenté à chaque rediffusion).

Pour le protocole ADC-GRAB, le taux de livraison des paquets augmente de 94 %

à 98 % avec la densité du réseau avec un taux d'activité de 0.5 % et reste égale à un peu plus de 99 % avec un taux d'activité de 1 %. Le délai moyen de bout en bout pour la livraison des paquets de données diminue significativement de 186 s à 79 s lorsque la densité du réseau augmente avec un taux d'activité de 0.5 % et de 28 s à 14 s avec un taux d'activité de 1 %. Le taux de livraison dans ADC-GRAB augmente et le délai diminue de façon très significative lorsque la densité du réseau augmente car non seulement les rencontres sont plus fréquentes quand il y a plusieurs voisins plus proches du puits, mais aussi la charge de trafic est moins importante et peut être supportée par les faibles taux d'activité.

4.3.3.2 Cas des taux d'activité $>$ 1 %

Dans cette deuxième partie, nous évaluons les performances des trois protocoles de routage E-ADCR, ADC-GRAB et CF lorsque le nombre de nœuds augmente. Nous considérons les taux d'activité de 2 % et de 10 %, avec une période de génération de trafic de 5 s et une période de génération de trafic de 60 s.

Cas d'une période de génération de trafic de 5 s

Les figures 4.23 et 4.24 montrent respectivement le taux de livraison des paquets de données et le délai moyen de bout en bout en fonction du nombre de nœuds, avec à gauche un taux d'activité de 2 % et à droite un taux d'activité de 10 % pour les trois protocoles lorsque la période de génération de trafic est de 5 s.

Pour le protocole CF, le taux de livraison diminue légèrement lorsque le nombre de nœuds dans le réseau augmente avec les deux taux d'activité (2 % et 10 %). Le délai moyen de bout en bout pour la livraison des paquets de données est autour de 3 s avec un taux d'activité de 2 % et autour de 2 s avec un taux d'activité de 10 %, indépendamment de la densité du réseau.

Pour E-ADCR, le taux de livraison diminue légèrement de 84 % à 82 % avec un taux d'activité de 2 % et augmente de 67 % à 85 % avec un taux d'activité de 10 % lorsque la densité du réseau augmente. Le délai moyen de bout en bout pour la livraison des paquets de données varie entre 4 s et 8 s avec un taux d'activité de 2 % et entre 3 s et 10 s avec un taux d'activité de 10 % lorsque la densité du réseau augmente. E-ADCR fournit de faibles performances pour des grands taux d'activité indépendamment de la densité du réseau. Ceci s'explique par le fait que lorsque les nœuds ont une longue durée d'activité il y a un débordement rapide des files d'attentes dû au nombre important de copies des paquets de données. Il y a aussi une augmentation du nombre de collisions qui fait que les paquets retransmis

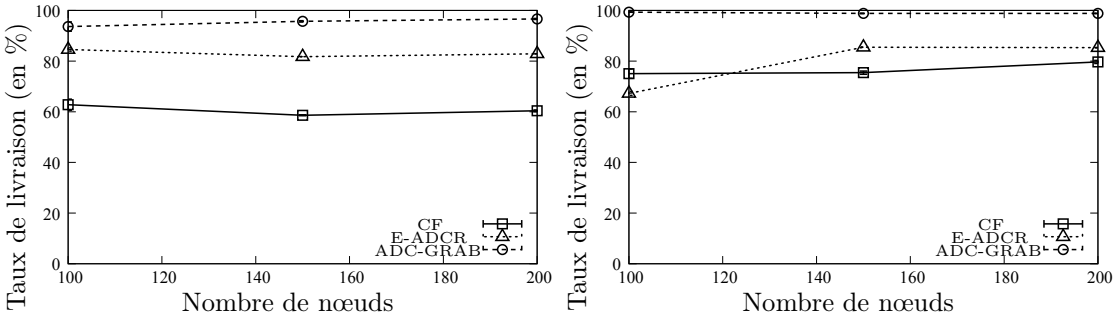


Figure 4.23 – Taux de livraison des données en fonction du nombre de nœuds, avec un taux d'activité de 2 % (à gauche) et avec un taux d'activité de 10 % (à droite) lorsque la période de génération de trafic est de 5 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

plusieurs fois sont supprimés après quatre tentatives. Cela augmente la latence et réduit le taux de livraison des paquets.

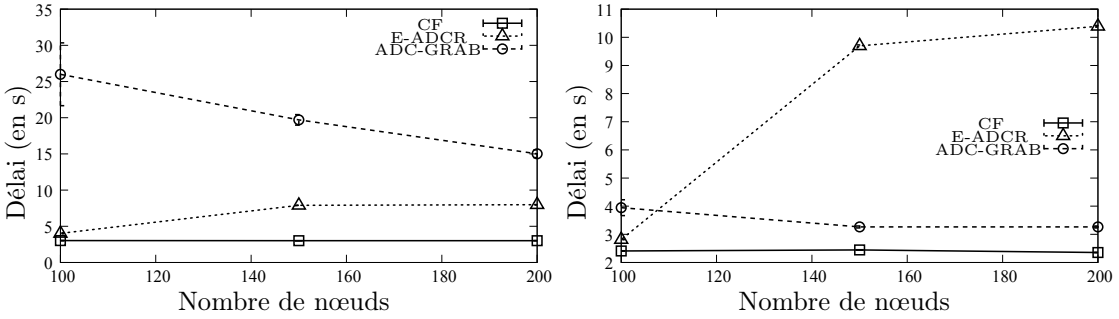


Figure 4.24 – Délai moyen de livraison des paquets de données en fonction du nombre de nœuds, avec un taux d'activité de 2 % (à gauche) et avec un taux d'activité de 10 % (à droite) lorsque la période de génération de trafic est de 60 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Pour le protocole ADC-GRAB, le taux de livraison des paquets augmente significativement lorsque la densité du réseau augmente d'environ 94 % à environ 97 % avec un taux d'activité de 2 %, et reste autour de 99 % avec un taux d'activité de 10 %. Le délai moyen de bout en bout pour la livraison des paquets de données diminue de 25 s à 15 s pour un taux d'activité de 2 % et diminue de 4 s à 3 s pour un taux d'activité de 10 % lorsque la densité du réseau augmente. On remarque que la densité du réseau a plus d'impact sur les performances de ADC-GRAB quand le taux d'activité est de 2 % que lorsqu'il est de 10 %. Cela s'explique par le fait que, lorsque le taux d'activité est grand, la durée d'activité est longue et la probabilité qu'un nombre important de nœuds soient tous actifs en même temps devient importante (vu la densité du réseau qui augmente). Cela augmente la contention et les collisions, donc réduit le taux de livraison des paquets.

Cas d'une période de génération de trafic de 60 s

Les figures 4.23 et 4.24 montrent respectivement le taux de livraison des paquets de données et le délai moyen de bout en bout en fonction du nombre de nœuds, avec à gauche un taux d'activité de 2 % et à droite un taux d'activité 10 % pour les trois protocoles lorsque la période de génération de trafic est de 60 s.

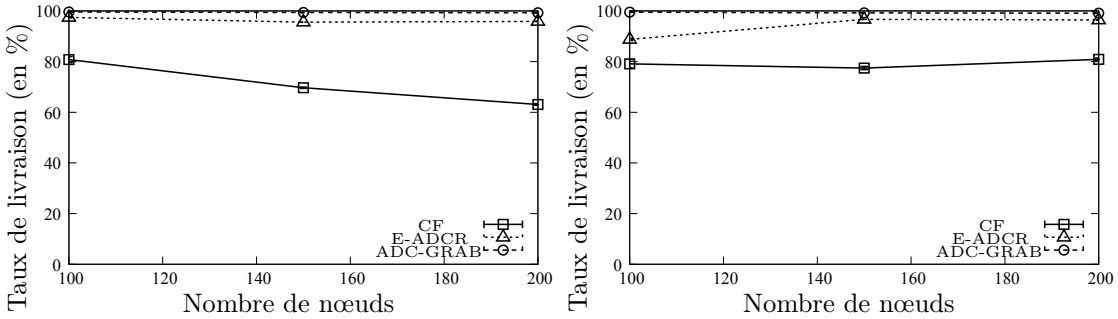


Figure 4.25 – Taux de livraison des paquets de données en fonction du nombre de nœuds, avec un taux d'activité de 2 % (à gauche) et avec un taux d'activité de 10 % (à droite) lorsque la période de génération de trafic est de 60 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

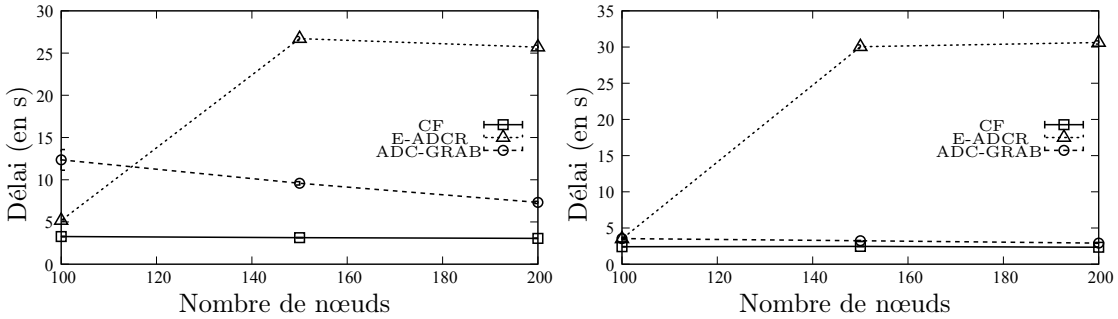


Figure 4.26 – Délai moyen de livraison des paquets de données en fonction du nombre de nœuds, avec un taux d'activité de 2 % (à gauche) et un taux d'activité de 10 % (à droite) lorsque la période de génération de trafic est de 60 s, pour les protocoles de routage CF, E-ADCR et ADC-GRAB.

Pour le protocole CF, le taux de livraison diminue d'environ 81 % à 63 % avec un taux d'activité de 2 % et varie légèrement entre 79 % et 81 % avec un taux d'activité de 10 % lorsque la densité du réseau augmente. Le délai moyen de bout en bout pour la livraison des paquets de données est autour de 3 s avec un taux d'activité de 2 % et autour de 2 s avec un taux d'activité de 10 % lorsque la densité du réseau augmente.

Pour E-ADCR, le taux de livraison diminue de 97 % à 95 % avec un taux d'activité de 2 % et augmente de 89 % à 96 % avec un taux d'activité de 10 % lorsque la densité du réseau augmente. Le délai moyen de bout en bout pour la livraison des paquets de

données augmente de 5 s à 26 s avec un taux d'activité de 2 % et augmente d'environ 3 s à 30 s avec un taux d'activité de 10 % lorsque la densité du réseau augmente. Le taux de livraison dans E-ADCR augmente lorsque la densité du réseau augmente et que le taux d'activité est de 10 % car même si la contention est forte et que le nombre de collisions est important, la charge de trafic est faible et peut être supportée malgré les collisions. Néanmoins, les paquets vont mettre plus de temps dans les files d'attente (à cause du nombre important de retransmissions) pour atteindre la destination finale.

Pour le protocole ADC-GRAB, le taux livraison des paquets diminue légèrement avec la densité du réseau indépendamment du taux d'activité. Le délai moyen de bout en bout pour la livraison des paquets de données diminue de 12 s à 7 s avec un taux d'activité de 2 % et reste autour de 3 s avec un taux d'activité de 10 %. Le taux de livraison des paquets dans ADC-GRAB baisse légèrement quand le trafic est moins important et que le taux d'activité est grand car les périodes de rencontres sont longues et très fréquentes (vu le nombre important de voisins) ce qui augmente la contention, les collisions et les retransmissions.

4.3.4 Bilan sur les résultats de simulation des protocoles de routage

En résumé nous pouvons noter que les performances du protocole CF en termes de taux de livraison des paquets restent en dessous des performances des deux protocoles E-ADCR et ADC-GRAB. Lorsque le taux d'activité est supérieur à 1 %, le protocole ADC-GRAB montre de meilleures performances que le protocole E-ADCR. En effet, lorsque la densité du réseau augmente ou que la période d'activité des nœuds augmente, E-ADCR perd sa spécificité de tirer profit des activités communes rares et courtes pour échanger efficacement les paquets de données. Contrairement à E-ADCR, pour le protocole ADC-GRAB, augmenter la densité du réseau ou le taux d'activité revient à des activités communes plus fréquentes et longues. Il est néanmoins intéressant de noter que lorsque le taux d'activité est inférieur à 1 %, les performances du protocole ADC-GRAB sont beaucoup plus faibles que les performances du protocole E-ADCR. Le protocole E-ADCR relève le défi de fonctionner efficacement avec des taux d'activité faibles. Par exemple, pour un réseau avec un degré moyen de 8 nœuds, E-ADCR garantit un taux de livraison de 92 % et un délai moyen de bout en bout d'environ 11 s (dans nos simulations) même lorsque le taux d'activité est fixé à 0,25 % pour tous les nœuds et une période de génération du trafic est de 1 paquet par minute. Le protocole E-ADCR est donc adapté à la phase

d'attente des applications de surveillance (telles que les volcans ou les forêts) pendant laquelle les nœuds font des mesures régulières avec une faible fréquence et doivent être très économes en énergie. De même, le protocole ADC-GRAB est beaucoup plus adapté à la phase de travail intensif (par exemple en cas de détection d'événement critique) pendant laquelle le trafic devient important et donc qui nécessite que le taux d'activité augmente.

4.4 Expérimentation

Cette partie présente la validation de nos résultats par maquettage sur une plate-forme matérielle. Nous avons effectué des tests sur des nœuds TelosB de Crossbow en utilisant le système d'exploitation événementiel TinyOS [HSW+00b]. Nous validons les fonctionnalités de nos deux protocoles de routage (E-ADCR et ADC-GRAB) dans un environnement réel sur des nœuds capteurs. Dans la suite, nous décrivons l'environnement d'expérimentation et les résultats obtenus.

4.4.1 Environnement d'expérimentation

Dans cette sous-partie, nous décrivons le système d'exploitation événementiel TinyOS [HSW+00b], notre plate-forme d'expérimentation et la topologie de la maquette avec les différents paramètres utilisés.

4.4.1.1 Description de TinyOS

TinyOS est un système d'exploitation à code source libre principalement développé par l'université américaine de Berkeley pour les RCSF. TinyOS est programmé en langage NesC [GLR+03] (un dérivé du langage C) qui lui offre une architecture à base de composants pour un fonctionnement avec des ressources limitées. L'ordonnanceur dans TinyOS a deux niveaux de priorité. La haute priorité est pour les événements et les commandes et la priorité basse pour les tâches. Un événement est l'équivalent logiciel d'une interruption matérielle (comme la pression d'un bouton ou la réception d'un message), et une commande est l'exécution d'une fonctionnalité précise dans un autre composant (par exemple **AMControl.start()** pour démarrer la radio). Un événement est prioritaire par rapport à une tâche et peut interrompre la tâche en cours d'exécution. Les tâches sont enregistrées dans une file d'attente de tâches pour être exécutées sur la base d'un ordonnancement en FIFO (pour *First In First Out*). Cela signifie qu'une tâche est exécutée d'une manière non-préemptive

(une nouvelle tâche ne peut pas être exécutée avant la fin de la tâche en cours). Il n'y a donc pas de priorités parmi les tâches.

Il existe différentes plates-formes disponibles dans TinyOS (par exemple : TelosA, TelosB, Mica, MicaZ, Mica2, Iris, etc).

4.4.1.2 Description de la plate-forme d'expérimentation

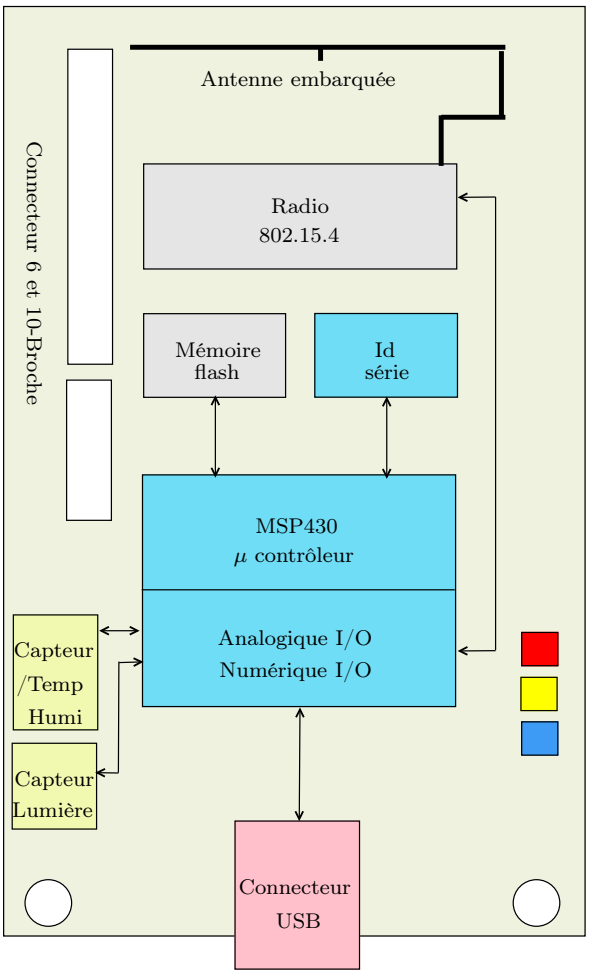


Figure 4.27 – Schéma fonctionnel d'un nœud capteur de type TelosB.

Nous avons utilisé une plate-forme TelosB (de type Crossbow [Teca] et MEMSIC [Tecb]) dans nos expérimentations. La figure 4.27 montre le schéma fonctionnel d'un nœud capteur de type TelosB. Cette plate-forme repose sur la norme IEEE 802.15.4 pour les communications radio dans la bande de fréquences de 2.4 à 2.4835 GHz. Le débit radio maximal est de 250 kbps, le nœud intègre un micro-contrôleur TI MSP430, une RAM de 10 kB, un composant radio CC2420 et un module antenne *Onboard*. La programmation et la récupération des informations (telles que la température, l'humidité, la lumière) se font *via* l'interface USB.

4.4.1.3 Topologie et paramètres de maquettage

La figure 4.28 montre la topologie d'expérimentation. Notre évaluation expéri-

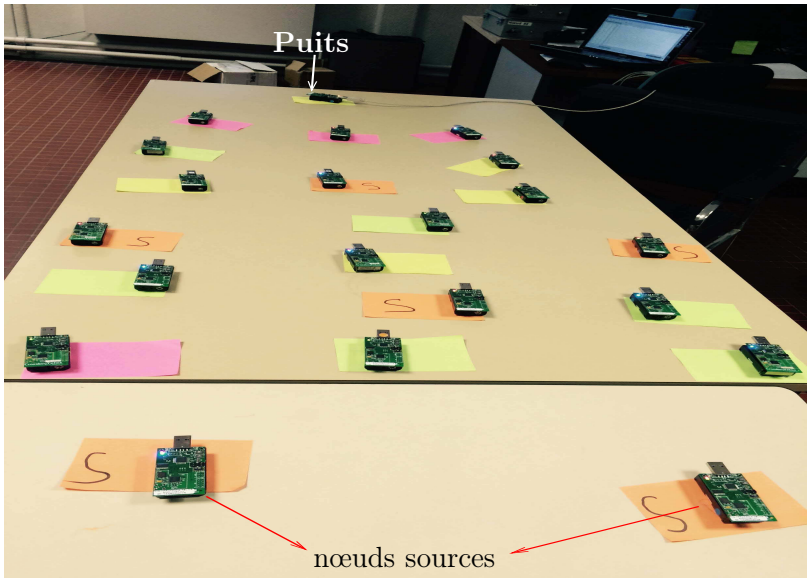


Figure 4.28 – Topologie d'expérimentation.

mentale se compose d'un ensemble de 20 telosB et d'un puits sur une table dans notre laboratoire. Afin de limiter la portée de communication, nous avons choisi une puissance de transmission de 1 pour les nœuds sur une grille de 0 à 31 (cela correspond à une puissance de transmission de -25 dBm qui représente la puissance minimale). Cela forme un réseau avec un maximum de quatre sauts du nœud puits. Nous avons au total 6 nœuds sources de données : 2 nœuds sont positionnés à 4 sauts du puits, 3 nœuds sont positionnés à 3 sauts du puits, et un nœud est positionné à 2 sauts du puits. Les six nœuds effectuent périodiquement des mesures et les acheminent *via* les autres nœuds vers le puits. Bien que nous ayons un réseau expérimental de taille plus petite que les simulations, les résultats de cette expérience permettent d'avoir une indication claire des performances dans un réseau réel et la preuve que nos protocoles fonctionnent sur des nœuds capteurs réels. Notre protocole expérimentale est détaillée dans la partie A. Nous avons réalisé l'expérimentation en faisant varier le taux d'activité de 0.25 % à 10 % avec des périodes de génération de trafic de 5 s et de 60 s.

4.4.2 Résultats d'expérimentation

4.4.2.1 Cas des taux d'activité faibles

La figure 4.29 montre le taux de livraison des paquets de données en fonction du taux d'activité (qui varie entre 0.25 % et 1 %) avec (à gauche) une période de génération de trafic de 5 s (ce qui correspond à une génération de trafic élevée compte tenu du taux d'activité faible) et (à droite) une période de génération de trafic de 60 s (ce qui correspond à une génération de trafic relativement faible) pour les protocoles E-ADCR et ADC-GRAB.

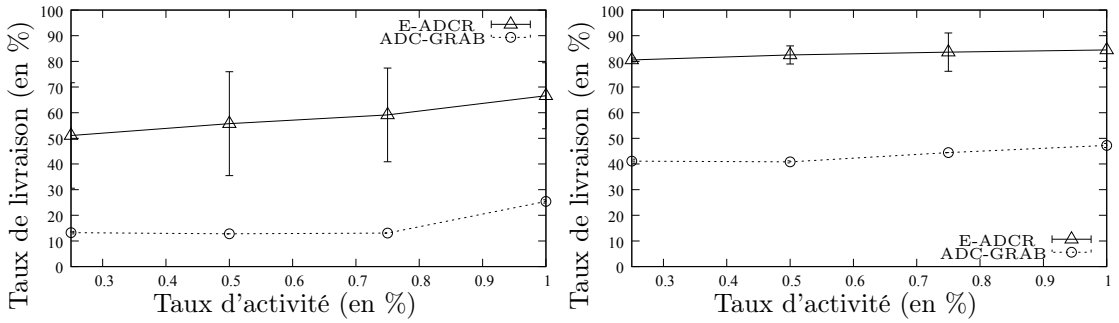


Figure 4.29 – Taux de livraison des paquets de données en fonction du taux d'activité avec (à gauche) une période de génération de trafic de 5 s et (à droite) une période de génération de trafic de 60 s, pour les protocoles de routage E-ADCR et ADC-GRAB.

Pour le protocole ADC-GRAB, le taux de livraison augmente de 13 % à 25 % lorsque la période de trafic est de 5 s, et de 41 % à 46 % lorsque la période de trafic est de 60 s. Le taux de livraison est très faible lorsque la charge de trafic est élevée (pour une période de 5 secondes), car la durée des rencontres entre les nœuds n'est pas suffisamment longue (vu le faible taux d'activité) pour supporter les coûts dus aux balises et échanger les paquets de données.

Pour E-ADCR, le taux de livraison augmente de 51 % à environ 67 % lorsque la période de trafic est de 5 s, et d'environ 81 % à 84 % lorsque la période de trafic est de 60 s. E-ADCR montre des performances élevées pour des taux d'activité faibles par rapport à ADC-GRAB. Il est bénéfique que les nœuds diffusent fréquemment leurs paquets de données durant leur courte période d'activité, plutôt que d'échanger des paquets de contrôle ou d'attendre un voisin qui est plus proche de la destination finale.

La figure 4.30 montre le délai moyen de bout en bout pour la livraison des paquets de données en fonction du taux d'activité avec (à gauche) une période de génération de trafic de 5 s et (à droite) une période de génération de trafic de 60 s, pour les protocoles E-ADCR et ADC-GRAB.

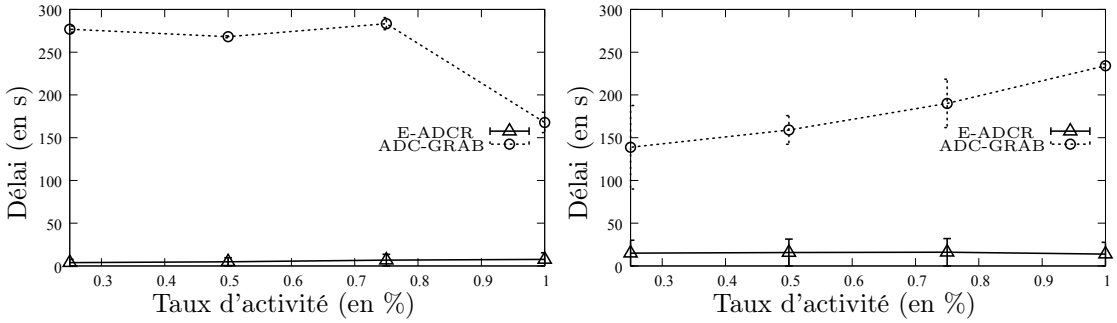


Figure 4.30 – Délai moyen de livraison des paquets de données en fonction du taux d'activité avec (à gauche) une période de génération de trafic de 5 s et (à droite) une période de génération de trafic de 60 s, pour les protocoles de routage E-ADCR et ADC-GRAB.

Pour le protocole ADC-GRAB, le délai moyen de bout en bout pour la livraison des paquets de données varie entre 167 s et 283 s lorsque le trafic est élevé (période de 5 s) et augmente de de 138 s à 234 s lorsque le trafic est relativement faible (période de 60 s). La grande variation du délai pour les périodes de génération de 5 s est due à la perte importante des paquets de données (cf figure 4.29 à gauche) et notamment de ceux qui devraient avoir un délai plus long, ce qui réduit artificiellement le délai.

Pour E-ADCR, le délai moyen de bout en bout pour la livraison des paquets de données est faible (il varie entre 4 s et environ 8 s avec une période de génération de trafic de 5 s, et entre 14 s et 16 s avec une période de génération de trafic de 60 s). Cela montre que l'envoi des paquets à tous les voisins possibles permet au protocole E-ADCR de trouver l'itinéraire le plus rapide pour les paquets (parmi beaucoup d'autres routes empruntées en parallèle).

Cas des taux d'activité important

La figure 4.31 montre le taux de livraison des paquets de données en fonction du taux d'activité (qui varie entre 1 % et 10 %) avec (à gauche) une période de génération de trafic de 5 s et (à droite) une période de génération de trafic de 60 s, pour les protocoles E-ADCR et ADC-GRAB.

Pour le protocole ADC-GRAB, le taux de livraison des paquets augmente de manière significative avec le taux d'activité de 25 % à 99 % avec une période de génération de trafic de 5 s et de 41 % à environ 100 % pour une période de 60 s.

Pour le protocole E-ADCR, le taux de livraison de paquets augmente entre environ 67 % et 94 % avec une période de génération de trafic de 5 s. Lorsque la période de génération de trafic est de 60 s, le taux de livraison augmente de 84 % (pour un taux d'activité de 1 %) à un maximum de 94 % (pour un taux d'activité de 5 %)

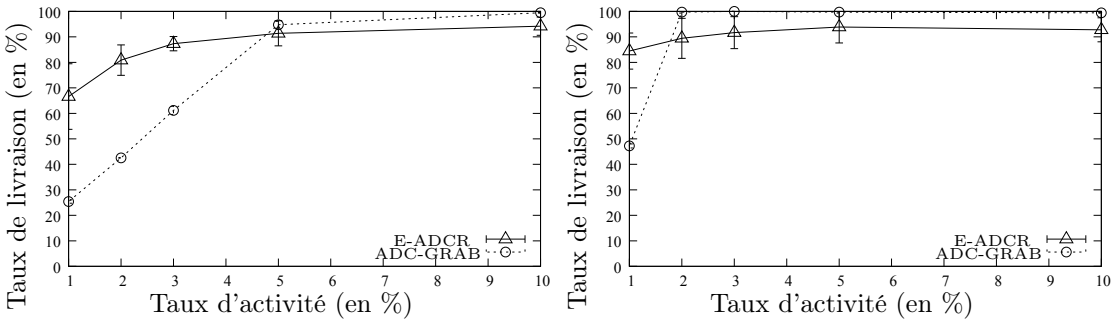


Figure 4.31 – Taux de livraison des paquets de données en fonction du taux d'activité avec (à gauche) une période de génération de trafic de 5 s et (à droite) une période de génération de trafic de 60 s, pour les protocoles de routage E-ADCR et ADC-GRAB.

puis diminue de 94 % à un peu plus de 92 % (pour un taux d'activité de 10 %). Ceci s'explique par le fait que lorsque les nœuds ont une longue durée d'activité, les activités communes avec les voisins se produisent plus fréquemment. Cela se traduit par un nombre important de copies des paquets de données et une augmentation du nombre de collisions, ce qui réduit le taux de livraison des paquets de données.

La figure 4.32 montre le délai moyen de bout en bout pour la livraison des paquets de données en fonction du taux d'activité (qui varie de 1 % à 10 %) avec (à gauche) une période de génération de trafic de 5 s et (à droite) une période de génération de trafic de 60 s, pour les protocoles E-ADCR et ADC-GRAB.

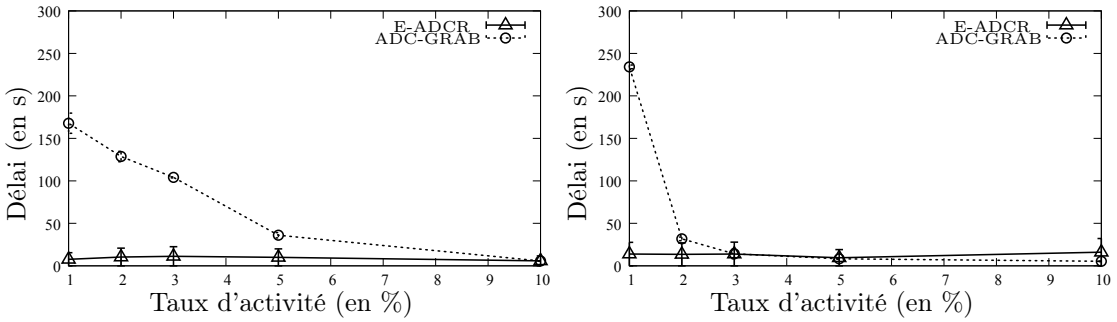


Figure 4.32 – Délai moyen de livraison des paquets de données en fonction du taux d'activité avec (à gauche) une période de génération de trafic de 5 s et (à droite) une période de génération de trafic de 60 s, pour les protocoles de routage E-ADCR et ADC-GRAB.

Pour le protocole ADC-GRAB, le délai moyen de bout en bout pour la livraison des paquets de données diminue de manière significative (de 167 s à environ 6 s avec une période de génération de trafic de 5 s, et de 234 s à 5 s pour une période de 60 s) lorsque le taux d'activité augmente de 1 % à 10 %. Cette augmentation de performance est due au fait qu'avec un grand taux d'activité, les nœuds se rencontrent

plus souvent et pour de grandes durées, et n'ont pas besoin d'attendre longtemps avant de rencontrer un nœud voisin qui est plus près du puits.

Pour le protocole E-ADCR, le délai moyen de bout en bout pour la livraison des paquets de données varie entre environ 6 s et 11 s avec une période de génération de trafic de 5 s, et entre 9 s et 16 s avec une période de génération de trafic de 60 s.

4.4.3 Bilan sur les résultats d'expérimentation

En somme, on remarque expérimentalement qu'avec un taux d'activité faible (inférieur à 1 %) E-ADCR montre de meilleures performances que le protocole ADC-GRAB, tandis qu'avec un taux d'activité important (supérieur à 1 %), ADC-GRAB montre de meilleures performances que E-ADCR. En effet, lorsque la période d'activité des nœuds augmente, E-ADCR perd sa spécificité de tirer profit des activités communes rares et courtes pour échanger efficacement les paquets de données et provoque une surcharge importante du réseau et donc une perte importante des données, tandis que, pour le protocole ADC-GRAB, l'augmentation du taux d'activité revient à des activités communes plus fréquentes et longues.

Il est important de souligner que les conclusions des résultats d'expérimentation concordent avec celles obtenues par simulation : le protocole E-ADCR est le plus efficace pour des taux d'activité inférieurs à 1 %, et le protocole ADC-GRAB est le plus efficace pour des taux d'activité supérieurs à 1 %. Des tests expérimentaux supplémentaires réalisés montrent que lorsque la période de génération de trafic est de 5 min (300 s), E-ADCR atteint un taux de livraison de plus de 97 % avec un taux d'activité de 0,25 % pour tous les nœuds.

Chapitre 5

Conclusion

Les applications de surveillance environnementale sont généralement mises en place pour opérer pendant une longue durée. Ces applications sont réalisées sur des zones étendues, peu accessibles et sans infrastructure préexistante pour alimenter électriquement les composants chargés d'effectuer la surveillance. Les réseaux de capteurs sans fil sont une solution prometteuse pour la mise en place de telles applications. Cependant, un RCSF doit faire face à des contraintes liées aux ressources limitées des nœuds capteurs.

Dans cette thèse, nous nous sommes concentrés spécifiquement sur la gestion efficace de l'énergie pour assurer une longue durée de vie au réseau, et sur la possibilité de passage à l'échelle. Nous nous sommes servis du mécanisme qui séquence les périodes d'activité (où le module radio est allumé) et de sommeil (où le module radio est éteint) des nœuds capteurs suivant un certain taux d'activité, pour leur permettre d'économiser de l'énergie. Nous avons proposé des protocoles MAC asynchrones qui assurent de bonnes performances même quand le taux d'activité est moins de 1 % pour tous les nœuds.

Dans ce qui suit, nous faisons un résumé de nos différentes contributions, et nous donnons les perspectives pour les travaux futurs.

5.1 Résumé des contributions

Nous avons utilisé le protocole MAC de la norme IEEE 802.15.4 pour déterminer la meilleure approche de planification des périodes d'activité et de sommeil des nœuds capteurs. Le mode avec balise de cette norme est un bon candidat pour les RCSF, car les nœuds commencent ensemble leur période d'activité et passent

ensemble en période d'inactivité pour économiser leur énergie, ce qui permet d'équilibrer la consommation énergétique des nœuds et de contribuer à l'estimation de la durée de vie du réseau.

Cependant, la synchronisation nécessaire dans ce mode est difficile à réaliser et coûteuse (en termes d'énergie ou d'utilisation de la bande passante déjà limitée), notamment quand il y a un nombre important de nœuds dans le réseau. Ainsi, nous avons proposé de supprimer l'exigence de la synchronisation, en permettant aux nœuds de démarrer leur période d'activité de façon aléatoire dans chaque cycle global. Nous avons montré que les nœuds peuvent réaliser des rencontres aveugles avec une bonne probabilité et nous avons déterminé le temps d'attente moyen avant que ces rencontres aient lieu. Même lorsque le taux d'activité est faible (autour de 5 %), la durée avant une rencontre peut être maintenue faible (la moyenne est toujours au plus deux fois la durée d'un cycle si un nœud a trois voisins plus proches de la destination finale que lui). Les avantages de cette approche avec rencontres aveugles viennent du fait qu'elle ne nécessite pas de synchronisation, donc permet de développer un protocole MAC complètement asynchrone, et notamment robuste face aux dérives d'horloges, qui peuvent être très importantes lorsque les nœuds sont inactifs pendant de longues durées.

Nous nous sommes basés sur cette approche de planification pour proposer un protocole MAC asynchrone adapté aux applications des RCSF, et qui peut opérer avec des taux d'activité faibles (environ 1 %). Nous avons déterminé la durée optimale du cycle qui assure que les rencontres puissent être détectées, dans des conditions réalistes (en supposant des liens avec pertes et en prenant en compte la congestion). Nous avons montré que le protocole peut réaliser des délais raisonnables en fragmentant davantage les activités, sans impacter trop le taux de livraison. Des simulations approfondies menées sur NS-2 dans divers scénarios montrent que notre protocole peut assurer de bonnes performances en termes de taux de livraison, de délai et de consommation énergétique.

Pour réduire davantage le délai moyen de bout en bout pour la livraison des trames de données, nous avons proposé SLACK-MAC, qui est un protocole MAC asynchrone et adaptatif, fonctionnant avec un taux d'activité faible (1 %). Initialement, les nœuds dans SLACK-MAC, activent leur module radio aléatoirement et indépendamment, et construisent un historique des communications réussies. L'historique est utilisé pour déterminer les prochaines dates d'activation, il en résulte un comportement auto-adaptatif. Nous avons montré que SLACK-MAC atteint de bonnes performances avec une taille d'historique limitée à seulement six entrées, et quelques cycles seulement sont nécessaires pour remplir les listes.

Nous avons comparé SLACK-MAC et le protocole MAC de base que nous avons proposé avec les protocoles MAC existants. Nos protocoles assurent un bon compromis en termes de taux de livraison des trames, de délai de bout en bout et de consommation d'énergie.

La plupart des protocoles de routage pour les réseaux de capteurs nécessitent des messages de contrôle pour maintenir leur voisinage et prendre des décisions de routage. Ces protocoles fournissent généralement de très faibles performances lorsque le taux d'activité est faible (à savoir, inférieur à 1 %), vu que la charge en paquets de contrôle devient significativement plus élevée que le trafic de données.

Nous avons donc proposé dans un premier temps le protocole de routage par gradient ADC-GRAB, basé sur notre protocole MAC à rencontres aveugles. Dans ADC-GRAB, les nœuds utilisent un mécanisme de gradient pour sélectionner de façon opportuniste un nœud intermédiaire pour les paquets de données, pendant leur temps d'activité.

Nous avons aussi proposé le protocole de routage E-ADCR, basé sur un mécanisme d'inondation, et fonctionnant également avec notre protocole MAC à rencontres aveugles. Dans E-ADCR, une fois qu'un nœud est actif, il tente de diffuser autant de paquets que possible. Quand un voisin reçoit finalement l'une de ces transmissions, le voisin commence à diffuser ce paquet à son tour. Le protocole E-ADCR nécessite un nombre de messages de contrôle limités (uniquement pendant la phase d'initialisation).

Nous avons comparé par simulation E-ADCR avec un protocole de routage par inondation de la littérature, ainsi qu'avec ADC-GRAB, dans différents scénarios. Les résultats montrent que E-ADCR donne de bonnes performances par rapport aux deux autres protocoles en termes de délai de bout en bout et de taux de livraison des paquets, pour des taux d'activité inférieures à 1 %. Cependant, au delà des taux d'activité de 2 %, ADC-GRAB assure de meilleures performances que les deux autres protocoles.

Des résultats obtenus par expérimentation sur des nœuds TelosB de Crossbow confirment les résultats obtenus par simulation. E-ADCR assure de bonnes performances par rapport à ADC-GRAB pour de très faibles taux d'activité.

5.2 Perspectives pour les travaux futurs

Bien que les résultats présentés dans cette thèse par simulation et par expérimentation montrent une indication des performances de nos protocoles dans un réseau réel et montrent que nos concepts fonctionnent sur des nœuds capteurs réels, de nom-

breuses possibilités pour étendre la portée de cette thèse restent à exploiter. Ce qui suit présente quelques-unes de ces ouvertures. Nous présentons nos perspectives à court et long termes.

5.2.1 Perspectives à court terme

Dans nos évaluations, nous avons certes considéré un lien avec pertes entre les nœuds, mais la valeur de l'écart type du modèle de propagation shadowing que nous avons choisi fait l'hypothèse d'un médium stable, ce qui peut être considéré trop optimiste. Nous envisageons d'affiner le réalisme de cette étude en considérant un écart type qui varie en fonction du temps. Cette variation permet par exemple de prendre en compte les interférences avec d'autres protocoles travaillant dans la même bande de fréquences (exemple : interactions entre WiFi et 802.15.4 dans la bande de fréquences 2.4 GHz), ou des modifications de conditions de propagation dépendant de changements de l'environnement (exemple : entre le jour et la nuit, en présence de végétation ou non, etc.).

Pour le protocole A_{aa}MAC à rencontre aveugle, nous avons montré que diviser la période d'activité en plusieurs petites périodes d'activités dans un même cycle (au lieu d'avoir une seule longue activité) réduit le délai avant que deux paires de nœuds quelconques partagent une activité commune. Cependant, ce mécanisme réduit la durée de l'activité commune entre les nœuds. Nous envisageons d'utiliser ce mécanisme pour détecter une rencontre et si besoin étendre la durée de l'activité, tout en maintenant fixe la durée de l'activité dans chaque cycle. Par exemple, nous pouvons étendre la durée de l'activité tant que les nœuds communiquent et dormir plus longtemps ensuite. Cette modification du protocole MAC peut s'appliquer à d'autres protocoles étudiés, dont SLACK-MAC, ou d'autres protocoles de la littérature comme RI-MAC.

5.2.2 Perspectives à long terme

Nous envisageons d'améliorer le protocole de routage ADC-GRAB et E-ADCR. Nous avons supposé que les liens sont stables dans nos évaluations, ce qui interdit toute modification de la topologie. Adapter ces protocoles aux réseaux dynamiques (ajout ou disparition de nœud) ou mobiles est une perspective intéressante. Dans le cas du protocole de routage par inondation E-ADCR, les modifications nous semblent plus simples, étant donné que le protocole ne stocke pas beaucoup d'informations sur son voisinage. Dans le cas du protocole de routage par gradient ADC-GRAB, le travail sera plus conséquent, puisqu'il sera nécessaire d'adapter le gradient au fur et à mesure du déplacement des nœuds, ce qui nécessite la mise en relation de la vitesse

des nœuds et de la durée de mise à jour du gradient. Il est probable que le surcoût du protocole par gradient soit élevé.

Nous avons montré que le mécanisme basé sur l'historique du protocole SLACK-MAC permet d'améliorer la probabilité de communications fructueuses et de réduire le temps d'attente avant une activité commune. Il est possible d'appliquer ce mécanisme sur d'autres protocoles MAC probabilistes existants comme RI-MAC pour améliorer leurs performances.

Nous envisageons aussi de créer un protocole de routage hybride combinant le mécanisme d'inondation de E-ADCR avec le mécanisme basé sur un gradient de ADC-GRAB. Ce protocole de routage pourrait aussi intégrer un protocole MAC utilisant un mécanisme de priorité des paquets suivant le taux d'activité, et former ainsi un unique protocole cross-layering MAC/routage. L'utilisation d'un paramètre pourrait permettre à l'utilisateur de contrôler les performances selon le type de déploiement. Une étude parallèle pourrait permettre d'aider l'utilisateur à choisir le paramétrage du protocole en fonction du besoin des applications.

Nous avons réalisé une évaluation réaliste de nos mécanismes sur une plate-forme matérielle en vue de valider nos simulations. Les expérimentations ont été réalisées dans notre laboratoire sur une grande table avec une portée minimale des nœuds capteurs. Il serait intéressant d'étendre le champ de cette expérimentation, de différentes manières :

- en réalisant une expérimentation sur le terrain (sur un volcan par exemple) qui ferait face aux conditions temporaires,
- en comparant les sources de différences entre les résultats de simulation et les résultats expérimentaux.

Finalement d'autres perspectives seraient de :

- étudier nos protocoles pour d'autres types d'applications (exemple : domotique ? applications peu contraintes en énergie (taux d'activité élevés?)),
- nous intéresser aux autres couches du modèle OSI : couche physique pour des communications en extérieur, couche transport ou application,
- nous intéresser à des nœuds qui auraient plusieurs modules radios ou des modules radios ayant des bandes de fréquence paramétrables, pour pouvoir adapter la bande de fréquence aux conditions actuelles.

ANNEXE A

Protocole expérimental

Dans cette partie nous détaillons le processus d'expérimentation, de la conception du protocole jusqu'à la réalisation des tests d'évaluation.

A.1 Conception du protocole

La conception de notre protocole expérimental a été réalisée sur le système d'exploitation TinyOS pour la plateforme TelosB et les programmes sont en langage NesC [GLR+03].

Le langage NesC repose sur une architecture à base de composants. Une application en NesC peut être mise en œuvre par un certain nombre de fichiers avec une extension en *.h* et en *.nc*. Les extensions en *.h* définissent les constantes et les structures de données, et les extensions *.nc* définissent le *module* ou la *configuration* ainsi que les interfaces utilisées. Les interfaces sont fournies ou utilisées par les deux types de composants en NesC que sont *module* et *configuration*. Un *module* NesC définit un composant qui peut être appelé par un autre composant. Dans une configuration, les composants sont liés entre eux par des interfaces bidirectionnelles.

Pour nos applications, nous avons défini trois fichiers différents, un fichier en *.h* (Gradient.h) contenant les constantes et les différentes structures, et deux fichiers en *.nc*, dont l'un est pour la configuration (appelé GradientAppC.nc) et l'autre pour le module (appelé GradientC.nc). Le premier contient la déclaration des composants utilisés et leurs liaisons avec les interfaces explicitées dans le deuxième qui comporte en plus des interfaces le code de l'application. Les figures A.1, A.2, et A.3 montrent respectivement un extrait de ces fichiers pour l'application Gradient.

Comme nous pouvons voir sur la figure A.1 le fichier Gradient.h comporte nos

```
#ifndef GRADIENT_H
#define GRADIENT_H
enum {
        AM_GRADIENTMSG = 6,
        TIMER_CYCLE_MILLI = 5000U,
        TIMER_MINIMUM_MILLI = 5,
        TIMER_ACK_TIMEOUT_MILLI = 5,
        TIMER_DATE_MILLI = 1,
        MAXI_QUEUE_SIZE = 20,
        MAXI_RETRANSMISSION = 4,
        PUITS = 1,
        MinRSSI = 65495U,
        SimulationDuration = 2000U
        TRX_POWER = 1,
};
typedef nx_struct GradientMsg {
  nx_uint16_t nodeid;
  nx_uint16_t counter;
  nx_uint16_t data;
  nx_uint16_t hop;
  nx_uint16_t activity;
```

Figure A.1 – Exemple de déclaration des constantes et structures.

```
#include <Timer.h> #include <UserButton.h> #include "printf.h" #include "CC2420.h"
#include "Gradient.h"
 configuration GradientAppC {
 }
 implementation {
   components MainC;
   components LedsC;
   components RandomMlcgC;
   components UserButtonC;
   components CC2420ActiveMessageC;
   components GradientC as App;
   components new TimerMilliC() as TimerActivity;
   components ActiveMessageC;
   components SerialActiveMessageC;
   components new AMSenderC(AM_GRADIENTMSG);
   components new AMReceiverC(AM_GRADIENTMSG);
   components new SerialAMReceiverC(AM_GRADIENTMSG);
   App.Boot -> MainC;
   App.Leds -> LedsC;
   App.TimerActivity -> TimerActivity;
```

Figure A.2 – Exemple de déclaration des composants et leur liaison avec les interfaces utilisées.

constantes (par exemple le paramètre *TRX_ POWER=1* permet de fixer la puissance de transmission à 1) et nos structures de données (par exemple GradientMsg définit la structure de nos paquets).

```
#include <Timer.h> #include <UserButton.h> #include "printf.h" #include "CC2420.h"
#include "Gradient.h"
 module GradientC {
   uses interface Boot;
   uses interface Leds;
   uses interface Timer<TMilli> as TimerActivity;
   uses interface Packet;
   uses interface AMPacket;
   uses interface AMSend;
   uses interface SplitControl as AMControl;
   uses interface Receive;
   uses interface Random;
   uses interface Init;
   uses interface ParameterInit<uint16_t> as SeedInit;
   uses interface Read<uint16_t>;
   uses interface Get<button_state_t> as Get;
   uses interface Notify<button_state_t> as Notify;
   uses interface CC2420Packet;
 }
 implementation {
   bool busy = FALSE;
```

Figure A.3 – Exemple de définition de module.

Dans le fichier GradientAppC.nc (voir la figure A.2) nous déclarons les différents composants dont nous avons besoin et leurs liaisons avec les interfaces définies dans le module GradientC du fichier GradientC.nc (voir la figure A.3). La partie implémentation du fichier GradientC.nc comporte le code de notre application Gradient (protocole de routage décrit dans la partie 3.2.1).

A.2 Configuration des TelosB

La configuration des nœuds TelosB se fait comme suit :

- Dans un terminal se positionner à le chemin de l'application, par exemple :

  ```
  cd /opt/tinyos-2.1.0/apps/ABY-Apps/gradient
  ```

- Compiler l'application à l'aide de la commande : *make telosb*. La taille respective des codes pour les protocoles de routage ADC-GRAB et E-ADCR après compilation est représentée sur la figure A.5 et A.4.
- Insérer le telosb dans un port USB, puis taper la commande *motelist* pour avoir la liste des TelosB disponibles.

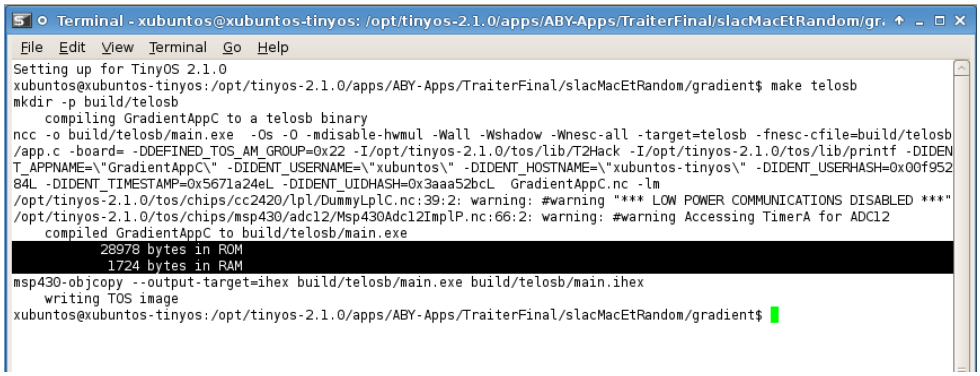


Figure A.4 – Taille du code pour ADC-GRAB après compilation.

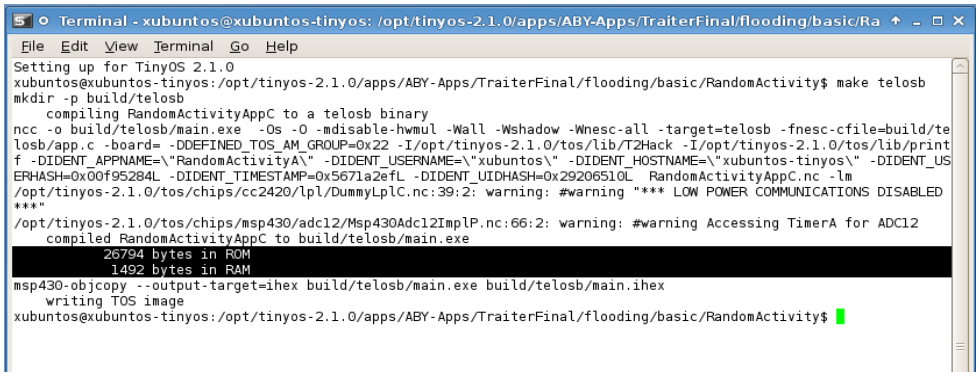


Figure A.5 – Taille du code pour E-ADCR après compilation.

```
$motelist
Reference  Device           Description
XBS5H6PH   /dev/ttyUSB0     XBOW Crossbow Telos Rev.B
```

- Installer l'application sur un TelosB spécifique (par exemple celui en port USB0) et lui attribuée un identifiant unique (par exemple 2) avec la commande :

  ```
  make telosb install,2 bsl,/dev/ttyUSB0
  ```

A.3 Mise en place du réseau et réalisation des tests

Une fois que tous les nœuds (y compris le puits) sont configurés, on procède dans un premier temps à la mise en place du réseau qui dure 20 secondes. Le puits diffuse successivement deux balises après l'appui du *UserButton*. Une balise comporte la distance en nombre de sauts du puits (saut 0 pour le puits) et l'instant de démarrage des mesures. Tous les nœuds qui entendent ces balises avec un RSSI supérieur au seuil fixé mettent à jour la distance (avec celle contenue dans la balise incrémentée de 1) et l'instant de démarrage, puis choisissent un instant aléatoire entre 0 et 2 ms après laquelle ils rediffusent la balise avec les nouvelles valeurs de distance et

de démarrage. Après cette petite phase d'initialisation, tous les nœuds y compris le puits ont le même repère temporel. Il faut noter que pour avoir une configuration d'un réseau multi-saut (avec 4 sauts maximum du puits) nous avons limité la portée de communication des nœuds à -25 dBm (cela représente la puissance minimale). Ce qui fait que deux nœuds même parfois proches peuvent ne pas être à portée de communication. Pour avoir un réseau connexe et la configuration souhaitée, parfois l'étape de mise en place doit être répétée plusieurs fois.

Une fois la mise en place réalisée, l'appui du *UserButton* des nœuds sources leur permet de réaliser des mesures périodiques et de s'auto-organiser avec le voisinage (en fonction du protocole de communication défini dans l'application) pour envoyer ces mesures au puits.

Chaque paquet comporte un identifiant unique (composé par l'identifiant du nœud l'ayant généré et du numéro du paquet, par exemple de 0 à x), l'instant en secondes et en milisecondes de la génération du paquet depuis l'instant de synchronisation. Ainsi, le puits peut déterminer aisément le délai de bout en bout d'un paquet quelconque à partir de sa date de génération jusqu'à sa réception. Il est important de noter que nous n'avons pas pris en compte les différentes dérives possible des nœuds TelosB. Le puits étant connecté à une machine, nous utilisons la fonction *printf* pour afficher les informations sur les paquets reçus. Ensuite, nous récoltons toutes ces informations en écoutant le port USB à l'aide de la commande :

```
java net.tinyos.tools.PrintfClient -comm serial@/dev/ttyUSB0:115200
```

Ces informations peuvent être inscrites dans un fichier *resultat.txt* par exemple à l'aide de la commande :

```
java net.tinyos.tools.PrintfClient -comm serial@/dev/ttyUSB0:115200 > resultat.txt
```

Un extrait du fichier *resultat.txt* est montré dans la figure A.6. Chaque test est répété à deux reprises et chacun dure 2000 secondes. Nous avons adapté le code pour faire évoluer le taux d'activité des nœuds et la période de génération de trafic pour une même configuration avec une petite phase de transition de 10 secondes entre deux tests de 2000 secondes. Chaque fichier *resultat.txt* comporte respectivement les résultats quand nous fixons une période de génération de 5 secondes (avec une variation du taux d'activité de 0.25 % à 10 %) et les résultats pour une période de génération de 60 secondes (avec la même variation du taux d'activité de 0.25 % à 10 %).

A.4 Exploitation des résultats

Nous utilisons un code en Perl pour exploiter nos résultats.

NodeID	NumP	Delay(s) mS	hop
20	0	36 218	2
8	0	55 444	2
19	0	60 793	3
14	0	81 285	4
14	0	81 287	4
14	0	88 800	4
8	1	38 963	2
8	1	39 986	2
19	1	39 988	3
19	1	39 991	3
4	1	74 949	3
8	2	15 898	2
8	2	24 383	2
20	1	103 63	2
14	0	163 65	4
14	3	49 838	4
4	3	63 692	3
4	4	15 488	3
4	4	15 491	3
4	4	23 682	3
20	3	102 580	2
14	3	102 596	4
19	4	56 557	3
19	4	56 638	3
8	3	116 664	2
8	4	56 666	2
8	4	56 669	2
19	4	60 86	3
20	4	98 563	2
20	5	38 565	2
8	5	43 567	2
8	5	43 570	2
8	5	47 398	2
8	6	15 437	2
8	6	15 463	2
14	6	30 973	4
20	6	104 759	2
19	5	167 695	3
4	7	47 697	3

Figure A.6 – Exemple de fichier résultats.

Sur la figure A.6 (extrait du fichier *resultat.txt*), le premier champ représente l'identifiant du nœud source, le deuxième champ représente le numéro de génération du paquet, le troisième est le délai en secondes, le quatrième représente les milisecondes additionnelles par rapport au délai en secondes et le cinquième est la distance en nombre de sauts par rapport au puits. Nous calculons le taux de livraison en faisant le rapport du nombre de paquets reçus par le puits sur le nombre total de paquets générés par les nœuds sources. Notons que les doublons ne sont pas considérés. Le délai moyen de livraison est calculé à partir du premier paquet reçu en cas de doublon.

Liste des publications

1. **Affoua Thérèse Aby**, Alexandre Guitton, Pascal Lafourcade, Michel Misson. History-based MAC Protocol for Low Duty-Cycle Wireless Sensor Networks: the SLACK-MAC Protocol. In journal EAI Endorsed Transactions on Mobile Communications and Applications, 2016.

2. **Affoua Thérèse Aby**, Alexandre Guitton, Pascal Lafourcade, Michel Misson. SlackMAC: Adaptive mac protocol for low duty-cycle wireless sensor networks. In ADHOCNETS (EAI International Conference on Ad Hoc Networks), 2015.

3. **Affoua Thérèse Aby**, Alexandre Guitton, Michel Misson. E-adcr: Energy-efficient asynchronous low duty-cycle routing protocol. In JCM (Journal of Communications), 2015.

4. **Affoua Thérèse Aby**, Alexandre Guitton, Michel Misson. Asynchronous blind MAC protocol for wireless sensor networks. In IWCMC (IEEE International Wireless Communications and Mobile Computing Conference), 2014.

5. **Affoua Thérèse Aby**, Alexandre Guitton, Michel Misson. Study of blind rendez-vous in low power wireless sensor networks. In VTC Spring (IEEE Vehicular Technology Conference), 2014.

6. **Affoua Thérèse Aby**, Alexandre Guitton, Michel Misson. MAC mechanism for a scalable wireless sensor network using independent duty cycles. In New and smart Information Communication Science and Technology to support Sustainable Development, 2013.

Nomenclature

ACK	Acknowledgement. 30
ADC	Analog to Digital Converter. 20
ADC-GRAB	Asynchronous low Duty-Cycle GRAgient Based routing protocol. 99
ALPL	Adaptive Low Power Listening. 56
AODV	AdHoc On-demand Distance Vector. 51
ASSORT	Asynchronous Sleep-wake Schedules and Opportunistic Routing. 56
B-MAC	Berkeley MAC. 39
BE	Backoff Exponent. 27
BI	Beacon Interval. 27
BO	macBeaconOrder. 27
CAP	Contention Access Period. 27
CCA	Clear Channel Assessment. 26
CFP	Contention Free Period. 27

CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance. 26
CTS	Clear To Send. 30
CW	Contention Window. 32
D-MAC	Data-gathering MAC. 29
DIFS	DCF (Distributed Coordination Function) InterFrame Space. 32
E-ADCR	Energy-efficient Asynchronous low Duty-Cycle Routing protocol. 94
ED	Energy Detection. 26
ENO	Energy Neutral Operation. 47
FFD	Full-Function Devices. 25
FIFO	First In First Out. 141
FTSP	Fast Time-dependent Shortest Path algorithm. 55
GPS	Global Positioning System. 20
GRAB	GRAdient Broadcast. 53
GTS	Guaranteed Time Slot. 27
HRP	Hierarchical Routing Protocol. 51
IETF	Internet Engineering Task Force. 57
LPL	Low Power Listening. 39
LPP	Low Power Probing. 43
LQI	Link Quality Indicator. 26

LR-WPANs	Low-Rate Wireless Personal Area Networks. 25
MAC	MAC Medium Access Control. 23, 24
MANET	Mobile Ad Hoc Network. 49
NB	Number of Backoffs. 27
NLDE	Network Layer Data Entity. 49
NLME	Network Layer Management Entity. 49
OECS	Opportunistic Energy Cost with Sleep-wake schedules. 56
OSI	Open Systems Interconnection. 24
OTCL	Object Tool Command Language. 107
PAN	Personal Area Network. 25
Q-MAC	Query-MAC. 29
R-MAC	Routing-enhanced duty cycle MAC. 29
RAM	Random Access Memory. 142
RCSF	Réseaux de Capteurs Sans Fil. 19
RFD	Reduced-Function Devices. 25
ROLL	Routing Over Low power and Lossy networks. 57
RPL	IPv6 routing protocol for low power and lossy networks. 57
RREP	Route REPly. 51
RREQ	Route REQuest. 51
RSSI	Received Signal Strength Indication. 94
RTS	Request To Send. 30

S-MAC	Sensor-MAC. 29
SAP	Service Access Points. 49
SD	Superframe Duration. 27
SLACK-MAC	Adaptive MAC Protocol for Low Duty-Cycle Wireless Sensor Networks. 88
SO	macSuperframeOrder. 27
T-MAC	Timeout-MAC. 29
TTL	Time-to-Live. 95
USB	Universal Serial Bus. 142
ZC	ZigBee Coordinator. 50
ZED	ZigBee End Device. 50
ZR	ZigBee Router. 50

Références

[ADJL13] K. Altisen, S. Devismes, R. Jamet, and P. Lafourcade. SR3: Secure resilient reputation-based routing. In *IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2013, Cambridge, MA, USA, May 20-23, 2013*, pages 258–265. IEEE, 2013.

[AGLM15] A. T. Aby, A. Guitton, P. Lafourcade, and M. Misson. Slack-mac: Adaptive mac protocol for low duty-cycle wireless sensor networks. In *Ad Hoc Networks*, pages 69–81. Springer, 2015.

[AGM13] A. T. Aby, A. Guitton, and M. Misson. MAC mechanism for a scalable wireless sensor network using independent duty cycles. In *New and smart Information Communication Science and Technology to support Sustainable Development*, 2013.

[AGM14a] A. T. Aby, A. Guitton, and M. Misson. Asynchronous blind mac protocol for wireless sensor networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*, pages 712–717. IEEE, 2014.

[AGM14b] A. T. Aby, A. Guitton, and M. Misson. Study of blind rendez-vous in low power wireless sensor networks. In *Vehicular Technology Conference (VTC Spring), 2014 IEEE 79th*, pages 1–5. IEEE, 2014.

[AGM15] A. T. Aby, A. Guitton, and M. Misson. Efficient flooding-based routing protocol with random wake-up for very low duty-cycle wsns. *Journal of Communications*, 10(6):385–395, 2015.

[AWA11] M. R. Akhavan, T. Watteyne, and A. H. Aghvami. Enhancing the performance of rpl using a receiver-based mac protocol in lossy wsns. In *Telecommunications (ICT), 2011 18th International Conference on*, pages 191–194. IEEE, 2011.

[Bag05] A. Baggio. Wireless sensor networks in precision agriculture. In *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005), Stockholm, Sweden*, 2005.

[BCH15] S. Bindel, S. Chaumette, and B. Hilt. F-etx: an enhancement of etx metric for wireless mobile networks. In *Communication Technologies for Vehicles*, pages 35–46. Springer, 2015.

[BGAH06] M. Buettner, Y. Gary, V., E. Anderson, and R. Han. X-MAC: A short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320. ACM, 2006.

[BHB+08] J. Balendonck, J. Hemming, Van T. B., L. Incrocci, A. Pardossi, P. Marzialetti, et al. Sensors and wireless sensor networks for irrigation management under deficit conditions (flow-aid). In *Proceedings of the International Conference on Agricultural Engineering (AgEng 2008)*, 2008.

[BM05] S. Biswas and R. Morris. Exor: opportunistic multi-hop routing for wireless networks. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 133–144. ACM, 2005.

[BN10] R. Bruno and M. Nurchis. Survey on diversity-based routing in wireless mesh networks: Challenges and solutions. *Computer Communications*, 33(3):269–282, 2010.

[BST11] L. Ben Saad and B. Tourancheau. Exploiting addresses correlation to maximize lifetime of ipv6 cluster-based wsns. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*, pages 483–489. IEEE, 2011.

[CGHN13] Long Cheng, Yu Gu, Tian He, and Jianwei Niu. Dynamic switching-based reliable flooding in low-duty-cycle wireless sensor networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 1393–1401. IEEE, 2013.

[CGM08] G. Chalhoub, A. Guitton, and M. Misson. Mac specifications for a wpan allowing both energy saving and guaranteed delay. In *Wireless Sensor and Actor Networks II*, pages 221–232. Springer, 2008.

[Cha09] G. Chalhoub. *MaCARI: une méthode d'accès déterministe et économe en énergie pour les réseaux de capteurs sans fil*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2009.

[CLG+09] G. Chalhoub, E. Livolant, A. Guitton, A. Van Den Bossche, M. Misson, and T. Val. Specifications and evaluation of a mac protocol for a lp-wpan. *Ad Hoc & Sensor Wireless Networks*, 7(1-2):69–89, 2009.

[DCABM03] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. pages 134–146. In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom'03), 2003.

[DCABM05] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[DDB13] M. Doudou, D. Djenouri, and N. Badache. Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks. *Communications Surveys & Tutorials, IEEE*, 15(2):528–550, 2013.

[DDBB14] M. Doudou, D. Djenouri, N. Badache, and A. Bouabdallah. Synchronous contention-based mac protocols for delay-sensitive wireless sensor networks: A review and taxonomy. *Journal of Network and Computer Applications*, 38:172–184, 2014.

[DSJ07] S. Du, A. K. Saha, and D. B. Johnson. Rmac: A routing-enhanced duty-cycle mac protocol for wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1478–1486. IEEE, 2007.

[Dun11] A. Dunkels. The contikimac radio duty cycling protocol. 2011.

[EHDH04] A. El-Hoiydil, J.-D. Decotigniel, and J. Hernandez. WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In *Algorithmic Aspects of Wireless Sensor Networks*, volume 3121 of *LNCS*, pages 18–31. Springer Berlin / Heidelberg, 2004.

[FVDBV12] N. Fourty, A. Van Den Bossche, and T. Val. An advanced study of energy consumption in an ieee 802.15. 4 based network: Everything but the truth on 802.15. 4 node lifetime. *Computer Communications*, 35(14):1759–1767, 2012.

[GGJH09] S. Guo, Y. Gu, B. Jiang, and T. He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. *Proc. ACM MobiCom'09*, pages 133–144, 2009.

[GGJH14] S. Guo, Y. Gu, B. Jiang, and T. He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. *Computers, IEEE Transactions on*, 63(11):2787–2802, 2014.

[GKZ+11] S. Guo, S. K. Kim, T. Zhu, Y. Gu, and T. He. Correlated flooding in low-duty-cycle wireless sensor networks. In *Proc. IEEE ICNP'11*, pages 383–392, October 2011.

[GLR+03] D. Gay, P. Levis, Von B. R., M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Acm Sigplan Notices*, volume 38, pages 1–11. ACM, 2003.

[GLS+14] E. Ghadimi, O. Landsiedel, P. Soldati, S. Duquennoy, and M. Johansson. Opportunistic routing in low duty-cycle wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 10(4):67, 2014.

[HB07] M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1–6. IEEE, 2007.

[HC12] L. Huang and S. Cheng. Unmanned monitoring system of rivers and lakes based on wsn. In *Systems and Informatics (ICSAI), 2012 International Conference on*, pages 495–498. IEEE, 2012.

[HCXT09] Pei Huang, Hongyang Chen, Guoliang Xing, and Yongdong Tan. SGF: a state-free gradient-based forwarding protocol for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 5(2):14, 2009.

[HHL88] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.

[HKWC14] C. Hsu, M. S. Kuo, S. C Wang, and C. F. Chou. Joint design of asynchronous sleep-wake scheduling and opportunistic routing in wireless sensor networks. *Computers, IEEE Transactions on*, 63(7):1840–1846, 2014.

[HSW+00a] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ACM SIGOPS operating systems review*, volume 34, pages 93–104. ACM, 2000.

[HSW+00b] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *In Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[HUM+11] D. Hughes, J. Ueyama, E. Mendiondo, N. Matthys, W. Horré, S. Michiels, C. Huygens, W. Joosen, K. L. Man, and S. Guan. A middleware platform to support river monitoring using wireless sensor networks. *Journal of the Brazilian Computer Society*, 17(2):85–102, 2011.

[IEE03] IEEE 802.15. Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). Standard 802.15.4 R2003, ANSI/IEEE, 2003.

[IEE06] IEEE 802.15. Part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs). Standard 802.15.4 R2006, ANSI/IEEE, 2006.

[IEE11] IEEE 802.15. Ieee standard for local and metropolitan area networks – part 15.4: Low-rate wireless personal area networks (LR-WPANs). Standard 802.15.4 R2011, ANSI/IEEE, 2011.

[IEE12] IEEE 802.15. Ieee standard for local and metropolitan area networks – part 15.4: Low-rate wireless personal area networks (LR-WPANs) amendment 1: Mac sublayer. Ieee standard 802.15.4e-2012 (amendment to ieee standard 802.15.4-2011), ANSI/IEEE, 2012.

[Ins06] Texas Instruments. CC2420: 2.4 GHz IEEE 802.15. 4/ZigBee-ready RF transceiver. *Disponible sur http://www.ti.com/lit/gpn/cc2420*, 2006.

[JBL07] R. Jurdak, P. Baldi, and C. V. Lopes. Adaptive low power listening for wireless sensor networks. *Mobile Computing, IEEE Transactions on*, 6(8):988–1004, 2007.

[JMB01] D. B. Johnson, D. A. Maltz, and J. Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Computer Science Department Carnegie Mellon University Pittsburgh, PA*, pages 15213–3891, 2001.

[JRO10] R. Jurdak, A. G. Ruzzelli, and G. M. P. O'Hare. Radio sleep mode optimization in wireless sensor networks. *Mobile Computing, IEEE Transactions on*, 9(7):955–968, 2010.

[KAT06] A. Koubaa, M. Alves, and E. Tovar. A comprehensive simulation study of slotted csma/ca for ieee 802.15. 4 wireless sensor networks. In *5th IEEE International Workshop on Factory Communication Systems*, pages 183–192. IEEE, 2006.

[KHCL07] K. Klues, G. Hackmann, O. Chipara, and C. Lu. A component-based architecture for power-efficient media access control in wireless sensor

networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 59–72. ACM, 2007.

[KHZS07] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):32, 2007.

[KJK15] Seong Cheol Kim, Jun Heon Jeon, and Joong Jae Kim. Energy and delay efficient duty-cycle mac protocol for multi-hop wireless sensor networks. 2015.

[Knu98] D. E. Knuth. *The art of computer programming: sorting and searching*, volume 3. Pearson Education, 1998.

[KS06] K. Kim and K. G. Shin. On accurate measurement of link quality in multi-hop wireless mesh networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking*, pages 38–49. ACM, 2006.

[LC08] D. Lee and K. Chung. Ra-mac: an energy efficient and low latency mac protocol using rts aggregation for wireless sensor networks. In *Advanced Technologies for Communications, 2008. ATC 2008. International Conference on*, pages 150–153. IEEE, 2008.

[LF76] K. Leentvaar and J. H. Flint. The capture effect in fm receivers. *Communications, IEEE Transactions on*, 24(5):531–539, 1976.

[LG09] Philip Levis and David Gay. *TinyOS programming*. Cambridge University Press, 2009.

[LGDJ12] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low power, low delay: Opportunistic routing meets duty cycling. In *Proc. ACM/IEEE IPSN'12*, Beijing, China, April 2012.

[LKK+08] J. Lee, J. Kim, D. Kim, P. K. Chong, J. Kim, and P. Jang. Rfms: Real-time flood monitoring system with wireless sensor networks. In *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, pages 527–528. IEEE, 2008.

[LKR04] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in sensor networks. In *Ad Hoc and Sensor Networks*, April 2004.

[LKR07] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency MAC for tree-based data gathering in sensor networks. In *Wireless Communications and Mobile Computing*, volume 7, pages 863–875, September 2007.

[LLL14] Z. Li, M. Li, and Y. Lui. Towards energy-fairness in asynchronous duty-cycling sensor networks. In *ACM Transactions on Sensor Networks (TOSN)*, volume 10, April 2014.

[LR10] S. Lai and B. Ravindran. On distributed time-dependent shortest paths over duty-cycled wireless sensor networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[LR13] S. Lai and B. Ravindran. Least-latency routing over time-dependent wireless sensor networks. *Computers, IEEE Transactions on*, 62(5):969–983, 2013.

[LW09] Jiakang Lu and Kamin Whitehouse. *Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks*. IEEE, 2009.

[MDZ+16] K. Mekki, W. Derigent, A. Zouinkhi, E. Rondeau, A. Thomas, and M. N. Abdelkrim. Non-localized and localized data storage in large-scale communicating materials: Probabilistic and hop-counter approaches. *Computer Standards & Interfaces*, 44:243–257, 2016.

[MELT08] Razvan Musaloiu-E, C-JM Liang, and Andreas Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*, pages 421–432. IEEE, 2008.

[ML08] D. Moss and P. Levis. Box-macs: Exploiting physical and link layer boundaries in low-power networking. *Computer Systems Laboratory Stanford University*, pages 116–119, 2008.

[MMKR13] J. Misic, V. B. Misic, M. S. I. Khan, and M. M. Rahman. Properties of blind rendezvous in channel hopping cognitive piconets. In *IEEE VTC-Fall*, 2013.

[mot] Motorola mc9s08gt60a. http://www.digchip.com/datasheets/newparts/200808/-1127519.php.

[Mou13] A. Mouradian. *Proposition et vérification formelle de protocoles de communications temps-réel pour les réseaux de capteurs sans fil*. PhD thesis, Lyon, INSA, 2013.

[MZD+16] K. Mekki, A. Zouinkhi, W. Derigent, E. Rondeau, A. Thomas, and M. N. Abdelkrim. Usee: A uniform data dissemination and energy efficient protocol for communicating materials. *Future Generation Computer Systems*, 56:651–663, 2016.

[NBP+06] V. D. Niek, L. Benoît, D. M. Pieter, M. Ingrid, D. Bart, and D. Piet. Throughput and delay analysis of unslotted IEEE 802.15.4. *Journal of Networks*, 1:20–28, 2006.

[NJY13] K. Nguyen, Y. Ji, and S. Yamada. Low overhead mac protocol for low data rate wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2013.

[NNL+14] K. Nguyen, V. Nguyen, D. Le, Y. Ji, D. A. Duong, and S. Yamada. A receiver-initiated mac protocol for energy harvesting sensor networks. In *Ubiquitous Information Technologies and Applications*, pages 603–610. Springer, 2014.

[ns202] Network simulator 2, 2002. http://www.isi.edu/nsnam/ns.

[NUF11] T. Nagayama, M. Ushita, and Y. Fujino. Suspension bridge vibration measurement using multihop wireless sensor networks. *Procedia Engineering*, 14:761–768, 2011.

[OH12] Hoon Oh and Trung-Dinh Han. A demand-based slot assignment algorithm for energy-aware reliable data transmission in wireless sensor networks. *Wireless Networks*, 18(5):523–534, 2012.

[OR96] A. Orda and R.l Rom. Distributed shortest-path protocols for time-dependent networks. *Distributed computing*, 10(1):49–62, 1996.

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing. Request For Comments 3561, IETF, July 2003.

[PHC04] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *ACM Sensys*, November 2004.

[Ram10] M. V. Ramesh. Wireless sensor network for disaster monitoring. *Wireless Sensor Networks: Application-Centric Design*, 2010.

[sc+99] LAN/MAN standards committee et al. Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Standard 802.11 R2003, ANSI/IEEE, 1999.

[SET09] Winston KG Seah, Zhi Ang Eu, and Hwee-Pink Tan. Wireless sensor networks powered by ambient energy harvesting (wsn-heap)-survey and challenges. In *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on*, pages 1–5. Ieee, 2009.

[SGD+09] Y. Sun, O. Gurewitz, S. Du, L. Tang, and D. B. Johnson. Adb: an efficient multihop broadcast protocol based on asynchronous duty-cycling in wireless sensor networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 43–56. ACM, 2009.

[SGJ08] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *ACM Sensys*, 2008.

[SHSL09] W. Z. Song, R. Huang, B. Shirazi, and R. LaHusen. Treemac: Localized tdma mac protocol for real-time high-data-rate sensor networks. *Pervasive and Mobile Computing*, 5(6):750–765, 2009.

[TCSL13] H. Tang, J. Cao, C. Sun, and K. Lu. REA-MAC: A low latency routing enhanced asynchronous duty-cycle MAC protocol for wireless sensor networks. *Journal of Central South University*, 20:678–687, 2013.

[Teca] Crossbow Technology. Telosb specification. http://www.willow.co.uk/TelosB-Datasheet.pdf.

[Tecb] Memsic Technology. Telosb specification. http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb-datasheet.pdf.

[tel] Telit specification. http://www.telit.com/.

[TS+11] L. Tang, , Y. Sun, O. Gurewitz, and D. B. Johnson. Pw-mac: An energy-efficient predictive-wakeup mac protocol for wireless sensor networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1305–1313. IEEE, 2011.

[TSG+11] L. Tang, Y. Sun, O. Gurewitz, S. Du, and D. B. Johnson. EM-MAC: A dynamic multichannel energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, page 23. ACM, 2011.

[TSLF13] H. Tang, C. Sun, Y. Liu, and B. Fan. Low-latency asynchronous duty-cycle MAC protocol for burst traffic in wireless sensor networks. In *IWCMC'13*, pages 412–417, July 2013.

[VA06] N. A. Vasanthi and S. Annadurai. Energy efficient sleep schedule for achieving minimum latency in query based sensor networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 2, pages 214–219. IEEE, 2006.

[VRG+14] M. Vucinic, G. Romaniello, L. Guelorget, B. Tourancheau, F. Rousseau, O. Alphand, A. Duda, and L. Damon. Topology construction in

rpl networks over beacon-enabled 802.15. 4. In *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pages 1–7. IEEE, 2014.

[VRS03] Thiemo Voigt, Hartmut Ritter, and Jochen Schiller. Utilizing solar power in wireless sensor networks. In *Local Computer Networks, 2003. LCN'03. Proceedings. 28th Annual IEEE International Conference on*, pages 416–422. IEEE, 2003.

[VST14] A. C. Valera, W. S. Soh, and H. P. Tan. Survey on wakeup scheduling for environmentally-powered wireless sensor networks. *Computer Communications*, 52:21–36, 2014.

[WAJR+05] Geoffrey Werner-Allen, Jeff Johnson, Mario Ruiz, Jonathan Lees, and Matt Welsh. Monitoring volcanic eruptions with a wireless sensor network. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 108–120. IEEE, 2005.

[WHC+12] Y. Wang, Y. He, D. Cheng, Y. Liu, and X. Li. Triggercast: Enabling wireless collisions constructive. *arXiv preprint arXiv:1208.0664*, 2012.

[WHM+12] Y. Wang, Y. He, X. Mao, Y. Liu, Z. Huang, and X. Li. Exploiting constructive interference for scalable flooding in wireless sensor network. *INFOCOM'12*, pages 2104–2112, 2012.

[WL12] F. Wang and J. Liu. On reliable broadcast in low duty-cycle wireless sensor networks. In *Mobile Computing, IEEE Transactions*, volume 11, pages 767–779, May 2012.

[WPB+09] T. Watteyne, K. Pister, D. Barthel, M. Dohler, and I. Auge-Blum. Implementation of gradient routing in wireless sensor networks. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pages 1–6. IEEE, 2009.

[WT12] T Winter and P Thubert. Rpl: Ipv6 routing protocol for low power and lossy networks. rfc6550 (proposed standard). *Mars*, 2012.

[WZCZ10] X. Wang, X. Zhang, G. Chen, and Q. Zhang. Opportunistic cooperation in low duty cycle wireless sensor networks. In *IEEE ICC'10*, pages 1–5, May 2010.

[YHE02] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1567–1576. IEEE, 2002.

[YHE04] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *Networking, IEEE/ACM Transactions on*, 12(3):493–506, 2004.

[YM11] P. Yadav and J. A. McCann. Ya-mac: Handling unified unicast and broadcast traffic in multi-hop wireless sensor networks. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–9. IEEE, 2011.

[YSH06] W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle mac with scheduled channel polling. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 321–334. ACM, 2006.

[YWM05] L. Yu, N. Wang, and X. Meng. Real-time forest fire detection with wireless sensor networks. In *Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on*, volume 2, pages 1214–1217. IEEE, 2005.

[YWW+14] S. Yu, X. Wu, P. Wu, D. Wu, H. Dai, and G. Chen. CIRF: Constructive interference-based reliable flooding in asynchronous duty-cycle wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pages 2734–2738. IEEE, 2014.

[YZLZ05] F. Ye, G. Zhong, S. Lu, and L. Zhang. Gradient broadcast: A robust data delivery protocol for large scale sensor networks. *Springer-Verlag Wireless Networks*, 11(3):285–298, 2005.

[ZF06] Y. Zhang and M. P. J. Fromherz. A robust and efficient flooding-based routing for wireless sensor networks. *Journal of Interconnection Networks*, 7:549–568, December 2006.

[Zig08] ZigBee. ZigBee Specification. Standard ZigBee 053474r17, ZigBee Standards Organization, January 2008.

[Zim80] Hubert Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *Communications, IEEE Transactions on*, 28(4):425–432, 1980.

[ZZHZ10] T. Zhu, Z. Zhong, T. He, and Z. Zhang. Exploring link correlation for efficient flooding in wireless sensor networks. In *NSDI*, pages 49–64, 2010.

Réseaux de capteurs sans fil étendus dédiés aux collectes de données environnementales

Résumé

Les réseaux de capteurs sans fil sont utilisés dans de nombreuses applications de surveillance de l'environnement (par exemple, pour surveiller les volcans ou pour détecter les incendies de forêts). Dans de telles applications, les nœuds capteurs disposent d'une quantité limitée d'énergie, mais doivent fonctionner pendant des années sans avoir leurs batteries changées. La principale méthode utilisée pour permettre aux nœuds d'économiser leur énergie est de séquencer les périodes d'activité et d'inactivité. Cependant, la conception de protocoles MAC et de routage pour les applications avec des taux d'activité faibles est un défi.

Dans cette thèse nous proposons des protocoles MAC avec de très faibles taux d'activité (moins de 1% d'activité) et des protocoles de routages adaptés pour des réseaux de capteurs sans fil dédiés aux applications de surveillance environnementale.

Nos protocoles sont analysés et comparés aux protocoles existants par simulation et par expérimentation sur des nœuds TelosB. Malgré un taux d'activité très faible pour tous les nœuds, nos protocoles sont capables d'obtenir de bonnes performances, contrairement aux autres protocoles de la littérature, qui ne sont pas adaptés à opérer avec de faibles taux d'activité.

Mots-clés : réseaux de capteurs sans fil, protocoles MAC, protocoles de routage, taux d'activité faible, économie d'énergie, surveillance environnementale.

Large wireless sensor networks dedicated to environmental monitoring applications

Abstract

Wireless sensor networks are used in many environmental monitoring applications (e.g., to monitor forest fires or volcanoes). In such applications, sensor nodes have a limited quantity of energy, but must operate for years without having their batteries changed. The main mechanism used to allow nodes to save energy is to sequence periods of activity and inactivity. However, the design of MAC and routing protocols for applications with low duty-cycle is still a challenge.

In this thesis, we proposed unsynchronized MAC and routing protocols for wireless sensor networks devoted to environmental monitoring applications. The main specificity of our protocols is that they are adapted to very low duty-cycle (less than 1 % for all nodes).

Our protocols are analyzed and compared to existing protocols by simulation and experimentation on TelosB nodes. Despite this low duty-cycle for all nodes, our protocols are able to achieve good performance, unlike other protocols in the literature, which are not adapted to these extreme conditions.

Keywords: wireless sensor networks, MAC protocols, routing protocols, low duty cycle, energy efficient, environmental monitoring.